Volume 34 Number 1 March 2010

ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Special Issue:

Computational Linguistics and its Applications

Guest Editor:

Yulia Ledeneva
Grigori Sidorov

1977

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Higher Education, Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Editorial

Computational Linguistics and its Applications

This special issue contains several interesting papers related to computational linguistics and its applications. The papers were carefully selected by the guest editors on the basis of peer reviews. We are happy that authors from various countries chose this forum for presenting their work: USA, Spain, Mexico, China, Germany, Hungary, India, Japan, Lithuania; submitting the high quality research results.

The first paper "Recent Advances in Computational Linguistics" by Yulia Ledeneva and Grigori Sidorov (Mexico), presents the view of the authors concerning some aspects of the current state of computational linguistics and its applications.

The paraphrase ability can be considered one of the important characteristics of the usage of natural language that proves the correct understanding of phrases. This interesting phenomenon is discussed in the paper "Paraphrase Identification using Weighted Dependencies and Word Semantics" by Mihai C. Lintean and Vasile Rus (USA). The authors analyze the paraphrase using syntactic and semantic (lexical) information. Evaluation data is presented.

Some important issues of automatic summarization are discussed in the paper "Challenging Issues of Automatic Summarization: Relevance Detection and Quality-based Evaluation" by Elena Lloret and Manuel Palomar (Spain). Namely, two related ideas are presented. First, it is shown that the code quantity principle (most important information) is applicable to automatic summarization. Second, an evaluation of quality of summaries is discussed.

Axel-Cyrille Ngonga Ngomo (Germany) presents in his paper "Low-Bias Extraction of Domain-Specific Concepts" a new approach for extraction of the domain specific terms that tries to be independent from the specific domain. The approach is based on graph clustering.

In the work of Alberto Téllez-Valero, Manuel Montes-y-Gómez, Luis Villaseñor-Pineda and Anselmo Peñas-Padilla (Mexico, Spain) "Towards Multi-Stream Question Answering using Answer Validation" a method of determining the correctness of automatically generated answers is discussed. The method uses the combination of several answering systems.

The paper by Asif Ekbal and Sivaji Bandyopadhyay (India) "Named Entity Recognition using Appropriate Unlabeled Data, Post-processing and Voting" evaluates the improvement of NER by using unlabeled data, post processing and weighted voting of various models.

In the paper "Assigning Library of Congress Classification Codes to Books Based Only on their Titles" by Ricardo Ávila-Argüelles, Hiram Calvo, Alexander Gelbukh, and Salvador Godoy-Calderón (Mexico and Japan), experiments are presented for book classification using very small amount of information, namely, the book title. Several measures of comparison are explored and encouraging results are reported.

An important problem of what is collocation and how it can be detected automatically using modern machine learning techniques is discussed in the paper "Automatic Identification of Lexical Units" by Vidas Daudaravicius (Lithuania).

The paper "Finding Maximal Sequential Patterns in Text Document Collections and Single Documents" whose authors are René García, J. Fco. Martínez-Trinidad, and J. Ariel Carrasco-Ochoa (Mexico), presents two efficient algorithms that allow for detection of all maximal sequential patterns in a text collection without the necessity of recomputing if a new document is added. Experiments that show that the algorithms are better than the state of the art methods are presented.

The paper "Grammar of ReALIS and the Implementation of its Dynamic Interpretation" written by Gábor Alberti and Judit Kleiber (Hungary) presents a grammar for formal discourse analysis and discusses its implementation using Hungarian as an example language.

The idea of using various classifiers simultaneously is explored in "Using Bagging and Boosting Techniques for Improving Coreference Resolution" by Smita Vemulapalli, Xiaoqiang Luo, John F. Pitrelli, and Imed Zitouni (USA). The area of application of these classifiers is coreference resolution. It is shown that the technique that uses a combination of classifiers outperforms single classifiers.

An interesting topic of temporal multi-document summarization, whose goal is to analyze the information according to its temporal perspective in various documents, is discussed in "Cascaded Regression Analysis based Temporal Multi-document Summarization" by Ruifang He, Bing Qin, Ting Liu, and Sheng Li (China). Macro and micro importance discriminative models are combined to form a cascaded regression analysis approach that is verified using several available dataset.

Yulia Ledeneva
Autonomous University of the State of Mexico
Santiago Tianguistenco, Mexico
yledeneva@yahoo.com

Grigori Sidorov
Center for Computing Research, National Polytechnic Institute
Mexico City, Mexico
www.cic.ipn.mx/~sidorov

Recent Advances in Computational Linguistics

Yulia Ledeneva
Autonomous University of the State of Mexico
Santiago Tianguistenco, Mexico
E-mail: yledeneva@yahoo.com

Grigori Sidorov
National Polytechnic Institute
D.F., Mexico
E-mail: sidorov@cic.ipn.mx

Keywords: computational linguistics, natural language processing, computer science, information retrieval, question answering, text summarization

Received: February 23, 2009

In this paper, we present an overview of recent advances in selected areas of computational linguistics. We discuss relation of traditional levels of language – phonetics/phonology, morphology, syntax, semantics, pragmatics, and discourse – to the areas of computational linguistics research. Then the discussion about the development of the systems of automatic morphological analysis is given. We present various morphological classifications of languages, discuss the models that are necessary for this type of systems, and then argue that an approach based on "analysis through generation" gives several advantages during development and the grammar models that are used. After this, we discuss some popular application areas like information retrieval, question answering, text summarization and text generation. Finally, usage of graph methods in computational linguistics is dealt with.

Povzetek: Podan je pregled računalniškega jezikoslovja.

1 Introduction

In this paper, we present an overview of recent advances in selected areas of computational linguistics (CL).

The objective of computational linguistics is to develop models of language that can be implemented in computers, i.e., the models with certain degree of formalism, and to develop applications that deal with computer tasks related to human language, like development of software for grammar correction, word sense disambiguation, compilation of dictionaries and corpora, intelligent information retrieval, automatic translation from one language to another, etc. Thus, computational linguistics has two sides: on the one hand, it is part of linguistics; on the other hand, it is part of computer science.

In this paper, we first discuss the relation of traditional levels of language – phonetics/phonology, morphology, syntax, semantics, pragmatics, and discourse – to the areas of computational linguistics research.

Then various issues related to automatic morphological analysis are presented. We remind morphological classification of languages and discuss the models that are necessary for development of the system of automatic morphological analysis. We argue that the usage of approach known as "analysis through generation" can significantly reduce the time and effort during development and allows for usage of intuitively clear grammar models.

After this we present most popular CL application areas:

– Information Retrieval (IR) consists of finding documents of an unstructured nature that satisfies an information need from within large collections of documents usually in local computer or in the Internet. This area overtakes traditional database searching, becoming the dominant form of information access. Now hundreds of millions of people use IR systems every day, when they use a web search engine or search their emails.
– Question Answering (QA) is a complex task that combines techniques from NLP, IR and machine learning. The main aim of QA is to localize the correct answer to a question written in natural language in a non-structured collection of documents. Systems of QA look like a search engine, where the input to the system is a question in natural language and the output is the answer to the question (not a list of entire documents like in IR).
– Text summarization is a popular application that allows for reduction of text size without significant loose of the content. Extractive and abstractive methods are briefly compared; resulting summary evaluation problem is addressed.
– Text generation consists in generation of coherent text from raw data, usually in a specific domain.

The paper finishes with discussion of application of graph methods in computational linguistics. Graph methods are widely used in modern research in CL. Text representation as graphs and graph ranking algorithms are discussed.

2 Levels of language and areas of computational linguistic research

Computational linguistic research is correlated with traditional levels of language that are commonly accepted in general linguistics. These levels are:

– Phonetics/phonology,
– Morphology,
– Syntax,
– Semantics,
– Pragmatics, and
– Discourse.

At the phonetic level, we analyze the phones (sounds), from two points of view: 1) as a physical phenomenon; here we are interested in its spectrum and other physical characteristics, 2) as an articulatory phenomenon, i.e., the position of the pronunciation organs that generate the specific sound (namely, the sound with specific physical characteristics). At the phonological level, we interpret these physical or articulatory features as phonological characteristics and their values. For example, the feature “vibration of the vocal cords” with the values “vibrating” or “not vibrating”; or the feature “mode of the obstacle” with the values like “explosive”, “sibilant”, “affricate”, etc. By phonological features we mean the features that depend on the given phonetic system, for example, long vs. short vowels are different phonemes in English, but they are not in many other languages, for example, Spanish, Russian, etc. So, the vowel duration is phonological feature in English and it is not in the mentioned languages.

The morphological level deals with word structure and grammar categories that exist in languages (or in the given language) and the expression of these grammar categories within words.

The syntactic level studies relations between words in sentences and functions of words in a sentence, like subject, direct object, etc.

The semantic level is related to the concept of meaning, its representation and description. Generally speaking, the meaning can be found at any other level, see below the discussion about the limits of the levels.

At the pragmatic level, the relationship between the meaning of the text and the real world is considered. For example, in indirect speech acts, when the phrase “*Can you pass me the salt*?” in fact is a polite mode of asking the salt, and it is not a question about a physical ability to pick it up.

And finally, the discourse level is related to analysis of the relationship between sentences in discourse. For example, at this level we can find the phenomenon of anaphora, when the task is to find out to which possible antecedent (noun) a pronoun refers; or phenomenon of ellipsis, when some substructure is omitted but can be restored by reader on the basis of the previous context.

Note that there are no strict criteria for level distinction: these levels are more like focus of research. That is why there are many intersections between levels, for example, the meaning, being part of the semantic level, can be observed at the syntactic or morphological levels, but still, if we focus on morphemes or syntactic constructions, though they have meaning, we will not consider them as belonging to the semantic level. If we consider the interpretation of syntactic relations or lexical meaning, then we deal with semantics.

Now, let us have a look at the computer side of computational linguistics. Among the most widely represented modern directions of research in computational linguistics we can mention:

1. Speech recognition and synthesis,
2. Morphological analysis of a variety of languages (say, morphological analysis in English is rather simple, but there are languages with much more complex morphological structure),
3. Grammar formalisms that allows for development of parsing programs,
4. Interpretation of syntactic relations as semantic roles,
5. Development of specialized lexical resources (say, WordNet or FrameNet),
6. Word sense disambiguation,
7. Automatic anaphora resolution, among others.

The correspondence between these directions of research and traditional linguistic levels is pretty obvious. For example, the research directions 4, 5, and 6 are attempts to invoke semantics in text analysis.

It should be mentioned that it is useful to distinguish between *methods* and *areas of research*. The areas of research are related to the mentioned language levels or to specific applications, see below. The methods of research are related to particular methods that are used. The tendency in modern computational linguistics as far as methods are concerned is to apply machine learning techniques accompanied with processing of huge amount of data, available usually in Internet. Note that each research area can have additional standard resources specific for this area.

Another important dichotomy is related to distinction of *procedural* and *declarative knowledge*, which in case of computational linguistics corresponds to the distinction between development of algorithms (or methods) and development of language resources (or data).

From the list of the seven mentioned research directions, number 5 (development of specialized lexical resources) represents a direct development of resources. The majority of other research directions are dedicated to methods. In case when methods are based on linguistic resources, we call these methods *knowledge rich*. If developed algorithms do not use any linguistic resource then we call them *knowledge poor*. Note that purely statistic algorithms are knowledge poor when they use raw data (raw corpora). If a statistic algorithm uses marked data, then it uses knowledge coded into a corpus, and, thus, it becomes knowledge rich.

Note that all these distinctions are basically tendencies, i.e., usually there is no clear representative of each member of a given class.

3 Automatic morphological analysis

As an example of implementation of a linguistic processing task let us discuss a problem of the automatic morphological analysis.

There exist many different models of the automatic morphological analysis for different languages. One of the most famous models is the two-level model (KIMMO) suggested by Kimmo Koskenniemi [Kos85]; there are other models that focus on different grammar phenomena or processing scheme [Gel00, Gel03, Hau99, Sid96, Sed01].

Morphological analysis is a procedure that has as an input a word form, and gives as an output a set of grammemes[1] that corresponds to this input. Sometimes, it is accompanied with normalization (lemmatization), when we also obtain lemma (normalized word form), that is usually presented as an entry in the dictionaries, for example, *take* is lemma for *took, taken, take, taking, takes*.

3.1 Morphological classification of languages

First of all, let us discuss the differences related to morphological structure of languages, because it is directly related to the morphological processing algorithms.

There are two well-known classifications of languages based on their morphological characteristics. The first one is centered on the predominant usage of auxiliary words vs. usage of affixes (grammar morphemes). According to this classification the languages can be analytic, when auxiliary words are predominant (say Chinese, English); or synthetic, when affixes are used in the majority of cases in the language (say Russian, Turk). Sometimes it is said that analytic languages have "poor" morphology in a sense that words do not have many affixes, while the synthetic languages have "rich" morphology. Again, these are tendencies, i.e., a language can have some deviations from the predominant tendency.

Sometimes, this classification is enriched by adding categories of isolating and polysynthetic languages. A language is called isolating if practically no affixes are used, like in Chinese. If we compare it with English, the latter still keeps using some affixes, for example, for plural in nouns, or for past indefinite tense in verbs. Polysynthetic languages are languages where not only affixes are added to a stem, but also other syntactically depending word stems. Thus, several lexical stems are combined within one word. An example of these languages is Chukchi or some North American Indian languages.

[1] *Grammeme* is a value of the grammar category, for example, *singular* and *plural* are grammemes of the category *number*, etc.

Other morphological classification of languages is applied to synthetic languages and it is based on the predominant morphological technique: agglutination vs. flexion. Here once again we are speaking about tendencies: a language can have some features from one class, and some features from the other class.

We say that the language is agglutinative if:

1. Each grammar morpheme expresses exactly one grammeme (value of a grammar category).
2. There are no stem alternations or stem alternations are subjected to very regular changes that do not presume any knowledge of specific stem type, for example, vowel harmony.
3. Morphemes are concatenated mechanically (agglutinated).
4. The stem usually exists as a separate word without any additional concatenation with affixes.

Examples of the agglutinative languages are Turk and the similar ones – Kazakh, Kyrgyz, etc., Hungarian.

On the other hand, inflective languages have the following features:

1. Each grammar morpheme can express various grammemes (values of grammar categories), for example, the flexion *-mos* in Spanish expresses grammemes of *person(first)* and *number(plural)*, among others. Usually, only one grammar morpheme exists in a word.

2. Stem alternations are not predictable. i.e., we should know if this specific stem should be alternated or not.

3. Morphemes can be concatenated with certain irregular morphonological processes at the connection point.

4. Stem usually does not exist without grammar morphemes. Note that in inflective languages it is common the usage of ∅ morpheme (empty morpheme), that expresses some grammemes by default (usually, most common grammemes like *singular, third person*, etc).

Examples of the inflective languages are Slavic languages (Russian, Czech, Polish, etc.).

So, different languages have different morphological tendencies. Computer methods of analysis that are perfectly suitable for languages with poor morphology (like English) or with agglutinative morphology (like Turk) can be not the best methods for inflective languages (like Russian).

Note that morphological system of inflective languages is finite and it is not very vast, so any method of analysis gives correct results, but not all methods are equally convenient and easy to implement.

The morphology of agglutinative languages is also finite, thought the number of possible word forms is much greater than in an inflective language. Still, since the agglutinative morphology is much more regular than the inflective morphology, it is easier to develop corresponding processing algorithms.

3.2 Models for automatic morphological analysis

Our next step is to find out what models or what types of models are necessary for morphological processing.

Speaking about automatic morphological analysis, we should take into consideration three types of models:

- Model of analysis (procedure of analysis).
- Model of grammar (morphology) of a given language. This model consists in assigning of grammar classes to words that define the unique word paradigm (=set of affixes and stem alternations).
- Computer implementation, i.e., the used formalism.

3.2.1 Model of analysis

As far as model of analysis is concerned, there are two main possibilities: store all word forms in a database or implement some processing algorithm.

One extreme point is storing all grammatical forms in a dictionary (database). Such method of analysis is known as “bag of words”. This method is useful for inflective languages, but it is not recommended for agglutinative or polysynthetic ones. Modern computers have the possibility of storing large databases containing all grammatical forms for inflective languages (a rough approximation for Spanish and Russian is 20 to 50 megabytes). Note that we anyway need an algorithm for generation of these word forms.

In our opinion, more sophisticated algorithms that allow for reducing the dictionary size to, say, 1 megabyte, are preferable. Indeed, a morphological analyzer is usually used together with a syntactic parser, semantic analyzer and reasoning or retrieval engine, so freeing physical memory for these modules is highly desirable – of course, the use of large virtual memory makes simultaneous access to very large data structures possible, but it does not make it faster.

Algorithmic (non-“bag-of-words”) solutions have a number of additional advantages. For example, such algorithms have the possibility to recognize unknown (new) words. This is a crucial feature for a morphological analyzer since new words constantly appear in the languages, not speaking of possible incompleteness of the dictionary.

Obviously, the algorithmic processing implies separation of possible flexion and possible stem in the input word form. Usually, we use programming cycle and try all possible divisions of the input word. After that, there are two possibilities for algorithmic processing. These possibilities are:

- Use straight forward processing, i.e., we use traditional algorithmic scheme of conditions and cycles, or
- Use a trick known as “analysis through generation” (see below), that saves a lot of work in code writing and allows the usage of grammar models oriented for generation. Note that all traditional grammar models are oriented to generation.

Another dichotomy related to the models of analysis is: store the stems with the corresponding grammar information in the dictionary or our algorithm will guess this grammar information. If we store the data then our algorithm will be exact. If we try to infer the grammar properties of a stem, then our algorithm will guess and it will not be exact, i.e., sometimes it will guess incorrectly.

3.2.2 Model of grammar

If we prefer the algorithmic processing, then we should have a model of grammar (morphology). It is necessary for defining what paradigms (sets of flexions and regular stem alternations) are associated with each word.

It is desirable to maintain an existing grammar model for a given language, if there is any, of course. It makes the algorithm development much easier and faster. Note that traditional grammar models are oriented to generation process. Usually, the speakers of a language consider these models intuitively clear.

As an alternative solution, we can develop an algorithm that will transform some traditional morphological description into a description that we would like to have for our algorithm. Note, that if we have an exact algorithm of conversion, these two models are equivalent in the sense that they represent the same information. Still, the grammar information is presented in different ways. Thus, from this point of view of a human, the traditional model usually is much more comprehensive. Let us remind that they are oriented to generation, while the purpose of the morphological analysis is analysis, i.e., it is a procedure with exactly opposite direction. We will show later that we can avoid developing a conversion algorithm using the approach of “analysis through generation”. This approach substitutes the process of analysis with the process of generation. Generally speaking, generation is much simpler than analysis because we do not have so many possible combinations to process.

3.2.3 Computer implementation

In our opinion, any computer implementation is acceptable. It can be direct programming, or finite state automata, or transducers, etc. All of them give equivalent results. Mainly, the choice of the implementation depends on the resources available for the development and the programming skills of the developers.

3.3 Method “analysis through generation”

In this section, we discuss how to develop an automatic morphological analysis system for an inflective language spending less effort and applying more intuitive and flexible morphological models. We show that the use of not so straightforward method – analysis through generation – can greatly simplify the analysis procedure and allows using morphological models that are much more similar to the traditional grammars.

"Analysis through generation" is an approach to analysis when some modules formulate hypothesis of analysis and other modules verify them using generation.

We first describe the suggested method (types of information, types of morphological models, etc.) and then briefly discuss its implementation with examples from Russian language.

The main idea of analysis through generation applied to morphological analysis is to avoid development of stem transformation rules in analysis and to use instead the generation module. Implementation of this idea requires storing in the morphological dictionary of all stems for each word with the corresponding information.

As we have mentioned, the main problem of automatic morphological analysis of inflective languages is usually stem alternations. The direct way to resolve this problem is constructing the rules that take into account all possible stem alternations during the analysis process; for example, for Russian the number of such rules is about a thousand [Mal85]. However, such rules do not have any correspondence in traditional grammars; in addition, they have no intuitive correspondence in language knowledge; finally, they are too many.

Another possibility to handle alternations is to store all stems in the dictionary, together with the information on their possible grammatical categories; this method was used for Russian [Gel92] and for Czech [Sed01]. We also adopt this possibility, but propose a different technique for treatment of grammatical information: our technique is dynamic while the techniques described in [Gel92, Sed01] are static. As we mentioned we use "analysis through generation" technique. The model based on this approach uses 50 grammar classes presented in the corresponding traditional grammars, while the systems that developed the algorithm for grammar classes' transformation ([Gel92], [Sed01]) had about 1,000 classes that do not have any intuitive correspondence in traditional grammars.

In the next subsections, we describe the types of morphological information we use; then we discuss the morphological models (and the corresponding algorithms) we have used to implement the method; and finally, we describe the functioning of our method: analysis, generation, and treatment of unknown (new) words.

3.3.1 Types of grammatical information

We use two types of grammatical information:

– Stem dictionary and
– List of grammatical categories and corresponding grammemes.

The information about the stems is stored in the morphological dictionary. This information is basically the data needed for generation, such as:

– Part of speech,
– Presence of alternations,
– Grammatical type (in Russian, there are three genders and for each gender there are several word formation types: say, for feminine there are 7 types, etc.),
– Special marks: for example, in Russian some nouns have two forms of the prepositional case (*шкафу* versus *шкафе* '(in) wardrobe' versus '(about) wardrobe'), which should be marked in the dictionary.

We explicitly store in the dictionary all variants of stems as independent forms, together with the stem number (first stem, second stem, etc.). In Russian, nouns and adjectives with alternations have two possible stems, while verbs can have up to four stems.

Another type of information is a list of grammatical categories and corresponding grammemes. Thus, any word form is characterized by a set of grammemes. For example, for a Russian noun this set contains a value of case and of number; for a Russian full adjective it is a value of case, number, and gender, etc.

3.3.2 Types of morphological models

Three morphological models are used:

– Correspondence between flexions and grammemes,
– Correspondence between stems and grammemes,
– Correspondence between alternating stems of the same lexeme.

The first model establishes the correspondence between flexions and sets of grammemes, taking into account different grammatical types fixed in the dictionary. In the process of analysis, we use the correspondence "flexions $\Rightarrow$ sets of grammemes", that is used to formulate hypothesis; and in the process of generation, the correspondence "sets of grammemes $\Rightarrow$ flexions", that is used to verify hypothesis.

A similar correspondence is established between the sets of grammemes and the types of stems; however, this correspondence is used only for generation. For example, if a Russian masculine noun of a certain grammar type has a stem alternation, then the first stem is used for all forms except for genitive (case) plural, for which the second stem is used. Note that corresponding model for analysis is unnecessary, which makes our method simpler than direct analysis.

To be able to generate all forms starting from a given form, it is necessary to be able to obtain all variants of stems from the given stem. There are two ways to do this: static and dynamic, which have their own pros and contras. The static method implies storing in the dictionary together with the stems the correspondence between them (e.g., each stem has a unique identifier by which stems are linked within the dictionary). Storing the explicit links increases the size of the dictionary.

We propose to do this dynamically. Namely, the algorithm of constructing all stems from a given stem is to be implemented. In fact, it must be implemented anyway since it is used to compile the dictionary of stems. It is sufficient to develop the algorithm for constructing the first stem (that corresponds to the normalized form, such as infinitive) from any other stem, and any other stem from the first stem. In this way, starting from any stem we can generate any other stem. The difference between static and dynamic method is that in the former case, the algorithm is applied during the compile time (when the dictionary is built), while in the latter case, during runtime.

Note that the rules of these algorithms are different from the rules that have to be developed to implement analysis directly. For Russian, we use about 50 rules, intuitively so clear that in fact any person learning Russian is aware of these rules of stem construction. Here is an example of a stem transformation rule:

-VC, * ⇒ -C

which means: if the stem ends in a vowel (V) following by a consonant (C) and the stem type contains the symbol "*" then remove this vowel. Being applied to the first stem of the noun *молоток* 'hammer', the rule gives the stem *молотк-*(*а*) 'of hammer'.

This contrasts with about 1,000 rules necessary for direct analysis, which in addition are very superficial and anti-intuitive. For example, to analyze a non-first-stem word, [Mal85] uses rules that try to invert the effect of the mentioned rule: if the stem ends in a consonant, try to insert a vowel before it and look up each resulting hypothetical stem in the dictionary: for *молотк-*(*а*), try *молотек-*, *молоток-*, etc. This also slows down the system performance.

Two considerations are related to the simplicity of our rules. First, we use the information about the type of the stem stored in the dictionary. Second, often generation of a non-first stem from the first one is simpler than vice versa. More precisely, the stem that appears in the dictionaries for a given language is the one that allows simpler generation of other stems.

3.3.3 Data preparation

Our method needs some preliminary work of data preparation, carrying out the following main steps:

– Describe and classify all words of the given language into unique grammatical classes (fortunately, for many languages this work is already done by traditional grammar writers);
– Convert the information about words into a stem dictionary (generating only the first stem);
– Apply the algorithms of stem generation (from the first stem to other stems) to generate all stems;
– Generate the special marks and the stem numbers for each (non-first) stem.

To perform the last two steps, the dictionary record generated for the first stem is duplicated, the stem is transformed into the required non-first stem, and the mark with the stem number is added.

3.3.4 Generation process

The generation process is simple. Given the data from the dictionary (including the stem and its number) and a set of grammemes, it is required to build a word form of the same lexeme that has the given set of grammemes. Using the models we have constructed, the flexion is chosen and the necessary stem is generated (if a non-first stem was given, then we generate the first stem and from it, the necessary stem). Finally, we concatenate the stem and the flexion.

If necessary, this process is repeated several times for adding more than one flexion to the stem. For example, Russian participles (which are verbal forms) have the same flexions as adjectives (which express the number and gender) and also special suffixes (which indicate that this is a participle), i.e., they are concatenation of a stem and two affixes: a suffix and a flexion (*пис-ать* → *пиш-ущ-ий* 'writ-e' → 'writt-en'). In this case, we first generate the stem of participle by adding the suffix (we use the information from the dictionary on the properties of the corresponding verbal stem) and then change the dictionary information to the information for an adjective of the corresponding type. In case of Russian, such recursion is limited to three levels (one more level is added due to the reflexive verbs that have a postfix morpheme *-ся*: *пиш-ущ-ий-ся* 'is written').

3.3.5 Analysis process

Given an input string (a word form), we analyze it in the following way:

1. The letters are separated one by one from right to left to get the possible flexion (the zero flexion is tried at first): given *stopping*, we try -∅ (zero flexion); at the next iterations *-g*, *-ng*, *-ing*, *-ping*, etc. are tried.
2. If the flexion (here *-ing*) is found in the list of possible flexions, we apply the algorithm "flexions ⇒ sets of grammemes", which gives us a hypothesis about the possible set of grammemes. Here it would be "verb, participle".
3. Then we obtain the information for the rest of the form, i.e., the potential stem, here *stopp-* from the stem dictionary.
4. Finally, we generate the corresponding grammatical form according to our hypothesis and the obtained dictionary information. Here, the generated past participle of the verbal stem *stopp-* is *stopping*.
5. If the obtained result coincides with the input form, then the hypothesis is accepted. Otherwise, the process repeats from the step 1.

If a word form consists of several morphemes (a stem and several affixes), then the analysis process is recursive, precisely as generation. In case of Russian, there are tree levels of recursion.

As one can see, our method of analysis is simple and invokes generation. Additional modules are the model "flexions ⇒ sets of grammemes" and the module of interaction between different models.

3.3.6 Treatment of unknown words

The treatment of unknown words is also simple. We apply the same procedure of analysis to single out the hypothetical stem. If the stem is not found in the dictionary, we use the longest match stem (matching the strings from right to left) compatible with the given set of affixes. The longest match stem is the stem present in the dictionary that has as long as possible *ending* substring in common with the given stem (and is compatible with already separated affixes).

In this way, for example, an (unknown) input string *sortifies* will be analyzed as *classifies*: verb, 3rd person, present, singular, given that *classifi-* is its longest match stem for *sortifi-* (matching by *-ifi-*) compatible with the affix *-es*.

To facilitate this search, we have another instance of the stem dictionary in inverse order, i.e., stems are ordered lexicographically from right to left.

Note that the systems like [Gel00, Gel92] based on the left-to-right order of analysis (first separating the stem and only then analyzing the resting affixes) have to imitate this process with a special dictionary of, say, a list of 5-letter stem endings, since in such systems the main stem dictionary is ordered by direct order (left to right, by first letters).

4 Computational linguistic applications

This section is divided into following subsections that correspond to each application of CL: Information Retrieval, Question Answering, Text Summarization, and Text Generation.

4.1 Information retrieval

Information Retrieval (IR) according to [Man07, Bae99] consists of finding documents of an unstructured nature that satisfies an information need within large collections of documents usually on a computer or on the internet. This area overtakes traditional database searching, becoming the dominant form of information access. Now hundreds of millions of people use IR systems every day when they use a web search engine or search their emails.

The huge amount of available electronic documents in Internet has motivated the development of very good information retrieval systems, see NTCIR (NII Test Collection for Information Retrieval) and Cross-Language Evaluation Forum (CLEF) web pages.

Information Retrieval models represent documents or collection of documents by weighting terms appearing in each document. Then, two directions are traced for further advances: new methods for weighting and new methods for term selection.

First, we look through classical models briefly, and then we describe recently proposed methods. IR classical models are:

– Vector Space Model [Sal88],
– Model based on term frequency (*tf*) [Luh57].
– Model based on inverse document frequency (*idf*) [Sal88],
– Model based *tf-idf* [Sal88],
– Probabilistic Models [Fuhr92],
– Transition Point [Pin06],
– n-grams [Man99].

Recent improvements to the following models should be mentioned. As far as term selection is concerned: MFS [Gar04, Gar06], Collocations [Bol04a, Bol04b, Bol05, Bol08], Passages [Yin07].

As far as weighting of terms is concerned: Entropy, Transition Point enrichment approach.

Various tasks where IR methods can be used are:

– Monolingual Document Retrieval,
– Multilingual Document Retrieval,
– Interactive Cross-Language Retrieval,
– Cross-Language Image, Speech and Video Retrieval,
– Cross-Language Geographical Information Retrieval,
– Domain-Specific Data Retrieval (Web, Medical, Scientific digital corpora) [Van08].

4.2 Question answering

Question Answering (QA) retrieves the correct answer to a question written in natural language from collection of documents. Systems of QA look like a search engine where the input to the system is a question in natural language and the output is the answer to the question.

The main goals of the state-of-the-art systems are targeted to improve QA systems performance, help humans in the assessment of QA systems output, improve systems self-score, develop better criteria for collaborative systems, deal with different types of questions.

There are several workshops and forums where main tasks of QA are proposed and discussed. For example, researchers compete to find the best solution for the following tasks proposed in Cross-Language Evaluation Forum (CLEF), NTCIR (NII Test Collection for Information Retrieval) and Text REtrieval Conference (TREC): Monolingual task, Multilingual task, Cross-Language task, Robust task.

And more specific tasks:

– Answer validation task,
– QA over speech transcription of seminars, meetings, telephone conversations, etc.
– QA on speech transcript where the answers to factual questions are extracted from spontaneous speech transcriptions.
– QA using machine translation systems,
– QA for “Other” questions, i.e. retrieval of other interesting facts about a topic,
– Time-constrained task (realized in real time),
– QA using Wikipedia,
– Event-targeted task on a heterogeneous document collection of news article and Wikipedia,
– QA using document collections with already disambiguated word senses in order to study their contribution to QA performance,
– QA using passage retrieval systems, etc.

Each task can propose different solutions depending on the question category. Actually, the following categories are considered: factoid, definition, closed list and topic-related. Factoid questions are fact-based questions, asking for the name of a person, location, organization, time, measure, count, object, the extent of something, the day on which something happened. Definition question are questions such as "*What/Who is*?", and are divided into the following subtypes: person, organization, object, and "other" questions. Closed list questions are questions that require in one single answer the requested number of items. Such questions may contain a temporal restriction. Topic related questions group questions which are related to the same topic and possibly contain co-references between one question and the others. Topics can be named entities, events, objects, natural phenomena, etc.

Answer validation task develops and evaluates a special module which validates the correctness of the answers given by a QA system. The basic idea is that once a pair (answer and snippet) is returned by a QA system, a hypothesis is built by turning the pair (question and answer) into an affirmative form. If the related text semantically entails this hypothesis, then the answer is expected to be correct [Peñ06, Peñ07, Peñ08, Tel08].

Machine Translation Systems are broadly implemented for Cross-Language QA [Ace06]. In recent studies, the negative effect of machine translation on the accuracy of Cross-Language QA was demonstrated. [Fer07]. As a result, Cross-Language QA Systems are modified [Ace07, Ace09].

QA over speech transcription provides a framework in which QA systems can be evaluated in a real scenario, where the answers of oral and written questions (factual and definitional) in different languages have to be extracted from speech transcriptions (manual and automatic transcriptions) in the respective language. The particular scenario consists in answering oral and written questions related to speech presentations. As an example, QA system automatically answers in Chinese about travel information. This system integrates a user interface, speech synthesis and recognition, question analysis, QA database retrieval, document processing and preprocessing, and some databases [Hu06].

QA systems for "Other" questions generally use question generation techniques, predetermine patterns, interesting keywords, combination of methods based on patterns and keywords, or exploring external knowledge sources like nuggets [Voo04a, Voo04b, Voo04c, Voo05a, Voo05b, TREC, Raz07].

4.3 Text summarization

Information retrieval systems (for example, *Google*) show part of the text where the words of the query appears. With the extracted part, the user has to decide if a document is interesting even if this part does not have useful information for the user, so it is necessary download and read each retrieved document until the user finds satisfactory information. A solution for such problem is to extract the important parts of the document which is the task of automatic text summarization.

More applications of automatic text summarization are, for example, summaries of news and scientific articles, summaries of electronic mails, summaries of different electronic information which later can be sent as SMS, summaries of found documents and pages returned by a retrieved system.

From one side, there is a single-document summarization which implies to communicate the principal information of one specific document, and from another side—a multi-document summarization which transmits the main ideas of a collection of documents. There are two options to achieve a summarization by computer: text abstraction and text extraction [Lin97]. Text abstraction examines a given text using linguistic methods which interpret a text and find new concepts to describe it. And then new text is generated which will be shorter with the same content of information. Text extraction means extract parts (words, sequences, sentences, paragraphs, etc.) of a given text based on statistic, linguistic or heuristic methods, and then join them to new text which will be shorter with the same content of information.

According to the classical point of view, there are three stages in automated text summarization [Hov03]. The first stage is performed by topic identification where almost all systems employ several independent modules. Each module assigns a score to each unit of input (word, sentence, or longer passage); then a combination module combines the scores for each unit to assign a single integrates score to it; finally, the system returns the n highest-scoring units, according to the summary length requested by the user. The performance of topic identification modules is usually measured using Recall and Precision scores.

The second stage denotes as the stage of interpretation. This stage distinguishes extract-type summarization systems from abstract-type systems. During the interpretation the topics identified as important are fused, represented in new terms, and expressed using a new formulation, using concepts or words not found in the original text. No system can perform interpretation without prior knowledge about the domain; by definition, it must interpret the input in term of something extraneous to the text. But acquisition deep enough prior domain knowledge is so difficult that summarizers to date have only attempted it in a small way. So, the disadvantage of this stage remains blocked by the problem of domain knowledge acquisition.

Summary generation is the third stage of text summarization. When the summary content has been created in internal notation, and thus requires the techniques of natural language generation, namely text planning, sentence planning, and sentence realization.

In 2008, new scheme was proposed by Ledeneva [Led08a] which include four steps for composing a text summary:

– Term selection: during this step one should decide what units will count as terms are, for example, they can be words, n-grams or phrases.

– Term weighting: this is a process of weighting (or estimating) individual terms.
– Sentence weighting: the process of assigning numerical measure of usefulness to the sentence. For example, one of the ways to estimate the usefulness of a sentence is to sum up usefulness weights of individual terms of which the sentence consists.
– Sentence selection: selects sentences (or other units selected as final parts of a summary). For example, one of the ways to select the appropriate sentences is to assign some numerical measure of usefulness of a sentence for the summary and then select the best ones.

4.3.1 Extractive text summarization methods

Most works appeared in recent researches are based on looking for appropriate terms. The most used option is select words as terms; however is not the only possible option. Liu *et al*. [Liu06] uses pairs of syntactically connected words (basic elements) as atomic features (terms). Such pairs (which can be thought of as arcs in the syntactic dependency tree of the sentence) have been shown to be more precise semantic units than words [Kos04]. However, while we believe that trying text units larger than a word is a good idea, extracting the basic elements from the text requires dependency syntactic parsing, which is language-dependent. Simpler statistical methods (cf. the use of n-grams as terms in [Vil06]) may prove to be more robust and language-independent.

Some approaches of text summaries match semantic units such as elementary discourse units [Mar01, Sor03], factoids [Teu04a, Teu04b], information nuggets [Voo04], basic elements [Liu06], etc. A big disadvantage of these semantic units is that the detection of these units is realized manually. For example, information nuggets are atomic pieces of interesting information about the target identified by human annotators as vital (required) or non-vital (acceptable but not required) for the understanding of the content of a summary.

Factoids are semantic units which represent the meaning of a sentence. For instance, the sentence “The police have arrested a white Dutch man” by the union of the following factoids: “A suspect was arrested”, “The police did the arresting”, “The suspect is white”, “The suspect is Dutch”, “The suspect is male”. Factoids are defined empirically based on the data in the set of summaries. Usually they are manually made summaries taken from [Duc]. Factoid definition starts with the comparison of the information contained in two summaries, and factoids get added or split as incrementally other summaries are considered. If two pieces of information occur together in all summaries and within the same sentence, they are treated as one factoid, because differentiation into more than one factoid would not help us in distinguishing the summaries. Factoids are labeled with descriptions in natural language; initially, these are close in wording to the factoid's occurrence in the first summaries, though the annotator tries to identify and treat equally paraphrases of the factoid information when they occur in other summaries. If (together with various statements in other summaries) one summary contains “was killed” and another “was shot dead”, we identify the factoids: “There was an attack”, “The victim died”, “A gun was used”. The first summary contains only the first two factoids, whereas the second contains all three. That way, the semantic similarity between related sentences can be expressed. When factoids are identified in the collection of summaries, most factoids turned out to be independent of each other. But when dealing with naturally occurring documents many difficult cases appear, e.g. ambiguous expressions, slight differences in numbers and meaning, and inference.

The text is segmented in Elementary Discourse Units (EDUs) or non-overlapping segments, generally taken as clauses or clauses like units of a rhetorical relation that holds between two adjacent spans of text [Mar01, Car03]. The boundaries of EDUs are determined using grammatical, lexical, syntactic information of the whole sentence.

Other possible option proposed by Nenkova in [Nen06] is Semantic Content Units (SCUs). The definition of the content unit is somewhat fluid, it can be a single word but it is never bigger than a sentence clause. The most important evidence of their presence in a text is the information expressed in two or more summaries, or in other words, is the frequency of the content unit in a text. Other evidence is that these frequent content units can have different wording (but the same semantic meaning) what brings difficulties for language-independent solution.

The concept of lexical chains was first introduced by Morris and Hirst. Basically, lexical chains exploit the cohesion among an arbitrary number of related words [Mor91]. Then, lexical chains are computed in a source document by grouping (chaining) sets of words that are semantically related (i.e. have a sense flow) [Bar99, Sil02]. Identities, synonyms, and hypernym/hyponyms are the relations among words that might cause them to be grouped into the same lexical chain. Specifically, words may be grouped when:

Two noun instances are identical, and are used in the same sense. (*The house on the hill is large. The house is made of wood.*)

Two noun instances are used in the same sense (i.e., are synonyms). (*The car is fast. My automobile is faster.*)

The senses of two noun instances have a hypernym/hyponym relation between them. (*John owns a car. It is a Toyota*.)

The senses of two noun instances are siblings in the hypernym/hyponym tree. (*The truck is fast. The car is faster.*)

In computing lexical chains, the noun instances were grouped according to the above relations, but each noun instance must belong to exactly one lexical chain. There are several difficulties in determining which lexical chain a particular word instance should join. For instance, a particular noun instance may correspond to several different word senses and thus the system must determine which sense to use (e.g. should a particular instance of “house” be interpreted as sense 1: dwelling or sense 2: legislature). In addition, even if the word sense of an instance can be determined, it may be possible to group that instance into several different lexical chains because it may be related to words in different chains. For example, the word's sense may be identical to that of a word instance in one grouping while having a hypernym/hyponym relationship with that of a word instance in another. What must happen is that the words must be grouped in such a way that the overall grouping is optimal in that it creates the longest/strongest lexical chains. It was observed that contention that words are grouped into a single chain when they are "about" the same underlying concept. That fact confirms the usage of lexical chains in text summarization [Bru01, Zho05, Li07].

Keyphrases, also known as keywords, are linguistic units, usually, longer than a words but shorter than a full sentence. There are several kinds of keyphrases ranging from statistical motivated keyphrases (sequences of words) to more linguistically motivated ones (that are defined in according to a grammar). In keyphrases extraction task, keyphrases are selected from the body of the input document, without a predefined list. Following this approach, a document is treated as a set of candidate phrases and the task is to classify each candidate phrase as either a keyphrase or nonkeyphrase [Dav07]. When authors assign keyphrases without a controlled vocabulary (free text keywords or free index terms), about 70% to 80% of their keyphrases typically appear somewhere in the body of their documents [Dav07]. This suggests the possibility of using author-assigned free-text keyphrases to train a keyphrases extraction system.

D'Avanzo [Dav07] extracts syntactic patterns using two ways. The first way focuses on extracting uni-grams and bi-grams (for instance, noun, and sequences of adjective and noun, etc.) to describe a precise and well defined entity. The second way considers longer sequences of part of speech, often containing verbal forms (for instance, noun plus verb plus adjective plus noun) to describe concise events/situations. Once all the uni-grams, bi-grams, tri-grams, and four-grams are extracted from the linguistic pre-processor, they are filtered with the patterns defined above. The result of this process is a set of patterns that may represent the current document.

For multi-document summarization, passages are retrieved using a language model [Yin07]. The goal of language modeling is to predict the probability of natural word sequences; or in other words, to put high probability on word sequences those actually occur and low probability on word sequences that never occur. The simplest and most successful basis for language modeling is the n-gram model.

4.3.2 Abstractive text summarization methods

Abstractive summarization approaches use information extraction, ontological information, information fusion, and compression. Automatically generated abstracts (abstractive summaries) moves the summarization field from the use of purely extractive methods to the generation of abstracts that contain sentences not found in any of the input documents and can synthesize information across sources. An abstract contains at least some sentences (or phrases) that do not exist in the original document. Of course, true abstraction involves taking the process one step further. Abstraction involves recognizing that a set of extracted passages together constitute something new, something that is not explicitly mentioned in the source, and then replacing them in the summary with the new concepts. The requirement that the new material not be in the text explicitly means that the system must have access to external information of some kind, such as an ontology or a knowledge base, and be able to perform combinatory inference.

Recently, Ledeneva *et al*. [Led08a, Led08b, Led08c] and Garcia *et al*. [Gar08a, Gar08b, Gar09] have successfully employed the word sequences from the self-text for detecting the candidate text fragments for composing the summary.

Ledeneva *et al*. [Led08a] suggest a typical automatic extractive summarization approach composed by term selection, term weighting, sentence weighting and sentence selection steps. One of the ways to select the appropriate sentences is to assign some numerical measure of usefulness of a sentence for the summary and then select the best ones; the process of assigning these usefulness weights is called sentence weighting. One of the ways to estimate the usefulness of a sentence is to sum up usefulness weights of individual terms of which the sentence consists; the process of estimating the individual terms is called term weighting. For this, one should decide what the terms are: for example, they can be words; deciding what objects will count as terms is the task of term selection. Different extractive summarization methods can be characterized by how they perform these tasks.

Ledeneva *et al*. [Led08a, Led08b, Led08c] has proposed to extract all the frequent grams from the self-text, but she only considers those that are not contained (as subsequence) in other frequent grams (maximal frequent word sequences). In comparison with n-grams, the Maximal Frequent Sequences (MFS) are attractive for extractive text summarization since it is not necessary to define the gram size (n), it means, the length of each MFS is determined by the self-text. Moreover, the set of all extracted MFSs is a compact representation all frequent word sequences, reducing in this way the dimensionality in a vector space model.

Garcia *et al*. [Gar08b, Gar09] have extracted all the sequences of n words (n-grams) from the self-text as features of its model. In this work, we evaluate the n-grams and maximal frequent sequences as domain- and language- independent models for automatic text summarization. In this work, sentences were extracted using unsupervised learning approach.

Some other methods are also developed for abstractive summarization. For example, techniques of sentence fusion [Dau04, Bar03, Bar05], information fusion [Bar99], sentence compression [Van04, Mad07], headline summarization [Sar05], etc.

4.3.3 Recent applications of text summarization

We should mention some systems base on summarization for the following applications:

- Legal texts [Far04, Har04],
- Emails [Cor04, Shr04, Wan04],
- Web pages [Dia06],
- Web documents using mobile devices [Ott06],
- Figures and graphics [Fut04, Car04, Car06],
- News [Eva05, Mck03, Nen05a].

4.3.4 Methods for evaluation of summaries

Up to date, the most recent evaluation system is ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*). ROUGE [Lin03a] was proposed by Lin and Hovy [Lin04a, Lin04b, Lin04c]. This system calculates the quality of a summary generated automatically by comparing to the summary (or several summaries) created by humans. Specifically, it counts the number of overlapping different units such as word sequences, word pairs and n-grams between the computer-generated summary to be evaluated and the ideal summaries created by humans. ROUGE includes several automatic evaluation measures, such as ROUGE-N (n-grams co-occurrence); ROUGE-L (longest subsequence); ROUGE-W (weighted longest subsequence); ROUGE-S (skip-bigram co-occurrence). For each of the measures (ROUGE-N, ROUGE-L, etc.), ROUGE returns Recall, Precision and F-measure scores.

Another evaluation schemes was proposed by Nenkova *et al*. [Nen04, Pas05, Nen06]. In this scheme, special terms are annotated using the pyramid scheme—a procedure specifically designed for comparative analysis of the content of several texts. The idea of this scheme is to evaluate presence of each term in all documents of the collection. The more documents contain the term, the more important is this term, and consequently it will have higher score.

4.4 Text generation

Text Generation (TG) automatically produces linguistically correct texts from a rough data that represent information in a specific domain, and that are organized in conventional databases, knowledge bases, or even being produced as result of some application processing.

Text generation process is traditionally seen as a goal-driven communication process. As a consequence, the final text, being written or spoken, just a single-clause or a multi-paragraph document, is always an attempt to address some communicative goal. Starting from a communicative goal, the generator decides which information from the original data source should be conveyed in the generated text. During the generation process, the communicative goal is refined in more specific sub-goals and some kind of planning takes place to progressively convert them together with the original data to a well-formed and linguistically correct final text.

The whole generation process is traditionally organized in three specific tasks:

- *Content determination* is the task of deciding which chunks of content, collected from the input data source, will make up the final text. Each chunk of content represents an indivisible information unit. These content units are usually grouped in a semantic unit of higher complexity for a given application domain. A semantic unit is called message. Considering for instance a system that generates soccer reports, the sentences "Brazilian soccer team has beaten the Argentines last Sunday" and "Sunday soccer report: Victory of Brazil over Argentine" represent different linguistic constructions for the same kind of message: "Victory".
- *Content organization* groups the generated messages appropriately as units for each level of linguistic hierarchy: the paragraph, the sentence and the phrase. In addition, it defines element ordering within a group for each respective level. Finally, it is in charge of specifying coordination and subordination dependencies between these groupings.
- *Surface realization* is the task of choosing the appropriated term and the syntactic construction for each content unit. This choice is constrained by lexical and grammatical rules of the language. Punctuation symbols are defined at this stage as well.

The applications of this area are usually built using ad-hoc software engineering practices, lacking a well-defined development process, standard software architecture, and the use of worldwide programming languages. A lot of researches have clarified many fundamentals issues and conceived solutions that are robust and scalable enough for practical use [Fon08].

Furthermore, opportunities for practical applications have multiplied with the information inundation from relevant Web content sources. Unfortunately, TG techniques remain virtually unknown and unused by mainstream and professional computing. This situation is probably due mainly to the fact that until recently, TG was built using ad-hoc software engineering practices with no explicit development process and no standard software architecture. Reliance on special-purpose esoteric modeling and implementation languages and tools is another TG issue. Every system is designed and implemented following specific domain complexities and needs and little has been done to change the portrayed situation. Many realization components have been built based on different grammatical formalisms and theories used to describe TG [Elh92].

Recent work [Fon08] describes a new development approach that leverages the most recent programming languages and standards of modern software engineering to enhance the practical use of TG applications. This work proposes an innovative approach to the development of TG systems, in which the pipeline of text generation tasks work as a set of consecutive rule base for model transformation. Such methodology for building applications by applying transformations on models in different levels of abstraction was recently popularized as a new software engineering paradigm [Omg01].

5 Graph methods

Graph methods are particularly relevant in the area of CL. Many language processing applications can be modeled by means of a graph. These data structures have the ability to encode in a natural way the meaning and structure of a cohesive text, and follow closely the associative or semantic memory representations.

One of the most important methods is TextRank [Mih04, Mih06]. TextRank has been successfully applied to three natural language processing tasks: keyword extraction [Mih04], document summarization [Mih06], word sense disambiguation [Mih06], and text classification [Has07] with results competitive with those of state-of-the-art systems. The strength of the model lies in the global representation of the context and its ability to model how the co-occurrence between features might propagate across the context and affect other distant features.

5.1 Graph representation of text

To enable the application of graph-based ranking algorithms to natural language texts, a graph that represents the text is built, and interconnects words or other text entities with meaningful relations. The graphs constructed in this way are centered around the target text, but can be extended with external graphs, such as off-the-shelf semantic or associative networks, or other similar structures automatically derived from large corpora.

Graph Nodes: Depending on the application at hand, text units of various sizes and characteristics can be added as vertices in the graph, e.g. words, collocations, word senses, entire sentences, entire documents, or others. Note that the graph-nodes do not have to belong to the same category.

Graph Edges: Similarly, it is the application that dictates the type of relations that are used to draw connections between any two such vertices, e.g. lexical or semantic relations, measures of text cohesiveness, contextual overlap, membership of a word in a sentence, and others.

Algorithm: Regardless of the type and characteristics of the elements added to the graph, the application of the ranking algorithms to natural language texts consists of the following main steps:

– Identify text units that best define the task at hand, and add them as vertices in the graph.
– Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.
– Apply a graph-based ranking algorithm to find a ranking over the nodes in the graph. Iterate the graph-based ranking algorithm until convergence. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.

5.2 Graph ranking algorithms

The basic idea implemented by a random-walk algorithm is that of “voting” or “recommendation.” When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex.

Moreover, the importance of the vertex casting a vote determines how important the vote itself is, and this information is also taken into account by the ranking algorithm. While there are several random-walk algorithms that have been proposed in the past, we focus on only one such algorithm, namely PageRank [Bri98], as it was previously found successful in a number of applications, including Web link analysis, social networks, citation analysis, and more recently in several text processing applications.

Given a graph G = (V, E), let In(V_i) be the set of vertices that point to vertex V_i (predecessors), and Out(V_i) be the set of vertices that vertex V_i points to (successors). The PageRank score associated with the vertex V_i is defined using a recursive function that integrates the scores of its predecessors:

$$S(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{S(V_j)}{|Out(V_j)|} \tag{1}$$

where d is a parameter that is set between 0 and 1.

The score of each vertex is recalculated upon each iteration based on the new weights that the neighboring vertices have accumulated. The algorithm terminates when the convergence point is reached for all the vertices, meaning that the error rate for each vertex falls below a pre-defined threshold.

This vertex scoring scheme is based on a random-walk model, where a walker takes random steps on the graph, with the walk being modeled as a Markov process. Under certain conditions (when the graph is acyclic and irreducible) the model is guaranteed to converge to a stationary distribution of probabilities associated with the vertices in the graph. Intuitively, the stationary probability associated with a vertex represents the probability of finding the walker at that vertex during the random-walk, and thus it represents the importance of the vertex within the graph.

Two of the most used algorithms are PageRank [Bri98] and HITS (Hyperlinked Induced Topic Search) [Kle99].

Undirected Graphs: Although traditionally applied on directed graphs, algorithms for node activation or ranking can be also applied to undirected graphs. In such graphs, convergence is usually achieved after a larger number of iterations, and the final ranking can differ significantly compared to the ranking obtained on directed graphs.

Weighted Graphs: When the graphs are built from natural language texts, they may include multiple or partial links between the units (vertices) that are extracted from text. It may be therefore useful to indicate and incorporate into the model the "strength" of the connection between two vertices V_i and V_j as a weight w_{ij} added to the corresponding edge that connects the two vertices. Consequently, we introduce new formulae for graph-based ranking that take into account edge weights when computing the score associated with a vertex in the graph.

5.3 Graph clustering algorithms

The main purpose of graph clustering algorithms is calculates clusters for large graphs and to extract concepts from similar graphs. These algorithms can be applied in various computational linguistics applications. For example, word sense disambiguation [Sch98], lexical acquisition [Ngo08], language separation [Bie06], taxonomy [Ngo09] and ontology extraction [Ngo09], etc.

The idea of graph clustering algorithm [Ngo09] is to maximize the flow from the border of each cluster to the nodes within the cluster while minimizing the flow from the cluster to the nodes outside of the cluster. The algorithm uses local information for clustering and archives a soft clustering of the input graph. The first advantage of this algorithm consists in efficiently handling large graphs which permits to obtain promising results for computational linguistics applications. The second advantage is that it can be used to extract domain-specific concepts from different corpora and show that it computes concepts of high purity.

6 Conclusions

In this paper, we presented an overview of recent advances in selected areas of computational linguistics. We discussed relation of traditional levels of language – phonetics/phonology, morphology, syntax, semantics, pragmatics, and discourse – to the areas of computational linguistics research.

The discussion about the development of the systems of automatic morphological analysis was given. We presented various morphological classifications of languages, discussed the models that are necessary for this type of systems, and then showed that an approach based on "analysis through generation" gives several advantages during development and the grammar models that are used.

After this, we discussed some popular application areas like information retrieval, question answering, text summarization and text generation.

Finally, he paper dealt with the usage of graph methods in computational linguistics.

7 References

[Ace06] Aceves-Perez R., Montes-y-Goméz M., Villaseñor-Pineda L. Using n-grams models to Combine Query Translations in Cross-Language Question Answering. Springer Verlag 3878, CICLing 2006.

[Ace07] Aceves-Peréz R., Montes y Gómez M., Villaseñor Pineda L. Enhancing Cross-Language Question Answering by Combining Multiple Question Translations. Lecture Notes in Computer Science, Springer-Verlag, vol. 4394, pp. 485–493, 2007.

[Bae99] Baeza-Yates R. Modern Informatio Retrieval. Addison Wesley Longman Publishing Co. Inc., 1999.

[Bar99] Barzilay R., Elhadad M. Using lexical chains for text summarization. In: Inderjeet Mani, Mark T. Maybury (Eds.), Advances in Automatic Text Summarization, Cambridge/MA, London/England: MIT Press, pp. 111–121, 1999.

[Bar03] Barzilay R. Information Fusion for Multi Document Summarization. Ph.D. thesis. Columbia University, 2003.

[Bar05] Barzilay R., McKeown K. Sentence Fusion for Multi Document News Summarization. Computational Linguistics, Vol. 31, Issue 3, pp. 297–328, ISSN: 0891-2017, 2005.

[Bie06] Biemann, C.: Chinese whispers – an afficient graph clustering algorithm and its application to natural language processing. Proc. Of the HLT-NAACL, 2006.

[Bol04a] Bolshakov I., Gelbukh A. Computational Linguistics: Models, Resources, Applications. IPN-UNAM-FCE, ISBN 970-36-0147-2, 2004.

[Bol04b] Bolshakov I. Getting One's First Million... Collocations. Lecture Notes in Computer Science, Springer-Verlag, ISSN 0302-9743, vol. 2945, pp. 226–239, 2004.

[Bol05] Bolshakov I., Galicia-Haro S., Gelbukh A. Detection and Correction of Malapropisms in Spanish by means of Internet Search. 8th International Conference Text, Speech and Dialogue (TSD-2005), Czech Rep. Lecture Notes in Artificial Intelligence (indexed in SCIE), ISSN 0302-9743, ISBN 3-540-28789-2, Springer-Verlag, vol. 3658, pp. 115–122, 2005.

[Bol08] Bolshakov I. Various Criteria of Collocation Cohesion in Internet: Comparison of Resolving Power. Computational Linguistics and Intelligent Text Processing (CICLing-2008, Israel). Lecture Notes in Computer Science, Springer-Verlag, vol. 4919, pp. 64–72, 2008.

[Bri98] Brin S., Page L. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, vol. 30, pp. 1–7, 1998.

[Bru01] Brunn M., Chali Y., Pinchak C. Text Summarization Using Lexical Chains. Proc. of Document Understanding Conference 2001, http://duc.nist.gov/pubs.html#2001.

[Car03] Carlson L., Marcu D., Okurowski M. E. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In: Jan van Kuppevelt and Ronnie Smith, editors, Current Directions in Discourse and Dialogue. Kluwer Academic Publishers, 2003.

[Car04] Carberry S., Elzer S., Green N., McCoy K., Chester D. Extending Document Summarization to Information Graphics. Proc. of the ACL-04 Workshop: Text Summarization Branches Out, pp. 3–9.

[Car06] Carberry S., Elzer S., Demir S. Information Graphics: An Untapped Recourse for Digital Libraries. SIGIR'06, ACM 1-59593-369-7/06/0008, 2006.

[CLEF] http://clef-qa.itc.it

[CICLing] CICLing. Conference on Intelligent Text Processing and Computational Linguistics (2000-2009): www.CICLing.org.

[Cor04] Corston-Oliver S., Ringger E., Gamon M., Campbell R. Task-focused Summarization of emails. Proc. of the ACL-04 Workshop: Text Summarization Branches Out, pp. 43–50.

[Dau04] Daume H., Marcu D. Generic Sentence Fusion and Ill-defined Summarization Task. Proc. of ACL Workshop on Summarization, pp.96–103, 2004.

[Dav07] D'Avanzo E., Elia A., Kuflik T., Vietri S. LAKE System at DUC-2007. Proc. of Document Understanding Conference 2007. http://duc.nist.gov/pubs.html#2007.

[Dia06] Dia Q., Shan J. A New Web Page Summarization Method. SIGIR'06, ACM 1-59593-369-7/06/0008, 2006.

[Elh92] Elhadad, M. Using argumentation to control lexical choice: a unification-based implementation. PhD thesis, Computer Science Department, Columbia University, 1992.

[Eva05] Evans D., McKeown K. Identifying Similarities and Differences Across Arabic and English News. Proc. of the International Conference on Intelligence Analysis. McLean, VA, 2005.

[Far04] Farzindar A., Lapalme G. Legal Text Summarization by Exploration of the Thematic Structure and Argumentative Roles. Proc. of the ACL-04 Workshop: Text Summarization Branches Out, pp. 27–33.

[Fer07] Ferrández S., Ferrández A. The Negative Effect of Machine Translation on Coross–Lingual Question Answering. Lecture Notes in Computer Science, Springer-Verlag, vol. 4394, pp. 494–505, 2007.

[Fon08]Marco Fonseca, Leonardo Junior, Alexandre Melo, Hendrik Macedo. Innovative Approach for Engineering NLG Systems: the Content Determination Case Study. . Springer Verlag LNCS 4919, pp. 489-460, 2006.

[Fuhr92] Fuhr N. Probabilistic Models in Information Retrieval, The Computer Journal, 35(3), pp. 243-254, 1992.

[Fut04] Futrelle R. Handling Figures in Document Summarization. Proc. of the ACL-04 Workshop: Text Summarization Branches Out, pp. 61–65.

[Gar04] García-Hernández R. A., Martínez-Trinidad J. F., and Carrasco-Ochoa J. A. A Fast Algorithm to Find All the Maximal Frequent Sequences in a Text, 9th Iberoamerican Congress on Pattern Recognition (CIARP), LNCS vol. 3287, pp. 478–486, Springer-Verlag 2004.

[Gar06] García-Hernández R. A., Martínez-Trinidad J. F., and Carrasco-Ochoa J. A. A New Algorithm for Fast Discovery of Maximal Sequential Patterns in a Document Collection, LNCS 2945, pp. 514–523, Springer-Verlag 2006.

[Gar08a] René García Hernández, Yulia Ledeneva, Alexander Gelbukh, Citlalih Gutiérrez-Estrada. An Assessment of Word Sequence Models for Extractive Text Summarization. Research in Computing Science, vol.38, pp. 253-262, ISSN 1870-4069, 2008.

[Gar08b] René García Hernández, Yulia Ledeneva, Alexander Gelbukh, Erendira Rendon, Rafael Cruz. Text Summarization by Sentence Extraction Using Unsupervised methods. LNAI 5317, pp. 133-143, Mexico, Springer-Verlag, ISSN 0302-9743, 2008.

[Gar09] René García Hernández, Yulia Ledeneva. Word Sequence Models for Single Text Summarization. IEEE Computer Society Press, Cancun México, pp.44-49, ISBN 9780769535296, 2009.

[Gel00] Gelbukh, A. A data structure for prefix search under access locality requirements and its application to spelling correction. Proc. of MICAI-2000: Mexican International Conference on Artificial Intelligence, Acapulco, Mexico, 2000.

[Gel92] Gelbukh, A. An effectively implementable model of morphology of an inflective language (in Russian). J. Nauchno-Tehnicheskaya Informaciya (NTI), ser. 2, vol. 1, Moscow, Russia, 1992, pp. 24-31.

[Gel03] Alexander Gelbukh and Grigori Sidorov. Approach to construction of automatic morphological analysis systems for inflective languages with little effort. Lecture Notes in Computer Science, N 2588, 2003, ISSN 0302-9743, Springer-Verlag, pp. 215–220

[Gel03a] Gelbukh A., Sidorov G., Sang Yong Han. Evolutionary Approach to Natural Language Word Sense Disambiguation through Global Coherence Optimization. WSEAS Transactions on Communications, ISSN 1109-2742, Issue 1 Vol. 2, pp. 11–19, 2003.

[Gel03b] Gelbukh A., Bolshakov I. Internet, a true friend of translator. International Journal of Translation, ISSN 0970-9819, Vol. 15, No. 2, pp. 31–50, 2003.

[Hac04] Hachey B., Grover C. A Rhetorical Status Classifier for Legal Text Summarization. Proc. of the ACL-04 Workshop: Text Summarization Branches Out, pp. 35–42.

[Hau99] Hausser, Ronald. Three principled methods of automatic word form recognition. Proc. of VEXTAL: Venecia per il Tratamento Automatico delle Lingue. Venice, Italy, Sept. 1999. pp. 91-100.

[Hov03] Hovy E. The Oxford handbook of Computational Linguistics, Chapter about Text Summarization, In Mitkov R. (ed.), NY, 2003.

[Hu06] Haiqing Hu, Fuji Ren et. al. A Question Answering System on Special Domain and the Implementation of Speech Interface. Springer Verlag 3878, CICLing 2006.

[Kle99] Kleinberg J. Authoritative sources in a hyperlinked enviroment. Journal of the ACM, vol. 46, num. 5, pp. 604–632, 1999.

[Kos83] Koskenniemi, Kimmo. Two-level Morphology: A General Computational Model for Word-Form Recognition and Production. University of Helsinki Publications, N 11, 1983.

[Led08a] Yulia Ledeneva. Effect of Preprocessing on Extractive Summarization with Maximal Frequent Sequences. MICAI-08, LNAI 5317, pp. 123-132, Mexico, Springer-Verlag, ISSN 0302-9743, 2008.

[Led08b] Yulia Ledeneva, Alexander Gelbukh, René García Hernández. Terms Derived from Frequent Sequences for Extractive Text Summarization. CICLing-08, LNCS 4919, pp. 593-604, Israel, Springer-Verlag, ISSN 0302-9743, 2008.

[Led08c] Yulia Ledeneva, Alexander Gelbukh, René García Hernández. Keeping Maximal Frequent Sequences Facilitates Extractive Summarization. In: G. Sidorov *et al* (Eds). CORE-2008, Research in Computing Science, vol. 34, pp. 163-174, ISSN 1870-4069, 2008.

[Lin97] Lin C. Y., Hovy E. Identify Topics by Position. Proc. of the 5th Conference on Applied NLP, 1997.

[Lin03a] Lin C. Y. ROUGE: A Package for Automatic Evaluation of Summaries. Proc. of the Human Technology Conference (HLT-NAACL), Canada, 2003.

[Lin03b] Lin C. Y., Hovy E. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. Proc. of the Human Technology Conference, Canada, 2003.

[Lin04a] Lin C. Y. Looking for a Few Good Metrics: Automatic Summarization Evaluation – How Many Samples Are Enough? Proc. of NTCIR Workshop, Japan, 2004.

[Lin04b] Lin C. Y., *et al*. ORANGE: a Method for Evaluating Automatic Evaluation Metrics for Machine Translation. In: Proc. of the 20th International Conference on Computational Linguistics (COLING), Switzerland.

[Lin04c] Lin C. Y., *et al*. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics. Proc. of the 42nd Annual Meeting of the ACL, Spain, 2004.

[Liu06] Liu D., He Yanxiang, and *et al*. Multi-Document Summarization Based on BE-Vector Clustering. CICLing 2006, LNCS, vol. 3878, Springer-Verlag, pp. 470–479, 2006.

[Li07] Li J., Sun L. A Query-Focused Multi-Document Summarizer Based on Lexical Chains. Proc. of Document Understanding Conference 2007. http://duc.nist.gov/pubs.html#2007.

[Luh57] Luhn H.P. A statistical Approach to Mechanical Encoding and Searching of literary information. IBM Journal of Research and Development, pp. 309–317, 1957.

[Mad07] Madnani N., Zajic D., Dorr B., etc. Multiple Alternative Sentence Compressions for Automatic Text Summarization. Document Understanding Conference 2007. http://duc.nist.gov/pubs.html#2007

[Mal85] Malkovsky, M. G. *Dialogue with the artificial intelligence system* (in Russian). Moscow State University, Moscow, Russia, 1985, 213 pp.

[Man99] Manning C. Foundations of Statistical Natural Language Processing, MIT Press, London, 1999.

[Man07] Manning C. An Introduction to Information Retrieval. Cambridge University Press, 2007.

[Mar01] Marcu D. Discourse-based summarization in DUC-2001. Document Understanding Conference 2001. http://duc.nist.gov/pubs.html#2001

[Mck03] McKeown K., Barzilay R., Chen J., Elson D., Klavans J., Nenkova A., Schiffman B., Sigelman S. Columbia's Newsblaster: New Features and Future Directions. Proc. of the Human Language Technology Conference, vol. II, 2003.

[Mih04] Mihalcea, R., Tarau, P. TextRank: Bringing Order into Texts. Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), Spain, 2004.

[Mih06] Mihalcea R. Ramdom Walks on Text Structures. CICLing 2006, LNCS, vol. 3878, pp. 249–262, Springer-Verlag 2006.

[Mor91] Morris J., Hirst G. Lexical cohesion computed by thesaurus relations as an indicator of the structure of text. Computational Linguistics, vol. 18, pp. 21–45, 1991.

[Nen04] Nenkova A., Passonneau R. Evaluating content selection in summarization: The pyramid method. In: Proc. of NLT/NAACL–2004, 2004.

[Nen05a] Nenkova A., Siddharthan A., McKeown K. Automatically Learning Cognitive Status for Multi-Document Summarization of Newswire. Proc. of HLT/EMNLP-05, 2005.

[Nen06] Nenkova A. Understanding the process of multi-document summarization: content selection, rewriting and evaluation. Ph.D. Thesis, Columbia University, 2006.

[Ngo08] Axel-Cyrille Ngonga Ngomo. SIGNUM: A graph algorithm for terminology extraction. Springer Verlag, vol. 4919, pp. 85-95, 2008.

[Ngo09] Axel-Cyrille Ngonga Ngomo and Frank Schumacher. Border Flow: A Local Graph Clustering Algorithm for Natural Language Processing. Springer Verlag, vol. 5449, pp. 547-558, 2009.

[Omg01] Object Management Group. Model Driven Architecture (MDA). OMG Document ormsc/2001.

[Ott06] Otterbacher J., Radev D., Kareem O. News to Go: Hierarchical Text Summarization for Mobile Devices. SIGIR'06, ACM 1-59593-369-7/06/0008.

[Pas05] Passonneau R., Nenkova A., McKeown K., Sigleman S. Pyramid evaluation at DUC-05. Proc. of Document Understanding Conference, http://duc.nist.gov/pubs.html#2007.

[Peñ06] Anselmo Peñas, Álvaro Rodrigo, Felisa Verdejo: Overview of the Answer Validation Exercise 2006, CLEF 2006.

[Peñ07] Álvaro Rodrigo, Anselmo Peñas, Felisa Verdejo: Overview of the Answer Validation Exercise 2007. CLEF 2007: 237-248.

[Peñ08] Anselmo Peñas, Álvaro Rodrigo, Felisa Verdejo: Overview of the Answer Validation Exercise 2007, CLEF 2008.

[Pin06] David Pinto, Héctor Jiménez-Salazar, and Paolo Rosso. Clustering Abstracts of Scientific Texts using the Transition Point Technique. Springer Verlag, Cicling 2006.

[Raz07] Razmara Marat, Kossein Leila. A Little Known Fact is … Answering Other Questions Using Interest-Markers. Springer Verlag 4394, CICLing 2007.

[Sal88] Salton G., Buckley C. Term-Weighting Approaches in Automatic Text Retreival, Information processing and Management, vol. 24, pp. 513–523, 1988.

[Sar05] Sarkar K., *et al*. Generating Headline Summary from a Document Set. CICLing, LNCS, vol. 3406, Springer-Verlag, pp. 637–640, 2005.

[Sch98] Schütze, H.: Automatic Word sense disambiguation. Computational Linguistics 24(1), pp. 97-123, 1998.

[Sed01] Sedlacek R. and P. Smrz, A new Czech morphological analyzer AJKA. Proc. of TSD-2001. LNCS 2166, Springer, 2001, pp 100-107.

[Sid96] Sidorov, G. O. Lemmatization in automatized system for compilation of personal style dictionaries of literature writers (in Russian). In: Word by Dostoyevsky (in Russian), Moscow, Russia, Russian Academy of Sciences, 1996, pp. 266-300.

[Sil02] Silber H., McCoy K. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. Computational Linguistics, 28(4), pp. 487–496, 2002.

[Sor03] Soricut R., Marcu D. Sentence level discourse parsing using sintactic and lexical information. In: HLT-NAACL, 2003.

[Teu04a] Teusel S., Halteren H. van. Agreement in human factoid annotation for summarization evaluation. Proc. of the 4th Conference on Language Resources and Evaluation (LREC), 2004.

[Teu04b] Teusel S., Halteren H. van. Evaluating Information content by factoid analysis: human annotation and stability. EMNLP, pp. 419–426, 2004.

[Tel08] Tellez-Valero A., Montes-y-Gomez M., Villaseñor-Pineda L., Peñas A. Improving Question Answering by Combining Multiple Systems via Answer Validation. Springer Verlag, CICLing 2008.

[TREC] http://trec.nist.gov

[Van04] Vandeghinste V., Pan Y. Sentence Compression for Automated Subtitling: A Hybrid Approach. Proc. of ACL Workshop on Summarization, pp. 89–95, 2004.

[Van08] Thanh-Trung Van, Michel Beigbeder. Hybrid Method for Personalized Search in Scientific Digital Libraries. CICLing 2008.

[Vil06] Villatoro-Tello E., Villaseñor- Pineda L., and Montes-y-Gómez M. Using Word Sequences for Text Summarization. 9th International Conference on Text, Speech and Dialogue (TSD). Lecture Notes in Artificial Intelligence, Springer-Verlag, 2006.

[Voo04a] Voorhees Ellen M. Overview of the TREC-2003 question answering task. In: Proc. of the 12th Text Retrieval Conference (TREC-2003), pp. 54–68, 2004.

[Voo04b] Voorhees E. M. Overview of the TREC 2004 Question Answering Track.

[Voo04c] Voorhees E. M., Buckland L.P. (Eds.) Proceedings of the 13-th TREC, National Institute of Standards and Technology (NIST), 2004.

[Voo05a] Voorhees E. M., Dang H.T. Overview of the TREC 2005 Question Answering Track.

[Voo05b] Voorhees E. M., Buckland L.P. (Eds.) Proceedings of the 14-th TREC, National Institute of Standards and Technology (NIST), 2004.

[Wan04] Wan S., McKeown K. Generating "State-of Affairs" Summaries of Ongoing Email Thread Discussions. Proc. of Coling 2004, Switzerland, 2004.

[Wan06] Wan X., Yang J., *et al*. Incorporating Cross-Document Relationships Between Sentences for Single Document Summarizations. ECDL, LNCS, vol. 4172, Springer-Verlag, pp. 403–414, 2006.

[Yin07] Ying J.-C., Yen S.-J., Lee Y.-S. Language Model Passage Retrieval for Question-Oriented Multi Document Summarization. Proc. of Document Understanding Conference 2007. http://duc.nist.gov/pubs.html#2007.

[Zho05] Zhou Q., Sun L. IS_SUM: A Multi-Document Summarizer based on Document Graphics and Lexical Chains. Proc. of Document Understanding Conference 2005. http://duc.nist.gov/pubs.html#2005.

Paraphrase Identification Using Weighted Dependencies and Word Semantics

Mihai C. Lintean and Vasile Rus
Department of Computer Science
The University of Memphis
Memphis, TN, 38120, USA
E-mail: {mclinten,vrus}@memphis.edu

Keywords: paraphrasing, syntactic dependencies, word semantics

Received: March 24, 2009

We present in this article a novel approach to the task of paraphrase identification. The proposed approach quantifies both the similarity and dissimilarity between two sentences. The similarity and dissimilarity is assessed based on lexico-semantic information, i.e., word semantics, and syntactic information in the form of dependencies, which are explicit syntactic relations between words in a sentence. Word semantics requires mapping words onto concepts in a taxonomy and then using word-to-word similarity metrics to compute their semantic relatedness. Dependencies are obtained using state-of-the-art dependency parsers. One important aspect of our approach is the weighting of missing dependencies, i.e., dependencies present in one sentence but not the other. We report experimental results on the Microsoft Paraphrase Corpus, a standard data set for evaluating approaches to paraphrase identification. The experiments showed that the proposed approach offers state-of-the-art results. In particular, our approach offers better precision when compared to other approaches.

Povzetek: Prispevek se ukvarja z vsebinsko primerjavo dveh stavkov, tj. parafrazami.

1 Introduction

We present in this paper a novel approach to the task of paraphrase identification. Paraphrase is a text-to-text relation between two non-identical text fragments that express the same idea in different ways. As an example of a paraphrase we show below a pair of sentences from the Microsoft Research (MSR) Paraphrase Corpus [5] in which Text A is a paraphrase of Text B and vice versa.

Text A: *York had no problem with MTA's insisting the decision to shift funds had been within its legal rights.*

Text B: *York had no problem with MTA's saying the decision to shift funds was within its powers.*

Paraphrase identification is the task of deciding whether two given text fragments have the same meaning. We focus in this article on identifying paraphrase relations between sentences such as the ones shown above. It should be noted that paraphrase identification is different from paraphrase extraction. Paraphrase extraction [1, 2] is the task of extracting fragments of texts that are in a paraphrase relation from various sources. Paraphrase could be extracted, for instance, from texts that contain redundant semantic content such as news articles from different media sources that cover the same topic, or multiple English translations, by different translators, of same source texts in a foreign language. Recognizing textual entailment [4, 20] is another task related to paraphrase identification. Entailment is a text-to-text relation between two texts in which one text entails, or logically infers, the other. Entailment defines an asymmetric relation between two texts, meaning that one text is entailed by the other text, while paraphrase requires a symmetric relation between the two texts, i.e. one text can be entailed from the other and viceversa. Rus and colleagues [20] showed that approaches to textual entailment can be extended to handle paraphrase identification.

In this paper, we focus on the problem of paraphrase identification. Paraphrase identification is an important task in a number of applications including Question Answering [9], Natural Language Generation [10], and Intelligent Tutoring Systems [6, 15]. In Natural Language Generation, paraphrases are a method to increase diversity of generated text [10]. In Question Answering, multiple answers that are paraphrases of each other could be considered as evidence for the correctness of the answer [9]. For Intelligent Tutoring Systems with natural language input [6, 15] paraphrases are useful to assess whether student's articulated answers to deep questions (e.g. conceptual physics questions) are similar-to/paraphrases-of ideal answers.

We propose in this article a fully automated approach to the task of paraphrase identification. The basic idea is that two sentences are in a paraphrase relation if they have many similarities (at lexico-semantic and syntactic levels) and few or no dissimilarities. For instance, the two sentences shown earlier from the MSR paraphrase corpus have many similarities, e.g., common words such as *York* and common syntactic relations such as the *subject* relationship between *York* and *have*, and only a few dissimilarities, e.g., Text A contains the word *saying* while Text B contains the word *insisting*. Thus, we can confidently deem the two sentences as being paraphrases of each other. Following this basic idea, to identify paraphrases we first compute two scores: one reflecting the similarity and the other the dissimilarity between the two sentences. A paraphrase score is generated by taking the ratio of the similarity and dissimilarity scores. If the ratio is above a certain threshold, the two sentences are judged as being paraphrases of each other. The threshold is obtained by optimizing the performance of the proposed approach on training data.

There are several key features of our approach that distinguish it from other approaches to paraphrase identification. *First*, it considers both similarities and dissimilarities between sentences. This is an advantage over approaches that only consider the degree of similarity [19] because the dissimilarity of two sentences can be very important to identifying paraphrasing, as shown by [18] and later in this article. *Second*, the similarity between sentences is computed using word-to-word similarity metrics instead of simple word matching or synonymy information in a thesaurus as in [19, 18]. The word-to-word similarity metrics can identify semantically related words even if the words are not identical or synonyms. We use the similarity metrics from the WordNet similarity package [17]. These metrics rely on statistical information derived from corpora and lexico-semantic information from WordNet [16], a lexical database of English. The basic idea behind the WordNet similarity metrics is that the closer the distance in WordNet between words/concepts is, the more similar they are. For instance, in the earlier example the semantic relationship between the words *insist* and *say* cannot be established using simple direct matching or synonymy. On the other hand, there is a relatively short path of three nodes in WordNet from *say* to *insist* via *assert*, indicating *say* and *insist* are semantically close. *Third*, we weight dependencies to compute dissimilarities between sentences as opposed to simple dependency overlap methods that do no weighting (see [13, 20]). The weighting allows us to make fine distinctions between sentences with a high similarity score that are paraphrases and those that are not due to the strength of the few dissimilarities. For instance, two sentences that are almost identical except their subject relations are likely to be non-paraphrases as opposed to two highly similar sentences that differ in terms of, say, determiner relations. We weight dependencies using two features: (1) the type/label of the dependency, and (2) the depth of a dependency in the dependency tree. To extract dependency information we used two parsers, Minipar [11] and the Stanford parser [14]. We report results with each of the parsers.

We used the MSR Paraphase Corpus [5], an industry standard for paraphrase identification, to evaluate our approach. The corpus is divided into two subsets: training and test data. The training subset was used to obtain the optimal threshold above which a similarity/dissimilarity ratio would indicate a paraphrase or a non-paraphrase, otherwise. We report state-of-the-art results on the testing data (72.06% accuracy, with Minipar), which are significantly better (Fisher's exact test yields a $p = 0.00005$) than the baseline approach of always predicting the most frequent class in the training data (66.49% accuracy) and than a simple dependency overlap method ($p<0.001$; with Minipar). Compared to results obtained using the Stanford parser (71.01% accuracy), Minipar led to statistically significant better results ($p = 0.004$).

Following this introductory part, in the next section, *What is a paraphrase?*, we offer a broader view of the concept of paraphrase. The article continues with a section on *Related Work*. The *Approach* section describes in detail how our similarity-dissimilarity method works. The following *Summary of Results* section provides details of the experimental setup, results, and a comparison with results obtained by other research groups. The *Discussion* section offers further insights into our approach and the MSR Paraphrase Corpus. The *Summary and Conclusions* section ends the article.

2 What is a paraphrase?

A quick search with the query *What is a paraphrase?* on a major search engine reveals many definitions for the concept of paraphrase. Table 1 presents a small sample of such definitions. From the table, we notice that the most common feature in all these definitions is *different/own words*. That is, a sentence is a paraphrase of another sentence if it conveys the same meaning using different words. While these definitions seem to be quite clear, one particular type of paraphrases, sentence-level paraphrases (among texts the size of a sentence), do not seem to follow the above definitions as evidenced by existing data sets of such paraphrases.

For sentential paraphrases, the feature of "different words" seems to be too restrictive, although not impossible. As we will show later in the article, the MSR Paraphrase corpus supports this claim as the paraphrases in the corpus tend to have many words in common as opposed to using *different words* to express the same meaning. While the high lexical overlap of the paraphrases in the MSR corpus can be explained by the protocol used to create the corpus - same keywords were used to retrieve same stories from different sources on the web, in general, we could argue that avoiding the high word overlap issue in sentential paraphrasing would be hard. Given an isolated sentence it would be quite challenging to omit/replace some core concepts when trying to paraphrase. Here is an example of a sentence (instance 735 in MSR corpus), *Counties with population declines will be Vermillion, Posey and Madison.*, which would be hard to paraphrase using many other/different words. The difficulty is due to the large number of named entities in the sentence. Actually, its paraphrase in the corpus is *Vermillion, Posey and Madison County populations will decline.*, which retains all the named entities from the original corpus as it is close to impossible to replace them with other words. It is beyond the

Table 1: Definitions of paraphrases from various sources.

Source	Definition. A paraphrase (is)...
Wikipedia	a restatement of a text or passage *using different words*.
Wordnet	express the same message in *different words*; rewording for the purpose of clarification.
Purdue's OWL	*your own rendition* of essential information and ideas expressed by someone else, presented in a new form.
Bedford/St.Martin's	a prose restatement of the central ideas of a poem, in *your own language*.
Pearson's Glossary	to record someone else's words in the *writer's own words*.
LupinWorks	restating the meaning *in own words*, retaining all of the ideas without making an interpretation or evaluation.

scope of this article to provide a final answer with respect to whether high lexical overlap should be acceptable or not in sentential paraphrases.

Another interesting aspect of sentential paraphasing is the fact that there seem to be two different ways to judge them. On one hand, two sentences are considered paraphrases of each other if and only if they are *semantically equivalent*, i.e. they both convey the same message with no additional information present in one sentence but not the other. An example of two sentences in a semantic equivalence was given in the previous section. Thus, in order to detect whether two sentences are *not* paraphrases of each other, we only need to find one concept that is present in one sentence but not in the other. On the other hand, two sentences can be judged as forming a paraphrase if they convey *roughly* the same message (minor details being different is acceptable). In this case, the paraphrase relation can be looked at as a bidirectional entailment relation [19]. To exemplify such loose paraphrases, we show below a pair of sentences that has been tagged as paraphrase in the MSR Paraphrase Corpus:

Text A: Ricky Clemons' brief, troubled Missouri basketball career is over.

Text B: Missouri kicked Ricky Clemons off its team, ending his troubled career there.

In this example, the first sentence specifies that the career of Mr. Clemons was brief, while the second sentence specifies the reason why Mr. Clemons' career is over. The MSR Paraphrase corpus, our experimental data set, contains both types of sentential paraphrases, i.e. precise and loose paraphrases. This characteristic of the MSR corpus impacts the performance of general approaches, such as ours, to paraphrase identification that are not biased towards judging styles. A general approach to paraphrase identification assumes that two sentences are paraphrases of each other if they have exactly the same meaning.

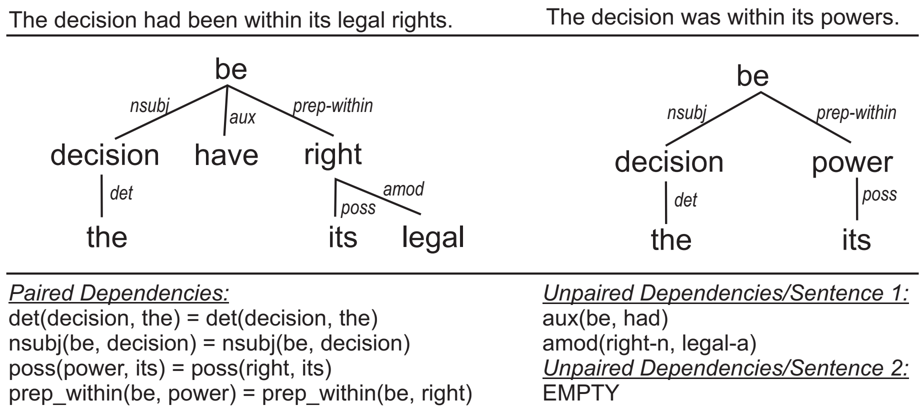
3 Related work

Paraphrase identification has been explored in the past by many researchers, especially after the release of the MSR Paraphrase Corpus [5]. We describe in this section four previous studies that are most related to our approach and leave others out, e.g., [8, 21] due to space reasons.

Rus and colleagues [19] addressed the task of paraphrase identification by computing the degree of subsumption at lexical and syntactic level between two sentences in a bidirectional manner: from Text A to Text B and from Text B to Text A. The approach relied on a unidirectional approach that was initially developed to recognize the sentence-to-sentence relation of entailment [20]. Rus and colleagues' approach only used similarity to decide paraphrasing, ignoring dissimilarities which could be important to the final decision. The similarity was computed as a weighted sum of lexical matching, i.e. direct matching of words enhanced with synonymy information from WordNet, and syntactic matching, i.e., dependency overlap. Dependencies were derived from a phrase-based parser which outputs the major phrases in a sentence and organizes them hierarchically into a parse tree. Our approach has a better lexical component based on word semantics and a finer syntactic analysis component based on weighted dependencies. Furthermore, the use of phrase-based parsing in [19] limits the applicability of the approach to free-order languages for which dependency parsing is more suitable.

Corley and Mihalcea [3] proposed an algorithm that extends word-to-word similarity metrics into a text-to-text semantic similarity metric based on which they decide whether two sentences are paraphrases or not. To obtain the semantic similarity between individual words, they used the same WordNet similarity package as we do. Our approach has the advantage that it considers syntactic information, in addition to word semantics, to identify paraphrases.

Qiu and colleagues [18] proposed a two-phase architecture for paraphrase identification. In the first phase, they identified similarities between two sentences, while in the second phase the dissimilarities were classified with respect to their relevance in deciding the presence of paraphrase. Their approach uses predicate argument tuples that capture both lexical and syntactic dependencies among words to find similarities between sentences. The first



Paired Dependencies:
det(decision, the) = det(decision, the)
nsubj(be, decision) = nsubj(be, decision)
poss(power, its) = poss(right, its)
prep_within(be, power) = prep_within(be, right)

Unpaired Dependencies/Sentence 1:
aux(be, had)
amod(right-n, legal-a)
Unpaired Dependencies/Sentence 2:
EMPTY

Figure 1: Example of dependency trees and sets of paired and non-paired dependencies.

phase is similar to our approach for detecting common dependencies. In the second phase, they used a supervised classifier to detect whether the dissimilarities are important. There are two advantages of our approach compared to Qiu and colleagues' approach (1) we use word semantics to compute similarities, (2) we take advantage of the dependency types and position in the dependency tree to weight dependencies as opposed to simply using non-weighted/unlabeled predicate-argument relations.

Zhang and Patrick [22] offer another ingenious solution to identify sentence-level paraphrase pairs by transforming source sentences into canonicalized text forms at the lexical and syntactic level, i.e. generic and simpler forms than the original text. One of the surprising findings is that a baseline system based on a supervised decision tree classifier with simple lexical matching features leads to best results compared to more sophisticated approaches that were experimented by them or others. They also revealed limitations of the MSR Paraphrase Corpus. The fact that their text canonicalization features did not lead to better than the baseline approach supports their findings that the sentential paraphrases, at least in the MSR corpus, share more words in common than one might expect given the standard definition of a paraphrase. The standard definition implies to use different words when paraphrasing. Zhang and Patrick used decision trees to classify the sentence pairs making their approach a supervised one as opposed to our approach which is minimally supervised - we only need to derive the value of the threshold from training data for which it is only necessary to know the distribution of true-false paraphrases in the training corpus and not the individual judgment for every instance in the corpus. They rely only on lexical and syntactic features while we also use semantic similarity factors.

We will compare the results of our approach on the MSR corpus with these related approaches. But first, we must detail the innerworkings of our approach.

4 Approach

As mentioned earlier, our approach is based on the observation that two sentences express the same meaning, i.e., are paraphrases, if they have all or many words and syntactic relations in common. Furthermore, the two sentences should have few or no dissimilar words or syntactic relations. In the example below, we show two sentences with high lexical and syntactic overlap. The different information, *legal rights* in the first sentence and *powers* in the second sentence, does not have a significant impact on the overall decision that the two sentences are paraphrases, which can be drawn based on the high degree of lexical and syntactic overlap.

Text A: *The decision was within its legal rights.*

Text B: *The decision was within its powers.*

On the other hand, there are sentences that are almost identical, lexically and syntactically, and yet they are not paraphrases because the few dissimilarities make a big difference. In the example below, there is a relatively "small" difference between the two sentences. Only the subject of the sentences is different. However, due to the importance of the subject relation to the meaning of any sentence the high similarity between the sentences is sufficiently dominated by the "small" dissimilarity to make the two sentences non-paraphrases.

Text A: *CBS is the leader in the 18 to 46 age group.*

Text B: *NBC is the leader in the 18 to 46 age group.*

Thus, it is important to assess both similarities and dissimilarities between two sentences S_1 and S_2 before making a decision with respect to them being paraphrases or not. In our approach, we capture the two aspects, similarity or dissimilarity, and then find the dominant aspect by computing a final paraphrase score as the ratio of the similarity and dissimilarity scores: Paraphrase(S_1, S_2)=Sim(S_1, S_2)/Diss(S_1, S_2). If the paraphrase score is above a learned threshold T the sentences are deemed paraphrases. Otherwise, they are non-paraphrases.

The similarity and dissimilarity scores are computed based on dependency relations [7], which are asymmetric relationships between two words in a sentence, a *head* or modifee, and a *modifier*. A sentence can be represented by a set of dependency relations (see the bottom half of Figure 1). An example of dependency is the *subject* relation between *John* and *drives* in the sentence *John drives a car.* Such a dependency can be viewed as the triple *subj(John, drive)*. In the triplets the words are lemmatized, i.e., all morphological variations of a word are mapped onto its base form. For instance, *go, went, gone, going* are all mapped onto *go*.

The Sim(S_1, S_2) and Diss(S_1,S_2) scores are computed in three phases: (1) map the input sentences into sets of dependencies, (2) detect common and non-common dependencies between the sentences, and (3) compute the Sim(S_1, S_2) and Diss(S_1,S_2) scores. Figure 2 depicts the general architecture of the system in which the three processing phases are shown as the three major modules.

In the first phase, the set of dependencies for the two sentences is extracted using a dependency parser. We use both Minipar [11] and the Stanford parser [14] to parse the sentences. Because these parsers do not produce perfect output the reader should regard our results as a lower bound, i.e. results in the presence of parsing errors. Should the parsing been perfect, we expect our results to look better. The parser takes as input the raw sentence and returns as output a dependency tree (Minipar) or a list of dependencies (Stanford). In a dependency tree, every word in the sentence is a modifier of exactly one word, its head, except the head word of the sentence, which does not have a head. The head word of the sentence is the root node in the dependency tree. Given a dependency tree, the list of dependencies can be easily derived by traversing the tree and for each internal node, which is head of at least one dependency, we retrieve triplets of the form *rel(head, modifier)* where *rel* represents the type of dependency that links the node, i.e., the *head*, to one of its children, the *modifier*. Figure 1 shows the set of dependencies in the form of triplets for the dependency trees in the top half of the figure.

In this phase, we also gather positional information about each dependency in the dependency tree as we will need this information later when weighting dependencies in Phase 3. The position/depth of a dependency within the dependency tree is calculated as the distance from the root of the node corresponding to the head word of the dependency. Because the Stanford parser does not provide the position of the dependencies within the tree, we had to recursively reconstruct the tree based on the given set of dependency relations and calculate the relative position of each relation from the root.

The second phase in our approach identifies the common and non-common dependencies of the sentences, based on word semantics and syntactic information. Three sets of dependencies are generated in this phase: one set of *paired*/common dependencies and two sets of *unpaired* dependencies, one corresponding to each of the two sentences. To generate the paired and unpaired sets a two-step procedure is used. In the first step, we take one dependency from the shorter sentence in terms of number of dependencies (a computational efficiency trick) and identify dependencies of the same type in the other sentence. In the second step, we compute a dependency similarity score (d2dSim) using the word-to-word similarity metrics applied to the two heads and two modifiers of the matched dependencies. Heads and modifiers are mapped onto all the corresponding concepts in WordNet, one concept for each sense of the heads and modifiers. The similarity is computed among all senses/concepts of the two heads and modifiers, respectively, and then the maximum similarity is retained. If a word is not present in WordNet exact matching is used. The word-to-word similarity scores are combined into one final dependency-to-dependency similarity score by taking the weighted average of the similarities of the heads and modifiers. Intuitively, more weight should be given to the similarity score of heads and less to the similarity score of modifiers because heads are the more important words. Surprisingly, while trying to learn a good weighting scheme from the training data we found that the opposite should be applied: more weight should be given to modifiers (0.55) and less to heads (0.45). We believe this is true only for the MSR Paraphrase Corpus and this weighting scheme should not be generalized to other paraphrase corpora. The MSR corpus was built in such a way that favored highly similar sentences in terms of major content words (common or proper nouns) because the extraction of the sentences was based on keyword searching of major events from the web. With the major content words similar, the modifiers are the heavy lifters when it comes to distinguishing between paraphrase and non-paraphrase cases. Another possible approach to calculate the similarity score between dependencies is to rely only on the similarity of the most disimilar items, either heads or modifiers. We also tried this alternative approach, but it gave slightly poorer results (around 2% decrease in performance), and therefore, using a weighted scheme to calculate the similarity score for dependencies proved to be a better choice. The dependency-to-dependency similarity score needs to exceed a certain threshold for two matched dependencies to be deemed similar. Empirically, we found out from training data that a good value for this threshold would be *0.5*. Once a pair of dependencies is deemed similar, we place it into the paired dependencies set, along with the calculated dependency-to-dependency similarity value. All the dependencies that could not be paired are moved into the unpaired dependencies sets.

$$sim(S_1, S_2) = \sum_{d_1 \in S_1} max_{d_2 \in S_2^*}[d2dSim(d_1, d_2)]$$

$$diss(S_1, S_2) = \sum_{d \in \{unpairedS_1, unpairedS_2\}} weight(d)$$

In the third and final phase of our approach, two scores are calculated from the three dependency sets obtained

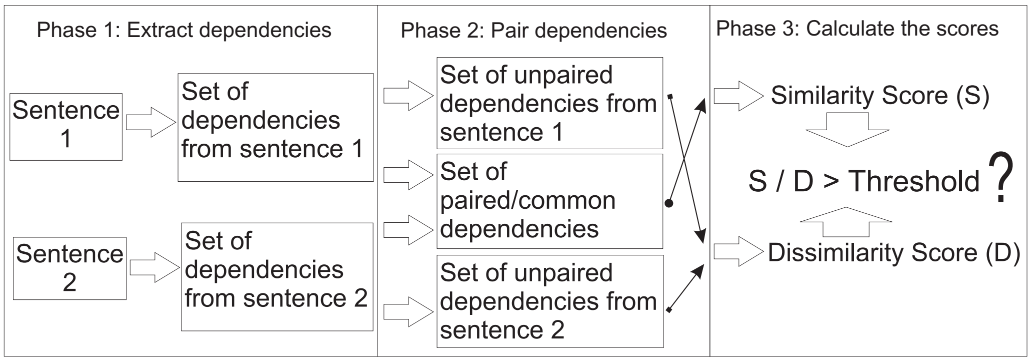

Phase 1: Extract dependencies
Sentence 1
Set of dependencies from sentence 1
Sentence 2
Set of dependencies from sentence 2
Phase 2: Pair dependencies
Set of unpaired dependencies from sentence 1
Set of paired/common dependencies
Set of unpaired dependencies from sentence 2
Phase 3: Calculate the scores
Similarity Score (S)
S / D > Threshold ?
Dissimilarity Score (D)

Figure 2: Architecture of the system.

Table 2: Performance and comparison of different approaches on the MS Paraphrase Corpus.

System	Accuracy	Precision	Recall	F-measure
Uniform baseline	0.6649	0.6649	1.0000	0.7987
Random baseline [3]	0.5130	0.6830	0.5000	0.5780
Lexical baseline (from Zhang et. al.)[22]	0.7230	0.7880	0.7980	0.7930
Corley and Mihalcea [3]	0.7150	0.7230	0.9250	0.8120
Qiu [18]	0.7200	0.7250	0.9340	0.8160
Rus - average [19]	0.7061	0.7207	0.9111	0.8048
Simple dependency overlap (Minipar) [13]	0.6939	0.7109	0.9093	0.7979
Simple dependency overlap (Stanford) [13]	0.6823	0.7064	0.8936	0.7890
Optimum results (Minipar)	0.7206	0.7404	0.8928	0.8095
Optimum results (Stanford)	0.7101	0.7270	0.9032	0.8056
No word semantics (Minipar)	0.7038	0.7184	0.9119	0.8037
No word semantics (Stanford)	0.7032	0.7237	0.8954	0.8005
No dependency weighting (Minipar)	0.7177	0.7378	0.8928	0.8079
No dependency weighting (Stanford)	0.7067	0.7265	0.8963	0.8025
No penalty for extra info (Minipar)	0.7067	0.7275	0.8936	0.8020
No penalty for extra info (Stanford)	0.7032	0.7138	0.9241	0.8055

in Phase 2: a cumulative *similarity score* and a cumulative *dissimilarity score*. The cumulative similarity score Sim(S_1, S_2) is computed from the set of paired dependencies by summing up the dependency-to-dependency similarity scores (S_2^* in the equation for similarity score represents the set of remaining unpaired dependencies in the second sentence). Similarly, the dissimilarity score Diss(S_1, S_2) is calculated from the two sets of unpaired dependencies. Each unpaired dependency is weighted based on two features: the depth of the dependency within the dependency tree and type of dependency. The depth is important because an unpaired dependency that is closer to the root of the dependency tree, e.g., the main verb/predicate of sentence, is more important to indicate a big difference between two sentences. In our approach, each unpaired dependency is initially given a perfect weight of 1.00, which is then gradually penalized with a constant value (*0.20* for the Minipar output and *0.18* for the Stanford output), the farther away it is from the root node. The penalty values were derived empirically from training data. Our tests show that this particular feature works well only when applied to the sets of unpaired dependencies. The second feature that we use to weight dependencies is the type of dependency. For example a *subj* dependency, which is the relation between the verb and its subject, is more important to decide paraphrasing than a *det* dependency, which is the relation between a noun and its determiner. Each dependency type is assigned an importance level between 0 (no importance) and 1 (maximum importance). The importance level for each dependency type has been established by the authors based on their linguistic knowledge and an analysis of the role of various dependency types in a subset of sentences from the training data.

Before comparing the similarity and dissimilarity scores, we consider one more feature that will affect the disimilarity score. This improvement, of a more statistical nature, is based on the idea that if one sentence contains a significant amount of extra information compared to the other sentence although they do refer to the same action or event, then the relation between the two sentences is not a bidirectional relation of paraphrase, but rather a unidirectional relation of entailnment, so they should be evaluated as non-paraphrases. This extra information is recorded in our dependency sets by the fact that the set of unpaired dependencies from the longer, more detailed sentence is larger than the set of unpaired dependencies from the shorter sentence. To account for this statistical feature, we add an absolute value to the dissimilarity score, which was empirically chosen to be *14*, for every case when the set of unpaired dependencies from the longer sentence has more than *6* extra dependencies compared to the set of unpaired dependencies from the shorter sentence. We chose these optimal constants values to tweak this feature, based on a series of tests made on the MSR Paraphrase Corpus, and because of that, by including it into the system, the performance was improved significantly.

Once the Sim(S_1, S_2) and Diss(S_1, S_2) scores are available, the paraphrase score is calculated by taking the ratio between the similarity score, S, and the disimilarity score, D, and compare it to the optimum threshold T learned from training data. Formally, if $S/D > T$ then the instance is classified as paraphrase, otherwise is a non-paraphrase. To avoid division by zero for cases in which the two sentences are identical ($D = 0$) the actual implementation tests for $S > T * D$. To find the optimum threshold, we did an exhaustive search on the training data set, looking for the value which led to optimum accuracy. This is similar to the sigmoid function of the simple voted perceptron learning algorithm used in [3].

5 Summary of results

We experimented with our approach on the MSR Paraphrase Corpus [5]. The MSR Paraphrase Corpus is the largest publicly available annotated paraphrase corpus which has been used in most of the recent studies that addressed the problem of paraphrase identification. The corpus consists of 5801 sentence pairs collected from newswire articles, 3900 of which were labeled as paraphrases by human annotators. The whole set is divided into a training subset (4076 sentences of which 2753 are true paraphrases) which we have used to determine the optimum threshold T, and a test subset (1725 pairs of which 1147 are true paraphrases) that is used to report the performance results. We report results using four performance metrics: accuracy (percentage of instances correctly predicted out of all instances), precision (percentage of predicted paraphrases that are indeed paraphrases), recall (percentage of true paraphrases that were predicted as such), and f-measure (harmonic mean of precision and recall).

In Table 1 three baselines are reported: a uniform baseline in which the majority class (paraphrase) in the training data is always chosen, a random baseline taken from [3], and a lexical baseline taken from [22] which uses a supervised learning decision tree classifier with various lexical-matching features. We next show the results of others including results obtained using the simple dependency overlap method in [13]. The simple dependency overlap method computes the number of common dependency relations between the two sentences divided by the average number of relations in the two sentences. Our results are then presented in the following order: our best/state-of-the-art system, that uses all three features described in the previous section: word semantics, weighted dependencies and penalties for extra information, then a version of the proposed approach without word semantics (similarity in this case is 1 if words are identical, case insensitive, or 0 otherwise), then one without weighted dependencies, and finaly, one version where the instances with extra information found in one of their sentences are not penalized. The conclusion based on our best approach is that a mix of word semantics and weighted dependencies leads to better accuracy and in particular better precision. The best approach leads to significantly better results than the naive baselines and the simple dependency overlap ($p<0.001$ for the version with Minipar). The comparison between our best results and the results reported by [3] and [13] is of particular importance. These comparisons indicate that weighted dependencies and word semantics leads to better accuracy and precision than using only word semantics [3] or only simple dependency overlap [13].

All results in Table 1 were obtained with the *lin* measure from the WordNet similarity package, except the case that did not use WordNet similarity measures at all – the *No word semantics* row. This *lin* measure consistently led to the best performance in our experiments when compared to all the other measures offered by the WordNet similarity package.

For reference, we report in Table 3 results obtained when various word-to-word similarity metrics are used with an optimum threshold calculated from the *test data set*. For *lin* measure we report results with optimum test thresholds when using both parsers, Minipar and Stanford, while for the rest of the measures we only report results when using Minipar. We deem these results as one type of benchmark results for approaches that rely on WordNet similarity measures and dependencies as they were obtained by optimizing the approach on the testing data. As we can see from the table, the results are not much higher than the results in Table 1 where the threshold was derived from training data.

One important advantage that our system has over other approaches ([18], [22]) is that it does not rely too much on the training. The training data is used merely to tune the parameters, rather than for training a whole classifier. Since the only parameter whose value fully depends on the training data is the final threshold value, we've made another set of experiments where the threshold value depends only on one piece of information about the caracteristic of the test data set: the percentage of paraphrase instances within the data set. In other words, when calculating the threshold value, the system needs to know only what is the

Table 3: Accuracy results for different WordNet metrics with optimum test threshold values

Metric	Acc.	Prec.	Rec.	F
Lin$_{Minipar}$	.7241	.7395	.9032	.8132
Lin$_{Stanford}$	.7130	.7387	.8797	.8030
Path	.7183	.7332	.9058	.8105
L & C	.7165	.7253	.9233	.8124
W & P	.7188	.7270	.9241	.8138
J & C	.7217	.7425	.8901	.8097
Lesk	.7148	.7446	.8692	.8021
Vector	.7200	.7330	.9093	.8117
Vector pairs	.7188	.7519	.8614	.8029

probability of finding a paraphrase within the given data set. The system then tries to find a threshold value that splits the instances into two sets with the same distribution of instances as the given data set. For the testing part of the MSR Paraphrase data corpus the distribution value is 0.6649. We used this information to decide on a threshold and the results were no more than 2.09 percent below the optimum performance scores (for example on Minipar output and when excluding the WordNet similarity feature, the accuracy performance was only 0.06 percent less than when the threshold is calculated from the training data).

6 Discussion

One item worth discussing is the annotation of the MSR Paraphrase Corpus. Some sentences are intentionally labeled as paraphrases in the corpus even when the small dissimilarities are extremely important, e.g. different numbers. Below is a pair of sentences from the corpus in which the “small" difference in both the numbers and the anonymous *stocks* in Text A are not considered important enough for the annotators to judge the two sentences as non-paraphrases.

Text A: *The stock rose $2.11, or about 11 percent, to close on Friday at $21.51 on the New York Stock Exchange.*

Text B: *PG&E Corp. shares jumped $1.63 or 8 percent to $21.03 on the New York Stock Exchange on Friday.*

This makes the corpus more challenging and the fully-automated solutions look less powerful than they would on a paraphrase corpus that followed the standard interpretation of what a paraphrase is, i.e. the two texts have exactly the same meaning.

Another item worth discussing is the comparison of the dependency parsers. Our experimental results show that Minipar consistently outperforms Stanford, in terms of accuracy of our paraphrase identification approach. Minipar is also faster than Stanford, which first generates the phrase-based syntactic tree for a sentence and then extracts the corresponding sets of dependencies from the phrase-based syntactic tree. For instance, Minipar can parse 1725 pairs of sentences, i.e. 3450 sentences, in 48 seconds while Stanford parser takes 1926 seconds, i.e. 32 minutes and 6 seconds. A faster parser means it could be used in interactive environments, such as Intelligent Tutoring Systems, where a fast response is needed.

Finally, we would like to discuss the impact of word weighting on our method. We weighted words by their importance as derived from Wikipedia. The reason we did not mention the IDF feature in previous sections of this article is because the results are less accurate, at least on the MSR corpus. However, we think it is informative to discuss these results as they provide more insights on the problem of paraphrase identification. In particular it highlights the difficulty of the problem and the challenging nature of sentential paraphases in general.

Corley and Mihalcea [3] suggested that word weighting could improve methods to paraphrase identification. Translated into our approach, the idea is to weight words according to their importance (or specificity) when calculating the similarity and dissimilarity scores. In general, a word is more important if it is more specific. The specificity of a word can be approximated by its IDF (Inverted Document Frequency) value calculated from a large collection of documents. The theoretical assumption for using IDF on the problem of paraphrase identification is that when a word is considered highly specific (e.g. an unusual name or a very uncommon noun), this word should play an important role when deciding paraphrasing. To further motivate this assumtion, we show below a pair of sentences extracted from the MSR test data (instance #89), where by using IDF, our method succesfully classifies an otherwise failed instance:

Text A: *Emily Church is London bureau chief of CBS.MarketWatch.com.*

Text B: *Russ Britt is the Los Angeles Bureau Chief for CBS.MarketWatch.com.*

Notice that even though the predicates are the same and there is a rather long common noun phrase, which results in a significant number of identical dependencies between the two sentences, the subjects and the locations are completely different. Because there are two different pairs of named entities, which have high IDF values, this will put a significant weight on the dissimilarity score, which in the end will lead to the decision that the two sentences are in fact not paraphrases.

We used Wikipedia, one of the largest and most diverse collection of documents freely available on the Internet, as the source for IDF values. IDF values are calculated from the DF (document frequency) of words which was extracted from over 2.2 million Wikipedia documents. To account for the data sparseness factor raised by the very high number of documents available, we calculated the IDF values from a maximum of 1 million (10^6) documents in the original collection. All DF values that exceeded the maximum number of documents were reduced to the maximum accepted value of 10^6. This means that the very few words that appeared in more than 1 million documents in Wikipedia will have the same minimal IDF value of 0. This means that the maximum absolute IDF value, for words that appeared in only one document is $\log(10^6) = 6$. In the equations below, these values are normalized.

We experimented with two aproaches with IDF weights: 1) apply IDF weights to both paired and unpaired dependencies 2) apply IDF weights only to unpaired dependencies. We adjusted our previously presented scores such that they consider the IDF values of words. We added IDF-based weights on the paired dependencies in the similarity score and IDF-based weights on the unpaired dependencies in the dissimilarity score. The weights for paired and unpaired dependencies, respectively, are calculated according to the following formulae:

$$W_{idf}(d_{(w_1,w_2)}, d_{(w_3,w_4)}) = [\sum_{i=1}^{4} idf(w_i)]/(4*6)$$

$$W_{idf}(d_{(head,mod)}) = [idf(head) + idf(mod)]/(2*6)$$

Table 4 shows results with these two IDF-based methods when used with both dependency parsers (Minipar and Stanford). We present the same performance scores as in the previous section using optimum thresholds derived from both the training and the testing data sets. An interesting observation drawn from these results is that the first IDF method works better when used on the Minipar parser, while the second method works better on the Stanford parser. Another interesting effect of IDF values can be noted by comparing the IDF-based results with results in Table 1. It seems that when only unpaired dependencies are IDF-weighted the precision increases, at the expense of lower recall.

7 Summary and conclusions

In this article, we presented a novel approach to solve the problem of paraphrase identification. The approach uses word semantics and weighted dependencies to compute degrees of similarity at word/concept level and at syntactic level between two sentences. Based on the degree of similarity, sentences are being judged as paraphrases or not. The proposed approach offers state of the art performance. In particular, the approach offers high precision due to the use of syntactic information.

References

[1] Barzilay, R., and Lee, L. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple Sequence Alignment. In *Proceedings of NAACL 2003*.

[2] Brockett, C., and Dolan, W. B. 2005. Support Vector Machines for Paraphrase Identification and Corpus Construction. In *Proceedings of the 3rd International Workshop on Paraphrasing*.

[3] Corley, C., and Mihalcea, R. 2005. Measuring the Semantic Similarity of Texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*. Ann Arbor, MI.

[4] Dagan, I., Glickman, O., and Magnini B. 2006. The PASCAL Recognising Textual Entailment Challenge. In Quiñonero-Candela, J.; Dagan, I.; Magnini, B.; d'Alché-Buc, F. (Eds.), *Machine Learning Challenges. Lecture Notes in Computer Science , Vol. 3944*, 177–190, Springer, 2006.

[5] Dolan, B.; Quirk, C.; and Brockett, C. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of COLING*, Geneva, Switzerland.

[6] Graesser, A.C.; Olney,A.; Haynes, B.; and Chipman, P. 2005. *Cognitive Systems: Human Cognitive Models in Systems Design.* Erlbaum, Mahwah, NJ. chapter AutoTutor: A cognitive system that simulates a tutor that facilitates learning through mixed-initiative dialogue.

[7] Hays, D. 1964. Dependency Theory: A Formalism and Some Observations. *Languages*, 40: 511–525.

[8] Kozareva, Z., and Montoyo, A. 2006. *Lecture Notes in Artificial Intelligence: Proceedings of the 5th International Conference on Natural Language Processing (Fin-TAL 2006).* chapter Paraphrase Identification on the basis of Supervised Machine Learning Techniques.

[9] Ibrahim, A.; Katz B.; and Lin, J. 2003. Extracting Structural Paraphrases from Aligned Monolingual Corpora. in *Proceeding of the Second International Workshop on Paraphrasing*, (ACL 2003).

[10] Iordanskaja, L.; Kittredge, R.; and Polgere, A. 1991. Natural Language Generation in Artificial Intelligence and Computational Linguistics. *Lexical selection and paraphrase in a meaning-text generation model*, Kluwer Academic.

Table 4: Performance scores when using IDF values from Wikipedia.

Method	Parser	Optimum training threshold				Optimum testing threshold			
		Acc	Prec	Recall	F-score	Acc	Prec	Recall	F-score
Sim & Diss	Minipar	0.6922	0.7133	0.8980	0.7951	0.6957	0.7418	0.8317	0.7842
	Stanford	0.7049	0.7228	0.9024	0.8026	0.7101	0.7289	0.8980	0.8047
Diss only	Minipar	0.7049	0.7450	0.8457	0.7922	0.7113	0.7246	0.9128	0.8079
	Stanford	0.7043	0.7323	0.8753	0.7975	0.7072	0.7242	0.9041	0.8042

[11] Lin, D. 1993. Principle-Based Parsing Without Overgeneration. In *Proceedings of ACL*, 112–120, Columbus, OH.

[12] Lin, D. 1995. A Dependency-based Method for Evaluating Broad-coverage Parsers. In *Proceedings of IJCAI-95*.

[13] Lintean, M.; Rus, V.; and Graesser, A. 2008. Using Dependency Relations to Decide Paraphrasing In *Proceedings of the Society for Text and Discourse Conference 2008*.

[14] Marneffe, M. C; MacCartney, B.; and Manning, C. D. 2006. Generating Typed Dependency Parsers from Phrase Structure Parses. In *Proceeding of LREC 2006*.

[15] McNamara, D.S.; Boonthum, C.; Levinstein, I. B.; and Millis, K. 2007. *Handbook of Latent Semantic Analysis.* Erlbaum, Mahwah, NJ. chapter Evaluating self-explanations in iSTART: comparing word-based and LSA algorithms, 227–241.

[16] Miller, G. 1995 WordNet: A Lexical Database of English. Communications of the ACM, v.38 n.11, p.39-41.

[17] Patwardhan, S.; Banerjee, S.; and Pedersen, T. 2003. Using Measures of Semantic Relatedness for Word Sense Disambiguation. in *Proceeding of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, February.

[18] Qiu, L.; Kan M. Y.; and Chua T. S. 2006. Paraphrase Recognition via Dissimilarity Significance Classification. In *Proceeding of EMNLP*, Sydney 2006.

[19] Rus, V.; McCarthy, P. M.; Lintean, M.; McNamara, D. S.; and Graesser, A. C. 2008. Paraphrase Identification with Lexico-Syntactic Graph Subsumption. In *Proceedings of the Florida Artificial Intelligence Research Society International Conference (FLAIRS-2008).*

[20] Rus, V.; McCarthy, P. M.; McNamara, D. S.; and Graesser, A. C. 2008. A Study of Textual Entailment. *International Journal of Artificial Intelligence Tools*, August 2008.

[21] Wu, D. 2005. Recognizing Paraphrases and Textual Entailnment using Inversion Transduction Grammars. in *Proceeding of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailnment*, Ann Arbor, MI.

[22] Zhang, Y., and Patrick, J. 2005. Paraphrase Identification by Text Canonicalization In *Proceedings of the Australasian Language Technology Workshop 2005*, 160–166.

Challenging Issues of Automatic Summarization: Relevance Detection and Quality-based Evaluation

Elena Lloret and Manuel Palomar
Department of Software and Computing Systems, University of Alicante, Spain
E-mail:{elloret, mpalomar}@dlsi.ua.es

Keywords: natural language processing, automatic summarization, relevance detection, quality-based evaluation

Received: April 12, 2009

This paper is about the Automatic Summarization task within two different points of view, focusing on two main goals. On the one hand, a study of the suitability for "The Code Quantity Principle" *in the Text Summarization task is described. This linguistic principle is implemented to select those sentences from a text, which carry the most important information. Moreover, this method has been run over the DUC 2002 data, obtaining encouraging results in the automatic evaluation with the ROUGE tool. On the other hand, the second topic discussed in this paper deals with the evaluation of summaries, suggesting new challenges for this task. The main methods to perform the evaluation of summaries automatically have been described, as well as the current problems existing with regard to this difficult task. With the aim of solving some of these problems, a novel type of evaluation is outlined to be developed in the future, taking into account a number of quality criteria in order to evaluate the summary in a qualitative way.*

Povzetek: Razvita je metoda za zbirni opis besedila, ki temelji na iskanju najpomembnejših stavkov.

1 Introduction

The high amount of electronic information available on the Internet increases the difficulty of dealing with it in recent years. Automatic Summarization (AS) task helps users condense all this information and present it in a brief way, in order to make it easier to process the vast amount of documents related to the same topic that exist these days. Moreover, AS can be very useful for neighbouring Natural Language Processing (NLP) tasks, such as Information Retrieval, Question Answering or Text Comprehension, because these tasks can take advantadge of the summaries to save time and resources [1].

A summary can be defined as a reductive transformation of source text through content condensation by selection and/or generalisation of what is important in the source [2]. According to [3], this process involves three stages: *topic identification*, *interpretation* and *summary generation*. To identify the topic in a document what systems usually do is to assign a score to each unit of input (word, sentence, passage) by means of statistical or machine learning methods. The stage of interpretation is what distinguishes extract-type summarization systems from abstract-type systems. During interpretation, the topics identified as important are fused, represented in new terms, and expressed using a new formulation, using concepts or words not found in the original text. Finally, when the summary content has been created through abstracting and/or information extraction, it requires techniques of Natural Language Generation to build the summary sentences. When an extractive approach is taken, there is no generation stage involved.

Another essential part of the Text Summarization (TS) task is how to perform the evaluation of a summary. Methods for evaluating TS can be classified into two categories [4]. The first, intrinsic evaluations, test the summary on itself. The second, extrinsic evaluations, test how the summary is good enough to accomplish some other task, for example, an Information Retrieval task. However, to determine whether an automatic, or even a human-made summary, is appropriate or not, is a subjective task which depends greatly on a lot of factors, for instance, what the summary is intended for, or to whom the summary is addessed [2].

In this paper, we focus on single-document[1] Text Summarization from an extractive point of view, and we set out two goals for this research. On the one hand, the first goal is to present a method to detect relevant sentences within a document, and therefore, select them to make up the final summary. On the other hand, the second aim of this piece of work is to discuss the current problems the automatic evaluation of summaries in a quantitative way have, so that we can outline a novel approach to measure the quality of a summary to be developed in further research.

The paper is structured as follows: Section 2 gives an overview of the Text Summarization task, describing the main criteria that have been used to determine the relevance of a sentence within a document. In Section 2.1, a new mechanism for detecting important sentences in a text, based on *"The Code Quantity Principle"* [5], is explained.

[1]Single-document differs from multi-document summarization in the number of input documents a system has, just one document or more than one, respectively.

Then, in Section 3 we analise the experiments performed and the results obtained for the approach we have proposed (Section 3.1). We also discuss current problems for evaluating summaries (Section 3.2), proposing a new qualitative model for the evaluation, by means of several quality criteria (Section 3.3). Finally, Section 4 draws the main conclusions and explains the work in progess.

2 Determining sentence's relevance in text summarization

Although there has been increased attention to different criteria such as well-formedness, cohesion or coherence when dealing with summarization [6], [7], most work in this NLP task is still concerned with detecting relevant elements of text and presenting them together to produce a final summary. As it has been previously mentioned, the first step in the process of summarization consists of identifying the topic of a document. To achieve this, the most common things systems do is to split the text into input units, usually sentences, and give them a relevance score to decide on which ones are the most important. Criteria such as *sentence position* within texts and *cue phrase indicators* [8], *word and phrase frequency* [9], [10], *query and title overlap* [11], *cohesive or lexical connectedness* [12], [13] or *discourse structure* [14] are examples of how to account for the relevance of a sentence. Furthermore, the use of a graph to obtain a representation of the text has proven effective, especially in multi-document summarization [15], [16], [17].

In contrast to all this work, this paper suggests a novel approach for determining the relevance of a sentence based on *"The Code Quantity Principle"* [5]. This principle tries to explain the relationship between syntax and information within a text. The first goal of this paper is to study whether this principle can be suitable or not as a criterion to select relevant sentences to produce a summary. This idea will be explained in detail in the next Section.

2.1 The code quantity principle within the text summarization task

"The Code Quantity Principle" [5] is a linguistic theory which states that: (1) a larger chunk of information will be given a larger chunk of code; (2) less predictable information will be given more coding material; and (3) more important information will be given more coding material. In other words, the most important information within a text will contain more lexical elements, and therefore it will be expressed by a high number of units (for instance, syllables, words or phrases). In [18], this principle have been proven to be fulfilled in written texts. Moreover, *"The Code Quantity, Attention and Memory Principle"* [19] states that the more salient and different coding information used within a text, the more reader's attention will be caught. As a result, readers will retain, keep and retrieve this kind of information more efficiently. There exists, then, a proportional relation between the relevance of information and the amount of quantity through it is coded. On the basis of this, a coding element can range from characters to phrases. A noun-phrase is the syntactic structure which allows more flexibility in the number of elements it can contain (pronouns, adjectives, or even relative clauses), and is able to carry more or less information (words) according to the user's needs. Furthermore, the longer a noun-phrase is, the more information it carries for its nucleus. For example, if a text contained two distinct noun-phrases referring to the same entity (*"the Academy of Motion Pictures Arts and Sciences"* and *"the Academy"*), the second one would lead to ambiguities. Therefore, if a summary selected this noun-phrase without having previously given more specific information about the concept, the real meaning could be misunderstood.

Starting from these principles, the approach we suggest here is to study how *"The Code Quantity Principle"* can be applied in the summarization task, to decide on which sentences of a document may contain more relevant information through its coding, and select these sentences to make up a summary. In this particular case, the lexical units considered as encoding elements are words inside a noun-phrase, without taking into account stopwords. The hypothesis is that sentences containing longer noun-phrases will be given a higher score so, at the end, the highest ranked sentences will be chosen to appear in the final summary. To identify noun-phrases within a sentence the *BaseNP Chunker*[2], which was developed at the University of Pennsylvania, was used. One important thing to take into consideration is that the use of a chunker (as well as any other NLP tool) can introduce some error rate. This tool achieves recall and precision rates of roughly 93% for base noun-phrase chunks, and 88% for more complex chunks [20]. For the experiments performed, the score for a sentence was increased by one unit, each time a word belonged to a sentence's noun-phrase. The way we compute the score of a sentence according to the length of the noun-phrase is shown in Formula 1.

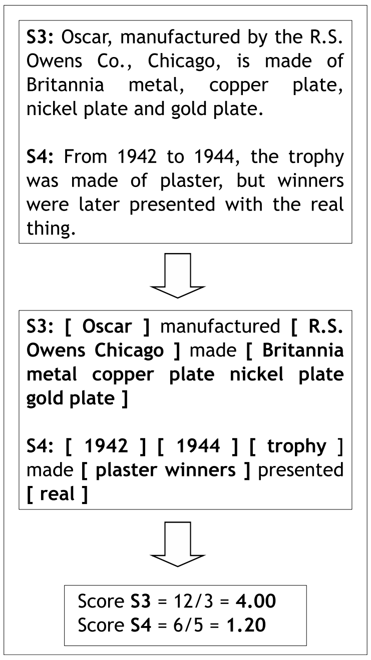
$$Sc_{s_i} = \frac{1}{\#NP_i} \sum_{w \in NP} |w| \,. \tag{1}$$

where:

#NPi = number of noun-phrases contained in sentence i,
|w |= 1, when a word belongs to a noun-phrase.

In Figure 1, an example of how we compute the score of a pair of sentences is showed. Firstly, two sentences that belong to the original document can be seen. Then, chunks of these sentences are identified and stopwords are removed from them. Lastly, scores are calculated according to Formula 1. These sentences have been extracted from the DUC 2002 test data[3]. Once we have the score for

[2]This resource is free available in ftp://ftp.cis.upenn.edu/pub/chunker/
[3]Document Understanding Conference: http://duc.nist.gov/



S3: Oscar, manufactured by the R.S. Owens Co., Chicago, is made of Britannia metal, copper plate, nickel plate and gold plate.

S4: From 1942 to 1944, the trophy was made of plaster, but winners were later presented with the real thing.

S3: [Oscar] manufactured **[R.S. Owens Chicago]** made **[Britannia metal copper plate nickel plate gold plate]**

S4: [1942] [1944] [trophy] made **[plaster winners]** presented **[real]**

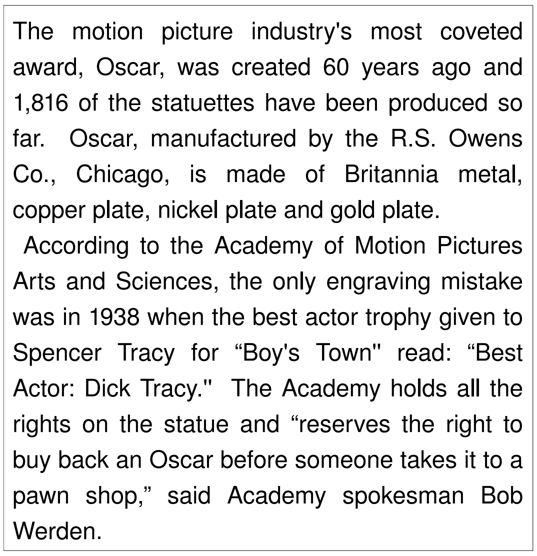
Score **S3** = 12/3 = **4.00**
Score **S4** = 6/5 = **1.20**

Figure 1: Example of sentence's scoring for document AP880217-0100

each sentence of the entire document, the sentences with the highest scores will be selected to form part of the final summary, presenting them in the same order as they were in the original text, to keep the order of the text. Figure 2 shows an example of 100-word summary using the proposed scoring method aforementioned. One particular remark of the approach suggested is how pronouns are dealt with. The use of pronouns is very common in written texts, and they substitute somebody/something that has been previously mentioned. Although they can sometimes carry important information, depending on what they are referring to, we decided not to consider them, and consequently they were treated as stopwords. The reason for taking such decision was mainly because they refer to entities previously mentioned in a document, so we strengthened the importance of those mentioned entities instead of noun-phrases containing pronouns.

3 Evaluating automatic summarization

Evaluating summaries, either manually or automatically, is a hard task. The main difficulty in evaluation comes from the impossibility of building a fair gold-standard against which the results of the systems can be compared [13]. Furthermore, it is also very hard to determine what a correct summary is, because there is always the possibility of a system to generate a good summary that is quite different from any human summary used as an approximation to the correct output [4]. In Section 1, we mentioned the two approaches that can be adopted to evaluate an automatic



The motion picture industry's most coveted award, Oscar, was created 60 years ago and 1,816 of the statuettes have been produced so far. Oscar, manufactured by the R.S. Owens Co., Chicago, is made of Britannia metal, copper plate, nickel plate and gold plate.
According to the Academy of Motion Pictures Arts and Sciences, the only engraving mistake was in 1938 when the best actor trophy given to Spencer Tracy for "Boy's Town" read: "Best Actor: Dick Tracy." The Academy holds all the rights on the statue and "reserves the right to buy back an Oscar before someone takes it to a pawn shop," said Academy spokesman Bob Werden.

Figure 2: Automatic summary for document AP880217-0100

summary: instrinsic or extrinsic evaluation. Instrinsic evaluation assesses mainly coherence and summary's information content, whereas extrinsic methods focus on determining the effect of summarization on some other task, for instance Question Answering.

Next, in Section 3.1, we show how we evaluated the novel source of knowledge and the results obtained. Afterwards, in Sections 3.2 and 3.3, we present the problems of the evaluation and the automatic methods developed so far, and we propose a novel idea for evaluating automatic summaries based on quality criteria, respectively.

3.1 The code quantity principle evaluation environment

For the approach we have suggested taking into consideration *"The Code Quantity Principle"*, we have chosen an intrinsic evaluation because we are interested in measuring the performance of the automatic summary by itself. To do this, we used the state-of-the-art measure to evaluate summarization systems automatically, ROUGE [21]. This metric measures content overlap between two summaries (normally between a gold-standard and an automatic summary), which means that the distance between two summaries can be established as a function of their vocabulary (unigrams) and how this vocabulary is used (n-grams).

In order to assess the performance of our novel approach based on *"The Code Quantity Principle"* and show that it is suitable for Text Summarization, we evaluated the summaries generated from the DUC 2002 data, consisting of 567 newswire documents. As a preprocessing step, we converted the HTML files into plain text, and we kept only the body of the news. In the DUC 2002 workshop[4], there was a task whose aim was to generate 100-word length sum-

[4]http://www-nlpir.nist.gov/projects/duc/guidelines/2002.html

maries. A set of human-made summaries written by experts was also provided. We evaluated our summaries against the reference ones, and we compared our results with the ones obtained by the systems in the real competition. Moreover, the organisation developed a simple baseline which consisted of taking the first 100 words of a document. In [22], the participating systems in DUC 2002 were evaluated automatically with the ROUGE tool, and we set up the same settings[5] for it, so that we could make a proper comparison among all the systems.

In Table 1 we can see the results of the top 3 performing DUC 2002 systems (S28, S21, S19), the baseline proposed in that workshop, and the approach we have suggested in this paper (CQPSum), only for the ROUGE-1, ROUGE-2, ROUGE-SU4 and ROUGE-L recall values. As it is shown in Table 1, the system 28 performed the best at DUC 2002, according to the ROUGE evaluation. From the 13 participating systems, there were only two systems (S28 and S21) that obtained better results than the baseline. The CQPSum approach performed slightly worse than the best system, but it performed, however, better than the rest of the participating systems in DUC 2002, including the baseline, except for the ROUGE-2 value. In S28 [23] two different algorithms, a Logistic Regression Model and a Hidden Markov Model were merged together to develop a single-document summarization system. The features this system used were: position of the sentence in the document, number of tokens in the sentence (stopwords discarded), and number of terms which were more likely to occur in the document (called "pseudo-terms"). They used a machine learning approach to train the data and afterwards, generate the final summary. In contrast, our proposal do not use any machine learning approach, and it is based on a linguistic principle using just one feature (the number of coding words that takes part in a noun-phrase) to discriminate the relevance among sentences. We have shown that this simple idea on its own performs well in the state-of-the-art of single-document summarization task. If more sources of knowledge were combined together, it could be expected that our approach would obtain better results.

3.2 Current difficulties in evaluating summaries automatically

The most common way to evaluate the informativeness of automatic summaries is to compare them with human-made model summaries. However, as content selection is not a deterministic problem, different people would chose different sentences, and even, the same person may chose different sentences at different times, showing evidence of low agreement among humans as to which sentences are good summary sentences [24]. Besides the human variability, the semantic equivalence is another problem, because two distinct sentences can express the same meaning but not using the same words. This phenomenon is known as paraphrase. In [25], we can find an approach to automatically evaluating summaries using paraphrases (ParaEval). Moreover, most summarization systems perform an extractive approach, selecting and copying important sentences from the source documents. Although humans can also cut and paste relevant information of a text, most of the times they rephrase sentences when necessary, or they join different related information into one sentence [26].

For years, the summarization community research has been actively seeking an automatic evaluation methodology. Several methods have been proposed, and thanks to the conferences carried out annually until 2007 within the DUC context[6], some of these methodologies, for instance, ROUGE [21] or the Pyramid Method [27] have been well adopted by the researchers to evaluate summaries automatically. Although ROUGE is a recall-oriented metric, the latest version (ROUGE-1.5.5) can compute precision and F-measure, too. It is based on content overlap and the idea behind it is to assess the number of common n-grams between two texts, with respect to different kinds of n-grams, like unigrams, bigrams or the longest common subsequence. In order to address some of the shortcomings of the comparison of fixed words n-grams, an evaluation framework in which very small units of content were used, called Basic Elements (BE) was developed [28].

The idea underlying the Pryamid method is to identify information with the same meaning across different human-authored summaries, which are tagged as *Summary Content Units* (SCU) in order to derive a gold-standard for the evaluation. Each SCU will have a weight depending on the number of summarizers who expressed the same information, and these weights will follow a specific distribution, allowing important content to be differentiated from less important one. The main disadvantages of this method are (1) the need to have several human-made summaries, and (2) the labourious task to annotate all the SCU. An attempt to automate the annotation of the SCUs in the pyramids can be found in [29].

More methods that perform the evaluation of automatic summaries can be found in [30] and [31]. In the former, Relative Utility (RU) is proposed as a metric to evaluate summaries, where multiple judges rank each sentence in the input with a score, giving them a value which ranged from 0 to 10, with respect to its suitability for inclusion in a summary. Highly ranked sentences would be very suitable for a summary, whereas low ranked ones should not be incuded. Like the commonly used information retrieval metric of precision and recall, it compares sentence selection between automatic and reference summaries. The latter have developed an evaluation framework, called QARLA, which provides three types of measures for the evaluation under the assumption that the best similarity metric should

[5]ROUGE version (1.5.5) run with the same parameters as in [22]: ROUGE-1.5.5.pl -n 2 -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -l 100 -d

[6]The summarization workshop will no longer be referred as DUC. From 2008, the new workshop is called Text Analysis Conference (TAC) and includes other NLP tasks apart from summarization (http://www.nist.gov/tac/).

Table 1: Results for the CQPSum approach

SYSTEM	ROUGE-1	ROUGE-2	ROUGE-SU4	ROUGE-L
S28	0.42776	0.21769	0.17315	0.38645
CQPSum	**0.42241**	**0.17177**	**0.19320**	**0.38133**
S21	0.41488	0.21038	0.16546	0.37543
DUC baseline	0.41132	0.21075	0.16604	0.37535
S19	0.40823	0.20878	0.16377	0.37351

be the one that best discriminates between manual and automatically generated summaries. These measures are: (1) a measure to evaluate the quality of any set of similarity metrics, (2) a measure to evaluate the quality of a summary using an optimal set of similarity metrics, and (3) a measure to evaluate whether the set of baseline summaries is reliable or may produce biased results.

Despite the fact that many approaches have been developed, some important aspects of summaries, such as legibility, grammaticality, responsiveness or well-formedness are still evaluated manually by experts. For instance, DUC assessors had a list of linguistic quality questions[7], and they manually assigned a mark to automatic summaries depending on what extent they accomplished each of these criteria.

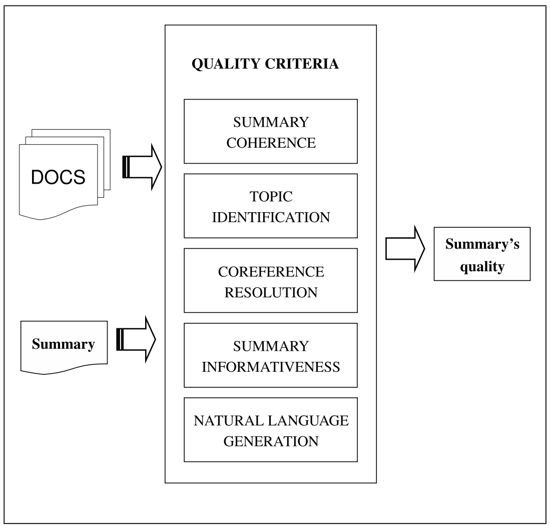


Figure 3: Quality criteria for evaluating summaries in a qualitative way

3.3 Evaluating summaries qualitatively

The main drawback of the evaluation systems existing so far is that we need at least one reference summary, and for some methods more than one, to be able to compare automatic summaries with models. This is a hard and expensive task. Much effort has to be done in order to have corpus of texts and their corresponding summaries. Furthermore, for some methods presented in the previous Section, not only do we need to have human-made summaries available for comparison, but also manual annotation has to be performed in some of them (e.g. SCU in the Pyramid Method). In any case, what the evaluation methods need as an input, is a set of summaries to serve as gold-standards and a set of automatic summaries. Moreover, they all perform a quantitative evaluation with regard to different similarity metrics. To overcome these problems, we think that the quantitative evaluation might not be the only way to evaluate summaries, and a qualitative automatic evaluation would be also important. Therefore, the second aim of this paper is to suggest a novel proposal for evaluating automatically the quality of a summary in a qualitative manner rather than in a quantitative one. Our evaluation approach is a preliminary approach which has to be studied more deeply, and developed in the future. Its main underlying idea is to define several quality criteria and check how a generated summary tackles each of these, in such a way that a reference model would not be necessary anymore, taking only into consideration the automatic summary and the original source. Once performed, it could be used together with any other automatic methodology to measure summary's informativeness.

Attempts to measure the quality of a summary have been previosuly described. In [32] indicativeness (by means of document topics) and sentence acceptability were evaluated by comparing automatic summaries with model ones. More recent approaches have suggested automatic methods to determine the coherence of a summary [33], or even an analisys of several factors regarding readability, which can be used for predicting the quality of texts [34].

As can be seen in Figure 3, the quality criteria aforementioned for the proposed methodoloy will include, among others, coherence within the summary, how anaphoric expressions have been dealt with, whether the topic has been identified correctly or not, or how language generation has been used. The final goal is to set up an independent summarization evaluation environment suitable for generic summaries, which tests a summary's quality, and decides on whether the summary is correct or not, with respect to its original document. Having available a methodology like the one proposed here, would allow automatic summaries to be evaluated automatically in an objective way on their own, without comparing them to any gold-standard in terms of more linguistic and readability aspects.

[7]http://www-nlpir.nist.gov/projects/duc/duc2007/quality-questions.txt

4 Conclusions and future work

In this paper we presented two main contributions. First of all, we studied *"The Code Quantity Principle"*, which is a linguistic theory about how humans codify the information in a text, depending on what they want a reader to pay more attention to. We presented an approach in which this principle was developed, and we ran it within a newswire domain document set, taking profit of the data provided by DUC 2002 workshop. The evaluation of this method was performed with the ROUGE tool, which made possible the comparison between automatic summaries and reference ones. The results obtained showed that our approach can be suitable for selecting important sentences of a document, and therefore can be a good idea to take this feature into account when building a summarization system. Secondly, owing to all the difficulties the summarization evaluation have, a novel manner of performing the evaluation of an automatic summary was also outlined. What we suggested was to define some quality indicators in order to assess an automatic summary in a qualitative way, rather than in a quantitative one, and therefore, determine if the generated summary can be suitable or not, with regard to its original source.

In future work, we plan to combine, on the one hand, the approach developed to select sentences according to their relevance with other sources of knowledge, such as the word-frequency, and extend this approach to multi-document summarization. Moreover, we are interested in exploring discourse structures in summarization and also, how other human languages technologies can affect the summarization process. Another research line to bear in mind for the future is to provide approaches to be developed with a Natural Language Generation module, in order to try to generate a real summary (that is an abstract, how humans would do summarization) and not only an extract. On the other hand, our second goal for the immediate future is to develop the idea outlined in this paper about evaluating automatic summaries qualitatively, with regard to specific quality criteria, starting from defining such criteria and studying how they can contribute to the evaluation of a summary.

Acknowledgement

This research has been supported by the FPI grant (BES-2007-16268) from the Spanish Ministry of Science and Innovation, under the project TEXT-MESS (TIN2006-15265-C06-01).

References

[1] Hassel, M.: Resource Lean and Portable Automatic Text Summarization. PhD thesis, Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden (2007)

[2] Spärck Jones, K.: Automatic summarising: The state of the art. Information Processing & Management **43**(6) (2007) 1449–1481

[3] Hovy, E.: Text Summarization. In: The Oxford Handbook of Computational Linguistics. Oxford University Press (2005) 583–598

[4] Mani, I.: Summarization evaluation: An overview. In: Proceedings of the North American chapter of the Association for Computational Linguistics (NAACL). Workshop on Automatic Summarization. (2001)

[5] Givón, T.: Isomorphism in the Grammatical Code. Simone, R. ed. Iconicity in Language (1994)

[6] Alonso i Alemany, L., Fuentes Fort, M.: Integrating cohesion and coherence for automatic summarization. In: EACL '03: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics. (2003) 1–8

[7] Hasler, L.: Centering theory for evaluation of coherence in computer-aided summaries. In (ELRA), E.L.R.A., ed.: Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), Marrakech, Morocco (2008)

[8] Edmundson, H.P.: New methods in automatic extracting. In: Inderjeet Mani and Mark Maybury, editors, *Advances in Automatic Text Summarization*, MIT Press (1969) 23–42

[9] Luhn, H.P.: The automatic creation of literature abstracts. In: Inderjeet Mani and Mark Maybury, editors, *Advances in Automatic Text Summarization*, MIT Press (1958) 15–22

[10] Lloret, E., Ferrández, O., Muñoz, R., Palomar, M.: A Text Summarization Approach Under the Influence of Textual Entailment. In: Proceedings of the 5th International Workshop on Natural Language Processing and Cognitive Science (NLPCS 2008) 12-16 June, Barcelona, Spain. (2008) 22–31

[11] Radev, D.R., Blair-Goldensohn, S., Zhang, Z.: Experiments in single and multi-document summarization using mead. In: First Document Understanding Conference, New Orleans, LA. (2001) 1–7

[12] Barzilay, R., Elhadad, M.: Using lexical chains for text summarization. In: Inderjeet Mani and Mark Maybury, editors, *Advances in Automatic Text Summarization*, MIT Press (1999) 111–122

[13] Fuentes Fort, M.: A Flexible Multitask Summarizer for Documents from Different Media, Domain, and Language. PhD thesis (2008) Adviser-Horacio Rodríguez.

[14] Marcu, D.: Discourse trees are good indicators of importance in text. In: Inderjeet Mani and Mark Maybury, editors, *Advances in Automatic Text Summarization*, MIT Press (1999) 123–136

[15] Mihalcea, R.: Graph-based ranking algorithms for sentence extraction, applied to text summarization. In: Proceedings of the ACL 2004 on Interactive poster and demonstration sessions. (2004) 20

[16] Radev, D.R., Erkan, G., Fader, A., Jordan, P., Shen, S., Sweeney, J.P.: Lexnet: A graphical environment for graph-based nlp. In: Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Sydney, Australia. (July 2006) 45–48

[17] Wan, X., Yang, J., Xiao, J.: Towards a unified approach based on affinity graph to various multi-document summarizations. In: Proceedings of the 11th European Conference, ECDL 2007, Budapest, Hungary. (2007) 297–308

[18] Ji, S.: A textual perspective on Givon's quantity principle. Journal of Pragmatics **39**(2) (2007) 292–304

[19] Givón, T.: A functional-typological introduction, II. Amsterdam : John Benjamins (1990)

[20] Ramshaw, L.A., Marcus, M.P.: Text chunking using transformation-based learning. In: Proceedings of the Third ACL Workshop on Very Large Corpora, Cambridge MA, USA. (1995)

[21] Lin, C.Y., Hovy, E.: Automatic evaluation of summaries using n-gram co-occurrence statistics. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL 2003). (2003) 71–78

[22] Steinberger, J., Poesio, M., Kabadjov, M.A., Ježek, K.: Two uses of anaphora resolution in summarization. Information Processing & Management **43**(6) (2007) 1663–1680

[23] Schlesinger, J.D., Okurowski, M.E., Conroy, J.M., O'Leary, D.P., Taylor, A., Hobbs, J., Wilson, H.: Understanding machine performance in the context of human performance for multi-document summarization. In: Proceedings of the DUC 2002 Workshop on Text Summarization (In conjunction with the ACL 2002 and including the DARPA/NIST sponsored DUC 2002 Meeting on Text Summarization), Philadelphia. (2002)

[24] Nenkova, A.: Summarization evaluation for text and speech: issues and approaches. In: INTERSPEECH-2006, paper 2079-Wed1WeS.1. (2006)

[25] Zhou, L., Lin, C.Y., Munteanu, D.S., Hovy, E.: Paraeval: Using paraphrases to evaluate summaries automatically. In: Proceedings of the Human Language Technology / North American Association of Computational Linguistics conference (HLT-NAACL 2006). New York, NY. (2006) 447–454

[26] Endres-Niggemeyer, B.: Summarizing Information. Berlin: Springer (1998)

[27] Nenkova, A., Passonneau, R., McKeown, K.: The pyramid method: Incorporating human content selection variation in summarization evaluation. ACM Transactions on Speech and Language Processing **4**(2) (2007) 4

[28] Hovy, E., Lin, C.Y., Zhou, L., Fukumoto, J.: Automated summarization evaluation with basic elements. In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC). Genoa, Italy. (2006)

[29] Fuentes, M., González, E., Ferrés, D., Rodríguez, H.: Qasum-talp at duc 2005 automatically evaluated with a pyramid based metric. In: the Document Understanding Workshop (presented at the *HLT/EMNLP* Annual Meeting), Vancouver, B.C., Canada. (2005)

[30] Radev, D.R., Tam, D.: Summarization evaluation using relative utility. In: CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management. (2003) 508–511

[31] Amigó, E., Gonzalo, J., Peñas, A., Verdejo, F.: QARLA: a framework for the evaluation of text summarization systems. In: ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. (2005) 280–289

[32] Saggion, H., Lapalme, G.: Selective analysis for automatic abstracting: Evaluating indicativeness and acceptability. In: Proceedings of Content-Based Multimedia Information Access (RIAO). (2000) 747–764

[33] Barzilay, R., Lapata, M.: Modeling local coherence: An entity-based approach. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), Ann Arbor, Michigan, Association for Computational Linguistics (June 2005) 141–148

[34] Pitler, E., Nenkova, A.: Revisiting readability: A unified framework for predicting text quality. In: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, Honolulu, Hawaii, Association for Computational Linguistics (October 2008) 186–195

Low-Bias Extraction of Domain-Specific Concepts

Axel-Cyrille Ngonga Ngomo
University of Leipzig, Johannisgasse 26, Leipzig D-04103, Germany
E-mail: ngonga@informatik.uni-leipzig.de,
WWW home page: http://bis.uni-leipzig.de/AxelNgonga

Keywords: natural language processing, local graph clustering, knowledge-free concept extraction

Received: February 5, 2009

The availability of domain-specific knowledge models in various forms has led to the development of several tools and applications specialized on complex domains such as bio-medecine, tourism and chemistry. Yet, most of the current approaches to the extraction of domain-specific knowledge from text are limited in their portability to other domains and languages. In this paper, we present and evaluate an approach to the low-bias extraction of domain-specific concepts. Our approach is based on graph clustering and makes no use of a-priori knowledge about the language or the domain to process. Therefore, it can be used on virtually any language. The evaluation is carried out on two data sets of different cleanness and size.

Povzetek: Od jezika neodvisna metoda iz besedila izlušči termine in nato domensko odvisne koncepte.

1 Introduction

The recent availability of domain-specific knowledge models in various forms has led to the development of information systems specialized on complex domains such as bio-medecine, tourism and chemistry. Domain-specific information systems rely on domain knowledge in forms such as terminologies, taxonomies and ontologies to represent, analyze, structure and retrieve information. While this integrated knowledge boosts the accuracy of domain-specific information systems, modeling domain-specific knowledge manually remains a challenging task. Therefore, considerable effort is being invested in developing techniques for the extraction of domain-specific knowledge from various resources in a semi-automatic fashion. Domain-specific text corpora are widely used for this purpose. Yet, most of the current approaches to the extraction of domain-specific knowledge in the form of terminologies or ontologies are limited in their portability to other domains and languages. The limitations result from the knowledge-rich paradigm followed by these approaches, i.e., from them demanding hand-crafted domain-specific and language-specific knowledge as input. Due to these constraints, domain-specific information systems exist currently for a limited number of domains and languages for which domain-specific knowledge models are available. An approach to remedy the high human costs linked with the modeling of domain-specific knowledge is the use of low-bias, i.e., knowledge-poor and unsupervised approaches. They require little human effort but more computational power to achieve the same goals as their hand-crafted counterparts.

In this work, we propose the use of low-bias approaches for the extraction of domain-specific terminology and concepts from text. Especially, we study the low-bias extraction of concepts out of text using a combination of metrics for domain-specific multi-word units and graph clustering techniques. The input for this approach consists exclusively of a domain-specific text corpus. We use the Smoothed Relative Expectation [9] to extract domain-specific multi-word units from the input data set. Subsequently we use SIGNUM [10] to compute a domain-specific lexicon. Finally, we use BorderFlow, a novel general-purpose graph clustering algorithm, to cluster the domain-specific terminologies to concepts. Our approach is unsupervised and makes no use of a-priori knowledge about language-specific patterns. Therefore, it can be applied to virtually all domains and languages. We evaluate our approach on two domain-specific data sets from the bio-medical domain. To achieve this goal, we present both a quantitative evaluation against kNN [19] and a qualitative evaluation against the MEdical Subject Headings(MESH)[1].

The remainder of this paper is structured as follows: first, we present related work on concept extraction. Then, we present our approach to the low-bias extraction of concepts using graph clustering, focusing especially on our clustering technique. Subsequently, we evaluate our concept extraction approach quantitatively and qualitatively. We conclude this paper by discussing our results and presenting some future work.

2 Related work

Approaches to concept extraction can be categorized by a variety of dimensions including units processed, data sources and knowledge support [20]. The overview of techniques for concept extraction presented in this section focuses on the knowledge support dimension. Accordingly, we differentiate between two main categories of

[1]`http://www.nlm.nih.gov/mesh`

approaches to concept extraction, namely knowledge-rich and low-bias approaches. Knowledge-rich approaches use knowledge about the structure of the data sources to process. Especially, text-based approaches include knowledge such as phrase structure, lemmas and part-of-speech to extract nouns or noun phrases as units to process [3]. The category of knowledge-rich approaches also includes supervised machine learning techniques and clustering techniques based on knowledge-rich features [11]. Knowledge-rich approaches are subject to limitations regarding their portability to other languages and domains because of the background knowledge they necessitate. Low-bias (also called knowledge-lean [20]) approaches try to remedy these problems by not using a-priori knowledge on the language to process. Rather, they make use of statistical features to extract the features of the terms which compose a concept. Clustering techniques based on low-bias features are the main constituent of this category of approaches.

An early work on low-bias concept extraction considered the use of collocation for measuring the degree of association of words [4]. A similar approach based on head modifiers and modifiers was implemented in [15]. For each term, the number of occurrences as head modifier/modifier of other terms is computed. The resulting vectorial descriptions are compared using the cosine metric. In [17], word vectors are used to describe terms in a corpus. The word vector to each term consist of all its close neighbors, i.e., of all the words which appear in the same sentence or within a larger context (e.g., a document [12]). Since the vectors generated are high-dimensional, salient features are extracted by using the Latent Semantic Analysis (LSA). Then, the cosine metric is applied to the transformed vectors to measure the correlation between the term descriptions. In [16], collocations are use to derive a concept hierarchy from a set of documents. They define a subsumption relation by stating that a term t subsumes a term t', when t appear in every document in which t' appears. Using this subsumption relation, a term hierarchy is computed automatically. A technique that generates concept hierarchies out of document hierarchies is proposed in [8]. The first step of this technique consists of selecting documents from the same domain. Then, a hierarchy of document clusters is generated by using the SOTA-Algorithm [5]. A keyword matching a Wordnet-concept is then assigned bottom-up to each cluster of the hierarchy in two steps: first, a concept representing the typical content of the documents of each leaf node is assigned to the node. In the second step, the labels of the interior nodes are assigned by using hypernyms of their children.

In all the approaches to low-bias concept extraction presented above, the terminology used for extracting concepts is commonly detected using either domain-specific knowledge such as reference vocabularies or language-specific techniques such as deep parsing. In this paper, we present a low-bias approach to concept extraction that makes no use of such a-priori knowledge.

3 An approach to low-bias concept extraction

Our approach is subdivided into two main steps. First, we extract the domain-specific terminology using no a-priori knowledge. Subsequently, we cluster to this terminology to domain-specific concepts.

3.1 Terminology extraction

The extraction of domain-specific terminology is carried out by using a combination of the SRE metric and the SIGNUM algorithm. We use the SRE metric [9] to extract domain-specific multi-word units (MWUs). This metric can efficiently detect domain-specific MWUs by using a combination of the relative expectation of co-occurrences and their distribution over the corpus. The general formula of SRE is given by

$$SRE(w) = \frac{nf(w)p(w)e^{-\frac{(d(w)-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}\sum_{i=1}^{n} f(c_1 \dots c_i * c_{i+2} \dots c_n)}, \quad (1)$$

where

- $d(w)$ is the number of documents in which w occurs,
- μ and σ^2 are the mean and the variance of the distribution of n-grams in documents respectively,
- $p(w)$ is the probability of occurrence of w in the whole corpus,
- $f(w)$ is the frequency of occurrence of w in the whole corpus and
- $c_1...c_i * c_{i+2}...c_n$ are patterns such that $ham(w, c_1...c_i * c_{i+2}...c_n) = 1$.

The results of SRE can be interpreted as a weighted graph. On this graph, we use SIGNUM [10], a local graph clustering algorithm for terminology extraction. The basic idea behind SIGNUM originates from the spreading activation principle, which has been used in several areas such as neural networks and information retrieval [2]: the simultaneous propagation of information across edges. In the case of SIGNUM, this information consists of the classification of the predecessors of each node in one of the two classes dubbed $+$ and $-$. Each propagation step consists of simultaneously assigning the predominant class of its predecessors to each node. The processing of a graph using SIGNUM thus consists of three phases: the *initialization phase*, during which each node is assigned an initial class; the *propagation phase*, during which the classes are propagated along the edges until a termination condition is satisfied, leading to the *termination phase*. The resulting categorization is then given out.

3.2 Concept extraction

For the extraction of concepts, we represent each of the domain-specific terms included in the terminology extracted priorly by its most significant co-occurrences [7] and compare these representations using the cosine metric. The resulting similarity values are used to compute a term similarity graph, which is used as input for the graph clustering algorithm BorderFlow.

4 BorderFlow

BorderFlow is a general-purpose graph clustering algorithm. It uses solely local information for clustering and achieves a soft clustering of the input graph. The definition of cluster underlying BorderFlow was proposed by [6]. They state that a cluster is a collection of nodes that have more links between them than links to the outside. When considering a graph as the description of a flow system, Flake et al.'s definition of a cluster implies that a cluster X can be understood as a set of nodes such that the flow within X is maximal while the flow from X to the outside is minimal. The idea behind BorderFlow is to maximize the flow from the border of each cluster to its inner nodes (i.e., the nodes within the cluster) while minimizing the flow from the cluster to the nodes outside of the cluster. In the following, we will specify BorderFlow for weighted directed graphs, as they encompass all other forms of non-complex graphs.

4.1 Formal specification

Let G = (V, E, ω) be a weighted directed graph with a set of vertices V, a set of edges E and a weighing function ω, which assigns a positive weight to each edge $e \in E$. In the following, we will assume that non-existing edges are edges e such that $\omega(e) = 0$. Before we describe BorderFlow, we need to define functions on sets of nodes. Let $X \subseteq V$ be a set of nodes. We define the set $i(X)$ of inner nodes of X as:

$$i(X) = \{x \in X | \forall y \in V : \omega(xy) > 0 \rightarrow y \in X\}. \quad (2)$$

The set $b(X)$ of border nodes of X is then

$$b(X) = \{x \in X | \exists y \in V \backslash X : \omega(xy) > 0\}. \quad (3)$$

The set $n(X)$ of direct neighbors of X is defined as

$$n(X) = \{y \in V \backslash X \; | \exists x \in X : \omega(xy) > 0\}. \quad (4)$$

In the example of a cluster depicted in Figure 1, $X = \{3, 4, 5, 6\}$, the set of border nodes of X is $\{3, 5\}$, $\{6, 4\}$ its set of inner nodes and $\{1, 2\}$ its set of direct neighbors.

Let Ω be the function that assigns the total weight of the edges from a subset of V to a subset of V (i.e., the flow between the first and the second subset). Formally:

$$\begin{array}{c} \Omega : 2^V \times 2^V \rightarrow \mathbb{R} \\ \Omega(X, Y) = \sum_{x \in X, y \in Y} \omega(xy). \end{array} \quad (5)$$

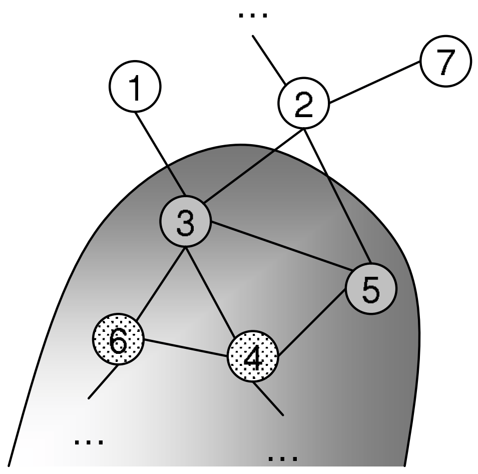


Figure 1: An exemplary cluster. *The nodes with relief are inner nodes, the grey nodes are border nodes and the white are outer nodes. The graph is undirected.*

We define the border flow ratio $F(X)$ of $X \subseteq V$ as follows:

$$F(X) = \frac{\Omega\big(b(X), X\big)}{\Omega\big(b(X), V \backslash X\big)} = \frac{\Omega\big(b(X), X\big)}{\Omega\big(b(X), n(X)\big)}. \quad (6)$$

Based on the definition of a cluster by [6], we define a cluster X as a node-maximal subset of V that maximizes the ratio $F(X)^2$, i.e.:

$$\forall X' \subseteq V, \; \forall v \notin X : X' = X + v \rightarrow F(X') < F(X). \quad (7)$$

The idea behind BorderFlow is to select elements from the border $n(X)$ of a cluster X iteratively and insert them in X until the border flow ratio $F(X)$ is maximized, i.e., until Equation (7) is satisfied. The selection of the nodes to insert in each iteration is carried out in two steps. In a first step, the set $C(X)$ of candidates $u \in V \backslash X$ which maximize $F(X + u)$ is computed is as follows:

$$C(X) := \underset{u \in n(X)}{\arg\max} \; F(X + u). \quad (8)$$

By carrying out this first selection step, we ensure that each candidate node u which produces a maximal flow to the inside of the cluster X and a minimal flow to the outside of X is selected. The flow from a node $u \in C(X)$ can be divided into three distinct flows:

- the flow $\Omega(u, X)$ to the inside of the cluster,
- the flow $\Omega(u, n(X))$ to the neighbors of the cluster and
- the flow $\Omega(u, V \backslash (X \cup n(X)))$ to the rest of the graph.

^{2}For the sake of brevity, we shall utilize the notation $X + c$ to denote the addition of a single element c to a set X. Furthermore singletons will be denoted by the element they contain, i.e., $\{v\} \equiv v$.

Prospective cluster members are elements of $n(X)$. To ensure that the inner flow within the cluster is maximized in the future, a second selection step is necessary. During this second selection step, BorderFlow picks the candidates $u \in C(X)$ which maximize the flow $\Omega(u, n(X))$. The final set of candidates $C_f(X)$ is then

$$C_f(X) := \underset{u \in C(X)}{\arg\max}\ \Omega(u, n(X)). \tag{9}$$

All elements of $C_f(X)$ are then inserted in X if the condition

$$F(X \cup C_f(X)) \geq F(X) \tag{10}$$

is satisfied.

4.2 Heuristics

One drawback of the method proposed above is that it demands the simulation of the inclusion of each node in $n(X)$ in the cluster X before choosing the best ones. Such an implementation can be time-consuming as nodes in terminology graphs can have a high number of neighbors. The need is for a computationally less expensive criterion for selecting a nearly optimal node to optimize $F(X)$. Let us assume that X is large enough. This assumption implies that the flow from the cluster boundary to the rest of the graph is altered insignificantly when adding a node to the cluster. Under this condition, the following two approximations hold:

$$\Omega(b(X), n(X)) \approx \Omega(b(X+v), n(X+v)), \tag{11}$$

$$\Omega(b(X), v) - \Omega(d(X, v), X+v) \approx \Omega(b(X), v). \tag{12}$$

Consequently, the following approximation holds:

$$\Delta F(X, v) \approx \frac{\Omega(b(X), v)}{\Omega(b(X+v), n(X+v))}. \tag{13}$$

Under this assumption, one can show that the nodes that maximize $F(X)$ maximize the following:

$$f(X, v) = \frac{\Omega(b(X), v)}{\Omega(v, V \backslash X)} \text{ for symmetrical graphs.} \tag{14}$$

Now, BorderFlow can be implemented in a two-step greedy fashion by ordering all nodes $v \in n(X)$ according to $1/f(X, v)$ (to avoid dividing by 0) and choosing the node v that minimizes $1/f(X, v)$. Using this heuristic, BorderFlow is easy to implement and fast to run.

5 Experiments and results

We evaluated our approach to concept extraction on two data sets of different cleanness and size. In the quantative evaluation, we compared the clustering generated by BorderFlow with that computed using kNN, which is the local algorithm commonly used for clustering tasks. The goal of the qualitative evaluation was to compute the quality of the clusters extracted by using BorderFlow by comparing them with the controlled MESH vocabulary.

5.1 Experimental setup

The data sets underlying the results presented in this chapter are the TREC corpus for filtering [13] and a subset of the articles published by BioMed Central (BMC[3]). Henceforth, we will call the second corpus *BMC*. The TREC corpus is a test collection composed of 233,445 abstracts of publications from the bio-medical domain. It contained 38,790,593 running word forms. The *BMC corpus* consists of full text publications extracted from the BMC Open Access library. The original documents were in XML. We extracted the text entries from the XML data using a SAX[4] Parser. Therefore, it contained a large amount of impurities that were not captured by the XML-parser. The main idea behind the use of this corpus was to test our method on real life data. The 13,943 full text documents contained 70,464,269 running word forms.

The most significant co-occurrences of the terms were computed in two steps. In a first step, we extracted function words by retrieving the f terms with the lowest information content according to Shannon's law [18]. Function words were not considered as being significant co-occurrences. Then, the s best scoring co-occurrences of each term that were not function words were extracted and stored as binary feature vectors.

5.2 Quantitative evaluation

In this section of the evaluation, we compared the average silhouettes [14] of the clusters computed by BorderFlow with those computed by kNN on the same graphs. The silhouette $\sigma(X)$ of a cluster X is given by:

$$\sigma(X) = \frac{1}{|X|} \sum_{v \in X} \frac{a(v, X) - b(v, V \backslash X)}{max\{a(v, X), b(v, V \backslash X)\}}, \tag{15}$$

where

$$a(v, X) = \frac{\sum_{v' \in n(v) \cap X} \omega(v, v')}{|n(v) \cap X|} \tag{16}$$

and

$$b(v, V \backslash X) = \max_{v' \in V \backslash X} \omega(v, v'). \tag{17}$$

To ensure that all clusters had the same maximal size k, we use the following greedy approach for each seed: first, we initiated the cluster X with the seed. Then, we sorted all $v \in n(X)$ according to their flow to the inside of the cluster $\Omega(v, X)$ in the descending order. Thereafter, we sequentially added all v until the size of the cluster reached k. If it did not reached k after adding all neighbors, the procedure was iterated with $X = X \cup n(X)$ until the size k was reached or no more neighbors were found.

One of the drawbacks of kNN lies in the need for specifying the right value for k. In our experiments, we used the average size of the clusters computed using BorderFlow as value for k. This value was 7 when clustering the TREC

[3]http://www.biomedcentral.com

[4]SAX stands for Simple Application Programming Interface for XML.

data. On the BMC corpus, the experiments with $f = 100$ led to $k = 7$, whilst the experiments with $f = 250$ led to $k = 9$. We used exactly the same set of seeds for both algorithms.

The results of the evaluation are shown in Table 1. On both data sets, BorderFlow significantly outperformed kNN in all settings. On the TREC corpus, both algorithms generated clusters with high silhouette values. BorderFlow outperformed kNN by 0.23 in the best case ($f = 100$, $s = 100$). The greatest difference between the standard deviations, 0.11, was observed when $f = 100$ and $s = 200$. On average, BorderFlow outperformed kNN by 0.17 with respect to the mean silhouette value and by 0.08 with respect to the standard deviation. In the worst case, kNN generated 73 erroneous clusters, while BorderFlow generated 10. The distribution of the silhouette values across the clusters on the TREC corpus for $f = 100$ and $s = 100$ are shown in Figure 2(a) for BorderFlow and Figure 2(b) for kNN.

The superiority of BorderFlow over kNN was better demonstrated on the noisy BMC corpus. Both algorithms generate a clustering with lower silhouette values than on TREC. In the best case, BorderFlow outperformed kNN by 0.57 with respect to the mean silhouette value ($f = 250$, $s = 200$ and $s = 400$). The greatest difference between the standard deviations, 0.18, was observed when $f = 250$ and $s = 400$. In average, BorderFlow outperformed kNN by 0.5 with respect to the mean silhouette value and by 0.16 with respect to the standard deviation. Whilst BorderFlow was able to compute a correct clustering of the data set, generating maximally 1 erroneous cluster, using kNN led to large sets of up to 583 erroneous clusters ($f = 100$, $s = 400$). Figures 2(c) and 2(d) show the distribution of the silhouette values across the clusters on the BMC corpus for $f = 100$ and $s = 100$.

5.3 Qualitative evaluation

The goal of the qualitative evaluation was to determine the quality of the content of our clusters. We focused on elucidating whether the elements of the clusters were labels of semantically related categories. To achieve this goal, we compared the content of the clusters computed by BorderFlow with the MESH taxonomy [1]. It possesses manually designed levels of granularity. Therefore, it allows to evaluate cluster purity at different levels. The purity $\varphi(X)$ of a cluster X was computed as follows:

$$\varphi(X) = \max_{C} \left(\frac{|X \cap M|}{|X \cap C^*|} \right), \qquad (18)$$

where M is the set of all mesh category labels, C is a MESH category and C^* is the set of labels of C and all its sub-categories. For our evaluation, we considered only clusters that contained at least one term that could be found in MESH.

The results of the qualitative evaluation are shown in Table 2. The best cluster purity, 89.23%, was obtained when clustering the vocabulary extracted from the TREC data with $f = 250$ and $s = 100$. In average, we obtained a lower cluster purity when clustering the BMC data. The best cluster purity using BMC was 78.88% ($f = 100$, $s = 200$). On both data sets, the difference in cluster quality at the different levels was low, showing that BorderFlow was able to detect fine-grained cluster with respect to the MESH taxonomy. Example of clusters computed with $f = 250$ and $s = 400$ using the TREC corpus are shown in Table 3.

6 Discussion

From a quantitative point of view, the average silhouette values μ on TREC were higher with lower standard deviations σ. The difference in silhouette can be conceivably explained by the higher amount of noise contained in the BMC corpus. On the TREC corpus, a higher size of the feature vectors led to a higher value μ of the average silhouette of the clusters. The same relation could be observed between the number f of function words omitted and the value of μ. The standard deviation σ was inversely proportional to the size of the feature vectors and the number of function words. The number of erroneous clusters (i.e., clusters with average silhouette value less than 0) was inversely proportional to the size of the feature vectors. This can be explained by the higher amount of information available, which led to a better approximation of the semantic similarity of the terms and, thus, to less clustering mistakes. In the worst case (f=100, s=100), 99.85% of the clusters had positive silhouettes. From a qualitative point of view, BorderFlow computed clusters with a high purity based on low-level features extracted on a terminology extracted using low-bias techniques. As expected, the average cluster purity was higher for clusters computed using the TREC data set. The results of the qualitative evaluation support the basic assumption underlying this work, i.e., it is indeed possible to extract high-quality concepts from text automatically without a-priori knowledge.

Acknowledgement

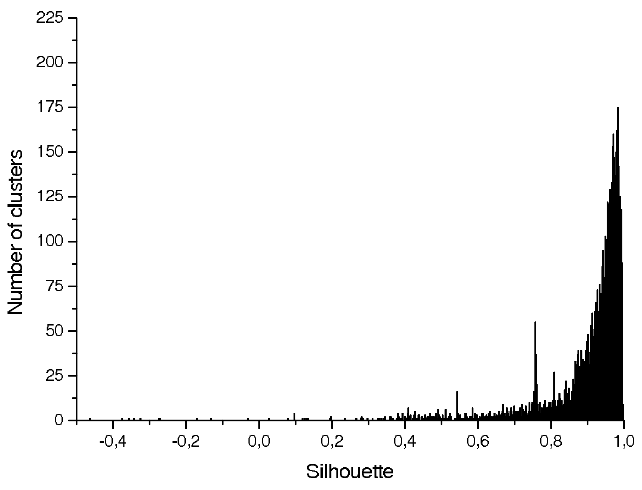
This work was supported by a grant of the German Ministry for Education and Research.

References

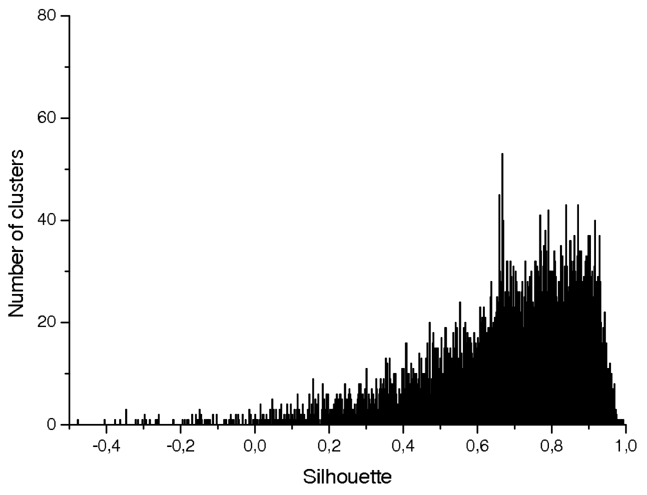
[1] S. Ananiadou and J. Mcnaught. *Text Mining for Biology and Biomedecine*. Norwood, MA, USA, 2005.

[2] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

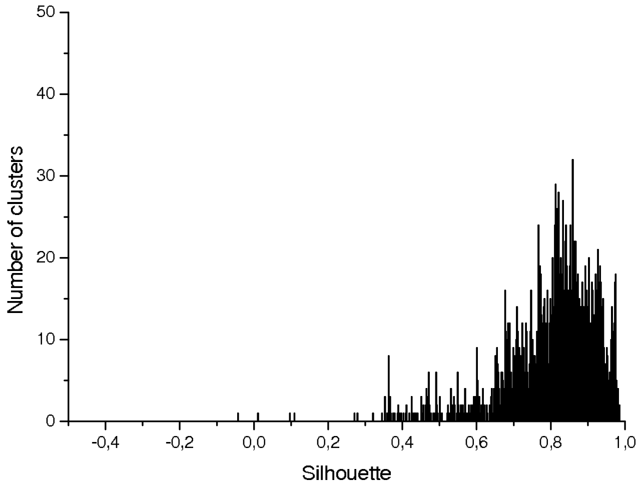
[3] C. Biemann. Ontology learning from text: A survey of methods. *LDV Forum*, 20(2):75–93, 2005.



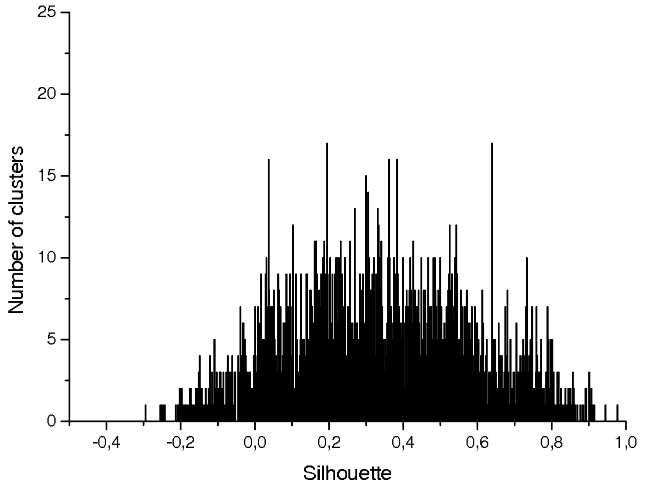
(a) BorderFlow on TREC



(b) kNN on TREC



(c) BorderFlow on BMC



(d) kNN on BMC

Figure 2: Distribution of the average silhouette values obtained by using BorderFlow and kNN on the TREC and BMC data set with f=100 and s=100

		$\mu \pm \sigma$				Erroneous clusters			
f	s	TREC		BMC		TREC		BMC	
		kNN	BF	kNN	BF	kNN	BF	kNN	BF
100	100	0.68±0.22	**0.91**±0.13	0.37±0.28	**0.83**±0.13	73	**10**	214	**1**
100	200	0.69±0.22	**0.91**±0.11	0.38±0.27	**0.82**±0.12	68	**1**	184	**1**
100	400	0.70±0.20	**0.92**±0.11	0.41±0.26	**0.83**±0.12	49	**1**	142	**1**
250	100	0.81±0.17	**0.93**±0.09	0.23±0.31	**0.80**±0.14	10	**2**	553	**0**
250	200	0.84±0.13	**0.94**±0.08	0.23±0.31	**0.80**±0.14	5	**2**	575	**0**
250	400	0.84±0.12	**0.94**±0.08	0.24±0.32	**0.80**±0.14	2	**1**	583	**0**

Table 1: Comparison of the distribution of the silhouette index over clusters extracted from the TREC and BMC corpora. BF stands for BorderFlow. μ the mean of silhouette values over the clusters and σ the standard deviation of the distribution of silhouette values. Erroneous clusters are cluster with negative silhouette silhouettes. Bold fonts mark the best results in each experimental setting.

[4] Kenneth W. Church and Patrick Hanks. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th. Annual Meeting of the Association for Computational Linguistics*, pages 76–83, Vancouver, B.C., 1989. Association for Computational Linguistics.

	f=100	f=100	f=100	f=250	f=250	f=250
Level	s=100	s=200	s=400	s=100	s=200	s=400
1	86.81	81.84	81.45	89.23	87.62	87.13
2	85.61	79.88	79.66	87.67	85.82	86.83
3	83.70	78.55	78.29	86.72	84.81	84.63
1	78.58	78.88	78.40	72.44	73.85	73.03
2	76.79	77.28	76.54	71.91	73.27	72.39
3	75.46	76.13	74.74	69.84	71.58	70.41

Table 2: Cluster purity obtained using BorderFlow on TREC and BMC data. *The upper section of the table displays the results obtained using the TREC corpus. The lower section of the table displays the same results on the BMC corpus. All results are in %.*

Cluster members	**Seeds**	**Hypernym**
b_fragilis, c_albicans, candida_albicans, l_pneumophila	c_albicans	Etiologic agents
acyclovir, methotrexate_mtx, mtx, methotrexate	methotrexate	Drugs
embryo, embryos, mouse_embryos, oocytes	embryo, embryos, mouse _embryos, oocytes	Egg cells
leukocytes, macrophages, neutrophils, platelets, pmns	platelets	Blood cells
flap, flaps, free_flap, muscle_flap, musculocutaneous_flap	flap, free_flap	Flaps
leukocyte, monocyte, neutrophil, polymorphonuclear_leukocyte	polymorphonuclear_leukocyte	White blood cells

Table 3: Examples of clusters extracted from the TREC corpus. The relation between the elements of the clusters is displayed in the rightmost column. Cluster members in italics are erroneous.

[5] J. Dopazo and J. M. Carazo. Phylogenic reconstruction using a growing neural network that adopts the topology of a phylogenic tree. *Molecular Evolution*, 44:226–233, 1997.

[6] G. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *Proceedings of the 6th ACM SIGKDD*, pages 150–160, Boston, MA, 2000.

[7] G. Heyer, M. Läuter, U. Quasthoff, T. Wittig, and C. Wolff. Learning relations using collocations. In *Workshop on Ontology Learning*, volume 38 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2001.

[8] Latifur Khan and Feng Luo. Ontology construction for information selection. In *Proceedings of the ICTAI'02*, pages 122–127, Washington DC, USA, 2002. IEEE Computer Society.

[9] A.-C. Ngonga Ngomo. Knowledge-free discovery of multi-word units. In *Proceedings of the 23rd Annual ACM Symposium on Applied Computing*, pages 1561–1565. ACM Press, 2008.

[10] A.-C. Ngonga Ngomo. SIGNUM: A graph algorithm for terminology extraction. In *Proceedings of CICLing' 2008*, pages 85–95. Springer, 2008.

[11] B. Omelayenko. Learning of ontologies for the web: the analysis of existent approaches. In *Proceedings of the International Workshop on Web Dynamics*, pages 268–275, 2001.

[12] Yonggang Qiu and Hans-Peter Frei. Concept-based query expansion. In *Proceedings of SIGIR'93*, pages 160–169, Pittsburgh, US, 1993.

[13] Stephen E. Robertson and David Hull. The TREC 2001 filtering track report. In *Proceedings of the Text REtrieval Conference*, 2001.

[14] Peter Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.

[15] G. Ruge. Experiments on linguistically-based term associations. *Information Processing and Management*, 28(3):317–332, 1992.

[16] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proceedings of SIGIR '99*, pages 206–213, New York, NY, USA, 1999. ACM.

[17] H. Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

[18] C. E. Shannon. A mathematic theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[19] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, 1 edition, 2005.

[20] L. Zhou. Ontology learning: state of the art and open issues. *Information Technology and Management*, 8(3):241–252, 2007.

Towards Multi-Stream Question Answering Using Answer Validation

Alberto Téllez-Valero, Manuel Montes-y-Gómez, Luis Villaseñor-Pineda
Laboratorio de Tecnologías del Lenguaje
Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrrique Erro no. 1, Sta. María Tonantzintla, Pue.; 72840; Mexico
E-mail: {albertotellezv,mmontesg,villasen}@inaoep.mx

Anselmo Peñas-Padilla
Depto. Lenguajes y Sistemas Informáticos
Universidad Nacional de Educación a Distancia
Juan del Rosal, 16; 28040 Madrid; Spain
E-mail: anselmo@lsi.uned.es

Keywords: question answering, information fusion, answer validation, textual entailment

Received: January 29, 2009

Motivated by the continuous growth of the Web in the number of sites and users, several search engines attempt to extend their traditional functionality by incorporating question answering (QA) facilities. This extension seems natural but it is not straightforward since current QA systems still achieve poor performance rates for languages other than English. Based on the fact that retrieval effectiveness has been previously improved by combining evidence from multiple search engines, in this paper we propose a method that allows taking advantage of the outputs of several QA systems. This method is based on an answer validation approach that decides about the correctness of answers based on their entailment with a support text, and therefore, that reduces the influence of the answer redundancies and the system confidences. Experimental results on Spanish are encouraging; evaluated over a set of 190 questions from the CLEF 2006 collection, our method responded correctly 63% of the questions, outperforming the best QA participating system (53%) by a relative increase of 19%. In addition, when they were considered five answers per question, our method could obtain the correct answer for 73% of the questions. In this case, it outperformed traditional multi-stream techniques by generating a better ranking of the set of answers presented to the users.

Povzetek: Metoda temelji na kombiniranju odgovorov več sistemov za QA.

1 Introduction

In the last two decades the discipline of Automatic Text Processing has showed an impressive progress. It has found itself at the center of the information revolution triggered by the emergence of Internet. In particular, the research in information retrieval (IR) has led to a new generation of tools and products for searching and navigating the Web. The major examples of these tools are search engines such as Google[1] and Yahoo[2]. This kind of tools allows users to specify their information needs by short queries (expressed by a set of keywords), and responds to them with a ranked list of documents.

At present, fostered by diverse evaluation forums (TREC[3], CLEF[4], and NTCIIR[5]), there are important efforts to extend the functionality of existing search engines. Some of these efforts are directed towards the development of question answering (QA) systems, which are a new kind of retrieval tools capable of answering concrete questions. Examples of pioneering Web-based QA systems are START[6] and DFKI[7].

Regardless of all these efforts, the presence of QA systems in the Web is still too small compared with traditional search engines. One of the reasons of this situation is that QA technology, in contrast to traditional IR methods, is not equally mature for all languages. For instance, in the TREC 2004, the best QA system for English achieved an accuracy of 77% for factoid questions[8] (Voorhees, 2004), whereas, two years later in the CLEF 2006, the best QA system for Spanish could only obtain an accuracy of 55% for the same kind of questions (Magnini et al, 2006). Taking into account that Spanish is the third language with more presence in the Web[9], and that it is the second language used

[1]http://www.google.com
[2]http://www.yahoo.com
[3]The Text REtrieval Conference. http://trec.nist.gov/
[4]The Cross Language Evaluation Forum. http://www.clef-campaign.org/
[5]The NTCIR Project. http://research.nii.ac.jp/ntcir/

[6]http://start.csail.mit.edu
[7]http://experimental-quetal.dfki.de
[8]Questions that asks for short, fact-based answers such as the name of a person or location, the date of an event, the value of something, etc.
[9]Internet World Stats (November 2007).

for searching it (de Sousa, 2007), these results clearly show the necessity of improving current accuracy of Spanish QA systems.

Recently, an alternative approach known as a multi-stream QA has emerged. In this approach the idea is to combine the output of different QA systems (streams) in order to obtain a better answer accuracy. This is an ideal solution due to the evidence that a perfect combination of the correct answers from several Spanish QA systems could improve by 31.5% the best individual result (Vallin et al, 2005).

In line with these efforts, in this paper we propose a new *multi-stream approach for QA*. Different to most previous methods, the proposed approach is specially suited to work with poor performance QA systems, representing the real situation in most non-English languages. In particular, it is based on an answer validation method that decides about the correctness of answers based on their entailment with a given support text. In this way the method does not rely on the stream's confidences, nor depend on the redundancy of the answers across the systems.

Our experimental results in a set of 190 questions from the CLEF 2006 collection demonstrate the appropriateness of the proposed method for combining the output of several (including poor performance) QA systems. It could correctly respond 63% of the questions, outperforming the best QA participating system (53%) by a relative increase of 19%. In addition, when we considered a set of five answers per question, our method could obtain the correct answer for 73% of the questions. In this case, it outperformed other multi-stream techniques by generating a better ranking of the set of answers presented to the users. This last characteristic is of great relevance for Web applications, where users hope to get the requested information as direct as possible.

The rest of the paper is organized as follows. Section 2 organizes the previous work in multi-stream QA. Section 3 and 4 describe our proposal for a multi-stream QA method based on an answer validation approach. Then, Section 5 presents the evaluation results of the proposed method in a set of 190 questions in Spanish language. Finally, Section 6 exposes our conclusions and outlines some future work directions.

2 Related work

Typically, QA systems consist of a single processing stream that performs three components in a sequential fashion: question analysis, document/passage retrieval, and answer selection (see e.g., (Hovy et al, 2000)). In this single-stream approach a kind of information combination is often performed within its last component. The goal of the answer selection component is to evaluate multiple candidate answers in order to choose from them the most likely answer for the question. There are several approaches for answer selection, ranging from those based on lexical overlaps and answer redundancies (see e.g., (Xu et al, 2002)) to those based on knowledge intensive methods (see e.g., (Moldovan et al, 2007)).

Recently, an alternative approach known as a multi-stream QA has emerged. In this approach the idea is to combine different QA strategies in order to increase the number of correctly answered questions. Mainly, multi-stream QA systems are of two types: internal and external.

Internal multi-stream systems use more than one stream (in this case, more than one strategy) at each particular component. For instance, Pizzato and Mollá-Aliod (2005) describes a QA architecture that uses several document retrieval methods, and Chu-Carroll et al (2003) presents a QA system that applies two different methods at each system component.

On the other hand, *external multi-stream systems* directly combine the output of different QA systems. They employ different strategies to take advantage of the information coming from several streams. Following we describe the main strategies used in external multi-stream QA systems. It is important to mention that most of these strategies are adaptations of well-known information fusion techniques from IR. Based on this fact, we propose organizing them into five general categories taking into consideration some ideas proposed elsewhere (Diamond, 1996; Vogt and Cottrell, 1999).

Skimming Approach. The answers retrieved by different streams are interleaved according to their original ranks. In other words, this method takes one answer in turn from each individual QA system and alternates them in order to construct the final combined answer list. This approach has two main variants. In the first one, that we called *Naïve Skimming Approach*, the streams are selected randomly. Whereas, in the second variant, which we called *Ordered Skimming Approach*, streams are ordered by their general confidence. In other words, QA systems are ordered by their global answer accuracy estimated from a reference question set. Some examples of QA systems that use this approach are described in (Clarke et al, 2002) and (Jijkoun and de Rijke, 2004).

Chorus Approach. This approach relies on the answer redundancies. Basically, it ranks the answers in accordance to their repetition across different streams. Some systems based on this approach are described in (de Chalendar et al, 2002), (Burger et al, 2002), (Jijkoun and de Rijke, 2004), (Roussinov et al, 2005), and (Rotaru and Litman, 2005).

Dark Horse Approach. This approach can be considered as an extension of the Ordered Skimming Approach. It also considers the confidence of streams, however, in this case, these confidences are computed separately for each different answer type. That is, using this approach, a QA system will have different confidence values associated to factoid, definition and list questions. A QA system based on this strategy is described in (Jijkoun and de Rijke, 2004).

Web Chorus Approach. This approach uses the Web information to evaluate the relevance of answers. It basically

http://www.internetworldstats.com/stats7.htm ranks the answers based on the number of Web pages containing the answer terms along with the question terms. It was proposed by Magnini et al (2001), and subsequently it was also evaluated in (Jijkoun and de Rijke, 2004).

Answer Validation Approach. In this approach the decision about the correctness of an answer is based on its entailment with a given support text. This way of answer selection not only allows assuring the rightness of answers but also their consistency with the snippets that will be showed to the users. This approach was suggested by Peñas et al (2007), and has been implemented by Glöckner et al (2007)[10].

In addition, it has also been used a combination of different approaches. For instance, Jijkoun and de Rijke (2004) describes a QA architecture that combines a chorus-based method with the dark horse approach. Its evaluation results indicate that this *hybrid approach* outperformed the results obtained by systems based on one single multi-stream strategy[11].

In this paper we propose a new multi-stream QA method based on the answer validation approach. We decide using this approach because it does not consider any confidence about the input streams as well as it does not exclusively depend on the answer redundancies. These characteristics make this approach very appropriate for working with poor performance QA systems such as those currently available for most languages except for English.

Our method distinguishes from existing answer-validation multi-stream methods (Glöckner, 2006; Tatu et al, 2006) in the following two concerns. First, it is the only one specially suited for Spanish, and second, whereas the other two methods are based on a deep semantic analysis of texts, ours is only based on a lexical-syntactic analysis of documents. We consider this last difference very important for constructing Web applications since it makes our method more easily portable across languages.

In particular, the proposed answer validation method is based on a supervised learning approach that considers a combination of two kinds of attributes. On the one hand, some attributes that indicate the compatibility between the question and the answer, and on the other hand, some attributes that allow evaluating the textual entailment between the question-answer pair and the given support text. The first kind of attributes has been previously used in traditional single-stream QA systems (e.g., (Vicedo, 2001)), whereas the second group of attributes is commonly used by answer validation (AV) and textual entailment recognition (RTE) systems (e.g., (Kozareva et al, 2006; Jijkoun and de Rijke, 2005)). In this case, our method not only considers attributes that indicate the overlap between the question-answer pair and the support text, but also includes some attributes that evaluates the non-overlapped information. In some sense, these new attributes allow analyzing the situations where exists a high overlap but not necessarily an entailment relation between these two elements.

The following section describes in detail the proposed method.

3 A multi-stream QA system based on answer validation

Figure 1 shows the general scheme of the proposed external multi-stream QA System. It uses an *answer validation module* to superficially combine the outputs (answers) from several streams (QA systems).

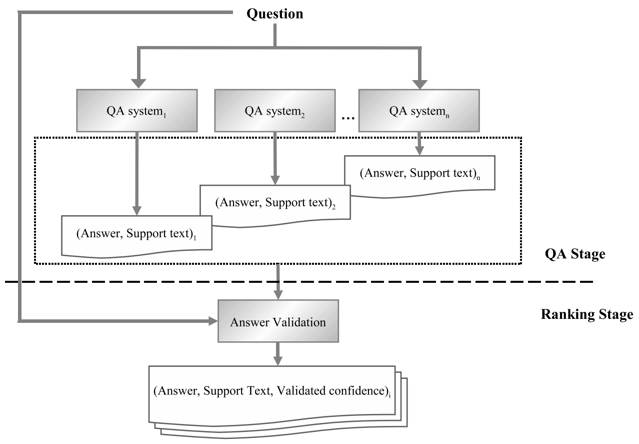

Question
QA system$_1$
QA system$_2$
...
QA system$_n$
(Answer, Support text)$_n$
(Answer, Support text)$_2$
(Answer, Support text)$_1$
QA Stage
Ranking Stage
Answer Validation
(Answer, Support Text, Validated confidence)$_i$

Figure 1: Multi-stream QA System based on Answer Validation

Mainly, the proposed multi-stream QA system consists of two main stages. In the first stage (called QA stage), several different QA systems extract — in parallel — a candidate answer and its corresponding support text for a given question. Then, in the second stage (called ranking stage), an answer validation module evaluates — one by one — the candidate answers and assigns them a confidence value from 0 to 1. A confidence value equal to 0 indicates that the answer is totally rejected, whereas a confidence equal to 1 indicates that the answer is completely accepted. At the end, answers are ranked in line with their confidence values.

The following section describes in detail the answer validation method. This method is an extension of our previous work described in Téllez-Valero et al (2007). In particular, it includes a novel set of attributes for answer validation which allow to increase our previous results by 14% as well as to outperform all results reported in the 2006 Spanish Answer Validation Exercise (Peñas et al, 2006).

4 The answer validation module

Given a question (Q), a candidate answer (A) and a support text (S), the answer validation module returns a confidence value (β) that allows deciding whether to accept or reject

[10](Harabagiu and Hickl, 2006) also describes an answer validation approach for multi-stream QA; nevertheless, it is an internal approach.

[11]They compared their hybrid method against all other approaches except the answer validation.

the candidate answer. In other words, it helps to determine if the specified answer is correct and if it can be deduced from the given support text.

Our answer validation module is mainly based on the idea of recognizing the textual entailment (RTE) between the support text (*T*) and an affirmative sentence (*H*) called hypothesis, created from the combination of the question and the answer. The entailment between the pair (*T, H*) occurs when the meaning of *H* can be inferred from the meaning of *T* (Dagan et al, 2005).

The returned confidence value is generated by means a supervised learning approach that considers three main processes: preprocessing, attribute extraction and answer classification. The following sections describe each one of these processes.

4.1 Preprocessing

The objective of this process is to extract the main content elements from the question, answer and support text, which will be subsequently used for deciding about the correctness of the answer. This process considers two basic tasks: on the one hand, the identification of the main constituents from the question-answer pair, and on the other hand, the detection of the core fragment of the support text as well as the consequent elimination of the unnecessary information.

4.1.1 Constituent identification

We detect three basic constituents from the questions: its main action, the action actors, and if exist, the action restriction. As an example, consider the question in Table 1. In this case, the action is represented by the verb `invade`, its actors are the syntagms `Which country` and `Iraq`, and the action restriction is described by the propositional syntagma `in 1990`.

In order to detect the question constituents we firstly apply a shallow parsing to the given question. Then, from the resulting syntactic tree (Q_{parsed}), we construct a new representation of the question (called *Q'*) by detecting and tagging the following elements:

1. *The action constituent.* It corresponds to the syntagm in Q_{parsed} that includes the main verb.

2. *The restriction constituent.* It is represented by the prepositional syntagm in Q_{parsed} having at least one explicit time expression (e.g., `in 1990`), or including a preposition such as `after` or `before`.

3. *The actors constituents.* These constituents are formed by the rest of the elements in Q_{parsed}. It is commonly divided in two parts. The first one, henceforth called *hidden actor constituent*, corresponds to the syntagm that includes the interrogative word and it is generally located at the left of the action constituent. The second part, which we call the *visible actor constituent*, is formed by the rest of the syntagms, generally located at the right of the action constituent.

Question	`Which country did Iraq invade in 1990?`
Candidate answer	`Kuwait`
Support text	`Kuwait was a close ally of Iraq during the Iraq-Iran war and functioned as the country's major port once Basra was shut down by the fighting. However, after the war ended, the friendly relations between the two neighboring Arab countries turned sour due to several economic and diplomatic reasons which finally culminated in an Iraqi invasion of Kuwait.`
Relevant support text	`Iraqi invasion of Kuwait`

Table 1: Example of excessive text to accept or reject an answer

Finally, we also consider an *answer constituent*, which is simply the lemmatized candidate answer (denoted by *A'*).

4.1.2 Support text's core fragment detection

Commonly, the support text is a short paragraph — of maximum 700 bytes according to CLEF evaluations — which provides the context necessary to support the correctness of a given answer. However, in many cases, it contains more information than required, damaging the performance of RTE methods based on lexical-syntactic overlaps. For instance, the example of Table 1 shows that only the last sentence (a smaller text fragment) is useful for validating the given answer, whereas the rest of the text only contribute to produce an irrelevant overlap.

In order to reduce the support text to the minimum useful text fragment we proceed as follows:

– First, we apply a shallow parsing to the support text, obtaining the syntactic tree (S_{parsed}).

– Second, we match the content terms (nouns, verbs, adjectives and adverbs) from question constituents against the terms from S_{parsed}. In order to capture the morphological inflections of words we compare them using the Levenshtein edition distance [12]. Mainly, we

[12]The Levenshtein edition distance has been previously used in other works related to answer validation in Spanish language, see for instance (Rodrigo et al, 2006).

consider that two different words are equal if its distance value is greater than 0.6.

- Third, based on the number of matched terms, we align the question constituents with the syntagms from the support text.
- Forth, we match the answer constituent against the syntactic tree (S_{parsed}). The idea is to find all occurrences of the answer in the given support text.
- Fifth, we determine the minimum context of the answer in the support text that contains all matched syntagms. This minimum context (represented by a sequence of words around the answer) is what we call the *core fragment*. In the case that the support text includes several occurrences of the answer, we select the one with the smallest context.

Applying the procedure described above we determine that the core fragment of the support text showed at Table 1 is `in an Iraqi invasion of Kuwait`

4.2 Attribute extraction

This stage gathers a set of processes that allow extracting several attributes from the question, the answer and the support text. These attributes can be categorized in two different groups: the attributes that indicate the relation between the question and the answer, and the attributes that measure the entailment relation between the question-answer pair and the support text.

The following sections describe both kinds of attributes and explain the way they are calculated from *Q'*, *A'* and *T'*.

4.2.1 Attributes about the question-answer relation

Question characteristics

We consider three different attributes from the question: the question category (factoid or definition), the expected answer type (date, quantity, name or other), and the type of question restriction (date, period, event, or none).

The question category and the expected answer type are determined using a set of simple lexical patterns. Some of these patterns are showed below. It can be observed that each of them includes information about the question category and the expected answer type.

`What is` [*whatever*] → DEFINITION-OTHER
`Who is` [*whatever*] → DEFINITION-PERSON
`How many` [*whatever*] → FACTOID-QUANTITY
`When` [*whatever*] → FACTOID-DATE

On the other hand, the value of the question restriction (date, period, event or none) depends on the form of the restriction constituent. If this constituent contains only one time expression, then this value is set to "date". In the case the restriction constituent includes two time expressions, it is set to "period". If the restriction constituent does not include any time expression, then the question restriction is defined as "event". Finally, when the question does not have any restriction constituent, the value of the question restriction is set to "none".

Question-answer compatibility

This attribute indicates if the question and answer types are compatible. The idea of this attribute is to capture the situation where the semantic class of the evaluated answer does not correspond to the expected answer type. For instance, having the answer `yesterday` for the question `How many inhabitants are there in Longyearbyen?`.

This is a binary attribute: it is equal to 1 when the answer corresponds to the expected answer type, and it is equal to 0 if this correspondence does not exist.

Answer redundancy

Taking into account the idea of "considering candidates as allies rather than competitors" (Dalmas and Webber, 2007), we decided to include an attribute related to the occurrence of the answers across streams.

Different from the Chorus Method (refer to Section 2) that directly uses the frequency of occurrence of the answers across streams, the proposed attribute indicates the average similarity of the candidate answer with the rest of stream answers (it takes values from 0 to 1).

In order to deal with the great language variability and also with the presence of some typing errors, we decide using the Levenshtein edition distance to measure the similarity between answers. Using this strategy, the answer `X` contributes to the redundancy rate of the answer `Y` and vice versa.

4.2.2 Attributes related to the textual entailment recognition

The attributes of this category are of two main types: (i) attributes that measure the overlap between the support text and the hypothesis (an affirmative sentence formed by combining the question and the answer); and (ii) attributes that denote the differences between these two components.

It is important to explain that, different from other RTE methods, we do not use the complete support text, instead we only use its core fragment *T'*. On the other hand, we neither need to construct an hypothesis text, instead we use as hypothesis the set of question-answer constituents (the union of *Q'* and *A'*, which we call *H'*).

Overlap characteristics

These attributes express the degree of overlap —in number of words — between *T'* and *H'*. In particular, we compute the overlap for each type of content term (nouns, verbs, adjectives and adverbs) as well as for each type of named entity (names of persons, places, organizations, and other things, as well as dates and quantities). In total we generate 10 different overlap attributes.

Non-overlap characteristics

These attributes indicate the number of non-overlapped terms from the core fragment of the support text, that is, the number of terms from *T'* that are not present in *H'*.

Similar to the previous kind of attributes, for this case we also compute the non-overlap for each type of content term (nouns, verbs, adjectives and adverbs) as well as for each type of named entity (names of persons, places, organizations, and other things, as well as dates and quantities). In total we generate 10 different non-overlap attributes.

4.2.3 Answer classification

This final process generates the answer validation decision by means of a supervised learning approach. In particular, it applies a boosting ensemble formed by ten decision tree classifiers[13].

The constructed classifier decides whether to accept or reject the candidate answer based on the twenty-five attributes described in the previous section. In addition, it also generates a validation confidence (β) that indicates how reliable is the given answer in accordance to the support text.

5 Experimental results

5.1 Experimental setup

5.1.1 Training and test data

As we describe in section 3, the core component of the proposed multi-stream QA method is the answer validation module, which relies on a *supervised learning* approach.

In order to train this module we used the SPARTE corpus. This corpus was build from the Spanish corpora used at CLEF for evaluating QA systems from 2003 to 2005. It contains 2962 training instances represented by the tuple (⟨question⟩, ⟨answer⟩, ⟨support-text⟩, ⟨entailment-value⟩), where ⟨entailment-value⟩ is a binary variable indicating whether the support text entails or not the question-answer pair.

One important fact about this corpus is that it is very unbalanced: 77% of the training instances are negative (their entailment value is FALSE), whereas just 695 instances (the rest 23%) correspond to positive entailment examples.

On the other hand, for evaluating the proposed method, we used a set of 190 questions and the answers from 17 different QA systems (i.e., 17 different streams). In total, we considered 2286 candidate answers with their corresponding support texts.

[13]We used the Weka implementations for the AdaBoot and ADTree algorithms (Witten and Frank, 1999).

The used test set gathers the outputs from all QA systems participating at the QA track of CLEF 2006 (Magnini et al, 2006), and it was employed at the first Spanish Answer Validation Exercise (Peñas et al, 2006).

5.1.2 Evaluation measure

The evaluation measure most commonly used in QA is the *accuracy*, i.e., the percentage of correctly answered questions. Following the CLEF evaluation criteria, this measure is calculated as the fraction of correct answers and correct nil-answers[14] with respect to the total number of questions (see formula 1).

$$Accuracy = \frac{|\text{right\_answers}| + |\text{right\_}\textit{nil's}|}{|\text{questions}|} \quad (1)$$

In particular, in our experiments we used an evaluation measure called *accuracy@N*, which basically indicates the accuracy of a QA system when considering *N* candidate answers for each question. In this case, an answer is evaluated as correct if it occurs in the list of *N* candidate answers, independently of its position.

5.1.3 Results from the input streams

Table 2 shows some data from the input streams. It mainly presents their number of right and wrong answers as well as their accuracy for each different type of question. From this table, it is noticeable that most QA systems (streams) have a very poor performance level, having an average accuracy of 26%.

5.2 Experiments

5.2.1 First experiment: general evaluation of the proposed method

The objective of an external multi-stream QA method is to combine the responses from different QA systems in order to increase the final answer accuracy. In other words, its goal is to obtain a better result than that from the best input stream.

In a first experiment, we attempted to evaluate the fulfillment of this objective. We compared the results obtained by our method with the accuracy from the *best input stream* (53%). In addition, we also compared our method against other multi-stream approaches (refer to Section 2). In particular, we implemented some methods from these approaches based on the following criteria:

- *The Naïve Skimming Method.* In this case, streams maintain the order showed in Table 2.

[14]Nil questions do not have an answer in the target document collection, or even worst, they do not have any possible answer. As an example consider the question `What is the capital of Neverland?`. For these questions give no answer is considered as a correct response.

	Right		Wrong	
Stream	answers	*nil's*	answers	Accuracy
1	25	16	77	0.22
2	48	17	46	0.34
3	49	7	113	0.30
4	34	10	92	0.23
5	10	1	179	0.06
6	24	5	142	0.15
7	16	3	138	0.10
8	88	12	69	**0.53**
9	31	7	125	0.20
10	26	10	125	0.19
11	15	11	78	0.14
12	85	12	63	0.51
13	33	10	88	0.23
14	21	18	34	0.21
15	57	13	89	0.37
16	45	12	102	0.30
17	64	16	55	0.42

Table 2: Results from the input streams

– *The Ordered Skimming Method.* It ranks the answers in accordance to the stream's overall accuracies (refer to the last column of Table 2).

– *The Chorus Method.* It ranks the answers based on their repetitions across different streams.

– *The Dark Horse Method.* It uses the factoid accuracies to rank answers corresponding to factoid questions and the definition accuracies for ranking the answers for definition questions (refer to the antepenultimate and penultimate columns of Table 2.

– *The Web Chorus Method.* It ranks the answers based on the number of Web pages that contain the terms of the question (without the question word) along with the terms of the answer.

Table 3 shows the results from the first experiment. This table also includes the accuracy corresponding to a *perfect combination* of the correct answers from all streams (87%). This value indicates the maximum reachable accuracy for a multi-stream approach in this data set.

The results from this first experiment show that our method was the only multi-stream approach that could improve the result from the best input stream; it responded correctly 58% of the questions outperforming the best individual result (53%) by a relative increase of 9%. Considering a list of five candidate answers (which is the typical configuration of existing online QA systems) our method outperformed the accuracy from the best input stream by 11%, a relative improvement of 18%.

The methods that rank answers based on the stream confidences, namely the Ordered Skimming Method and the Dark Horse Method, also obtained relevant results. However, it is necessary to mention that – in our implementations – these methods made use of a *perfect estimation* of these confidences[15]. For that reason, and given that in a real scenario it is practically impossible to obtain these perfect estimations, we consider that our proposal is more robust than these two methods.

The results from Table 3 also give evidence that the presence of several deficient streams (which generate a lot of incorrect answers) seriously affects the performance of the Naïve Skimming Method. This phenomenon also had an important effect over the Chorus Method, which normally is reported as one of the best multi-stream approaches.

Finally, it is important to comment that we attribute the poor results achieved by the Web Chorus Method to the quantity of online information for Spanish (which it is considerably less than that for English). In order to obtain better results it is necessary to apply some question/answer expansions, using for instance synonyms and hyperonyms.

5.2.2 Second experiment: the impact of rejecting less reliable answers

Taking into account that the accuracy of QA systems not only depends on the number of correctly answered questions, but also on the number of correctly *unanswered* nil questions, we decided to modify the basic multi-stream methods (including ours) in order to allow them rejecting some answers. The idea was to incorporate some filtering conditions that obligate the methods to eliminate the less reliable answers. In the cases that no answer could satisfy these conditions, the answer was set to nil. Following we describe the modifications incorporated to each one of the methods.

– *Ordered Skimming Method*.* It only considers answers from the best five streams (i.e., it only returns answers coming from the streams with the five highest global accuracies).

– *Chorus Method*.* It only considers answers recommended by two or more streams.

– *Dark Horse Method*.* It only returns answers coming from the best five streams for each question type. In this case there were selected the best five streams for answering factoid questions and the best five for answering definition questions.

– *Our Method*.* It only returns answers with a validation confidence greater than 0.5.

Table 4 shows the results from this second experiment. It is interesting to notice that all methods improved their results when they rejected some answers. The explanation of this behavior is that with these modifications all methods

[15]The confidences were calculated directly from the test set (refer to Table 2). It was so because there is no correspondence between the systems that were used to generate the train and test sets.

	Number of answers by question				
	1	2	3	4	5
Naïve Skimming Method	0.25	0.44	0.50	0.55	0.61
Ordered Skimming Method	0.52	0.63	0.66	0.68	0.71
Chorus Method	0.53	0.61	0.66	0.68	0.68
Dark Horse Method	0.52	0.61	0.67	0.68	0.72
Web Chorus Method	0.17	0.28	0.37	0.46	0.56
Our Method	**0.58**	**0.65**	**0.69**	**0.71**	**0.73**
Best Input Stream	0.53	0.56	0.58	0.62	0.62
Perfect Combination	0.87	-	-	-	-

Table 3: Results from the first experiment: general evaluation of the proposed method

could answer some nil questions. In particular, our method correctly respond 63% of the questions outperforming the best input stream by a relative increase of 19%.

This experiment also helped to reveal another important characteristic of our method. It could correctly reject several answers without using any information about the confidence of streams and without considering any restriction on the answer frequencies.

5.2.3 Third experiment: combination of our method with other approaches

In (Jijkoun and de Rijke, 2004), Jijkoun and de Rijke describe a multi-stream QA architecture that combines the Chorus and the Dark Horse Methods. Its evaluation results indicate that this combination outperformed the results obtained by other systems based on one single multi-stream strategy[16].

Motivated by this result, we designed a third experiment which considered the combination of our method with other confidence-based methods, in particular, the Dark Horse Method and the Ordered Skimming Method. The combination of our method with these two other approaches was performed as follows. In a first stage, our method selected a set of candidate answers, then, in a second stage, a confidence-based method ordered the candidate answers in accordance to their own ranking criteria[17].

Table 5 shows the results from the combination of these methods. On the one hand, these results confirm the conclusions of Jijkoun and de Rijke since they also indicate that the combination of methods outperformed the results obtained by individual approaches. On the other hand, and most important, these results demonstrate the competence of our method since they show that its individual result outperformed that from the combination of the Chorus Method with the Dark Horse Method (stated by Jijkoun and de Rijke as the best configuration for a multi-stream QA system).

[16]In their experiments, as mentioned in Section 2, they did not consider the answer validation approach.

[17]Given that we use the same implementations for the confidence-based methods that those described in the first experiment, in this case we also used a perfect estimation of the streams confidences.

6 Conclusions and future work

In this paper we proposed a new external multi-stream QA method. This method is founded on the idea of combining the output of different QA systems (streams) in order to obtain a better answer accuracy.

The proposed method is based on an answer validation approach. This way, it decides about the correctness of the answers based on their entailment with a support text, and does not exclusively rely on answer redundancies nor on the stream confidences. In addition, this method only considers lexical-syntactic information and does not make use of a deep semantic analysis of texts. All these features together make our method appropriate for dealing with poor performance QA systems which represent the current state for most non-English languages. In particular, we have evaluated our method in Spanish, where current average answer accuracy is of 26% (please refer to Table 2).

The core component of the proposed multi-stream method is the answer validation module. This module applies a supervised approach for recognizing the textual entailment. It mainly uses a set of attributes that capture some simple relations among the question, the answer and the given supported text. In particular, it considers some novel attributes that characterize: (i) the compatibility between question and answer types; (ii) the redundancy of answers across streams; and (iii) the overlap (as well as the non-overlap) between the question-answer pair and the support text. At this point, it is important to comment that an evaluation of the proposed attributes during the development phase — using the information gain algorithm — showed us that the non-overlap and answer-redundancy attributes were the most discriminative.

From the evaluation results achieved on a test set of 190 Spanish questions from the CLEF-2006 QA collection, we could observe the following:

– The proposed method significantly enhanced the accuracy from the best individual stream. It correctly responded to 63% of the questions, outperforming the best QA participating system (53%) by a relative increase of 19%.

– Although our method also takes advantage of the re-

	Number of answers by question				
	1	2	3	4	5
Ordered Skimming Method*	0.55	0.66	0.69	0.70	0.71
Chorus Method*	0.58	0.64	0.67	0.67	0.67
Dark Horse Method*	0.55	0.64	0.68	0.69	0.69
Our Method*	**0.63**	**0.69**	**0.72**	**0.73**	**0.73**
Best input stream	0.53	0.56	0.58	0.62	0.62
Perfect Combination	0.87	-	-	-	-

Table 4: Results from the second experiment: the impact of rejecting less reliable answers

	Number of answers by question				
	1	2	3	4	5
Chorus Method* + Ordered Skimming Method	0.63	0.65	0.66	0.67	0.67
Chorus Method* + Dark Horse Method	0.62	0.65	0.66	0.67	0.67
Our Method* + Ordered Skimming Method	**0.64**	**0.71**	**0.72**	**0.73**	**0.73**
Our Method* + Dark Horse Method	0.63	0.69	**0.72**	**0.73**	**0.73**
Best Input Stream	0.53	0.56	0.58	0.62	0.62
Perfect Combination	0.87	-	-	-	-

Table 5: Results from the third experiment: combination of our method with other approaches

dundancy of answers across streams, it turned out to be less sensible to their low frequency than other approaches. For instance, it outperformed the Chorus Method by 5%.

– The proposed method allowed to significantly reduce the number of wrong answers presented to the user. In relation to this aspect, our method was especially adequate to deal with nil questions. It correctly responded 65% of the nil questions, outperforming the best input stream by a relative increase of 8%.

– The combination of our method with the Dark Horse approach only produced a slightly improvement of 1%. This fact indicates that our method does not require knowing the input stream confidences.

Finally, it is clear that any improvement in the answer validation module will directly impact the performance of the proposed multi-stream method. Hence, our future work will be mainly focused on enhancing this module by: (i) considering some new features in the entailment recognition process, (ii) including a process for treatment of temporal restrictions, and (iii) using Wordnet in order to consider synonyms and hyperonyms for computing the term and structure overlaps.

Acknowledgement

This work was done under partial support of Conacyt (scholarship 171610). We also thank CLEF organizers for the provided resources.

References

Burger JD, Ferro L, Greiff WR, Henderson JC, Mardis S, Morgan A, Light M (2002) MITRE's qanda at TREC-11. In: TREC

Chu-Carroll J, Czuba K, Prager JM, Ittycheriah A (2003) In question answering, two heads are better than one. In: HLT-NAACL

Clarke CLA, Cormack GV, Kemkes G, Laszlo M, Lynam TR, Terra EL, Tilker PL (2002) Statistical selection of exact answers (multitext experiments for TREC 2002). In: TREC

Dagan I, Glickman O, Magnini B (2005) The PASCAL recognising textual entailment challenge. In: Quiñonero J, Dagan I, Magnini B, d'Alché Buc F (eds) MLCW, Springer, Lecture Notes in Computer Science, vol 3944, pp 177–190

Dalmas T, Webber BL (2007) Answer comparison in automated question answering. J Applied Logic 5(1):104–120

de Chalendar G, Dalmas T, Elkateb-Gara F, Ferret O, Grau B, Hurault-Plantet M, Illouz G, Monceaux L, Robba I, Vilnat A (2002) The question answering system QALC at LIMSI, experiments in using web and wordnet. In: TREC

de Sousa P (2007) El español es el segundo idioma más usado en las búsquedas a través de google (english translated: The spanish is the second language most used in the google's searches). CDT internet.net, URL `http://www.cdtinternet.net/`

Diamond T (1996) Information retrieval using dynamic evidence combination, PhD Dissertation Proposal, School of Information Studies, Syracuse University

Glöckner I (2006) Answer validation through robust logical inference. In: Peters et al (2007), pp 518–521

Glöckner I, Sven H, Johannes L (2007) Logical validation, answer merging and witness selection - a case study in multi-stream question answering. In: RIAO 2007, Large-Scale Semantic Access to Content, Pittsburgh, USA

Harabagiu SM, Hickl A (2006) Methods for using textual entailment in open-domain question answering. In: ACL, The Association for Computer Linguistics

Hovy EH, Gerber L, Hermjakob U, Junk M, Lin CY (2000) Question answering in webclopedia. In: TREC

Jijkoun V, de Rijke M (2004) Answer selection in a multi-stream open domain question answering system. In: McDonald S, Tait J (eds) ECIR, Springer, Lecture Notes in Computer Science, vol 2997, pp 99–111

Jijkoun V, de Rijke M (2005) Recognizing textual entailment: Is word similarity enough? In: Candela JQ, Dagan I, Magnini B, d'Alché Buc F (eds) MLCW, Springer, Lecture Notes in Computer Science, vol 3944, pp 449–460

Kozareva Z, Vázquez S, Montoyo A (2006) University of alicante at qa@clef2006: Answer validation exercise. In: Peters et al (2007), pp 522–525

Magnini B, Negri M, Prevete R, Tanev H (2001) Is it the right answer?: exploiting web redundancy for answer validation. In: ACL, The Association for Computational Linguistics, Morristown, NJ, USA, pp 425–432

Magnini B, Giampiccolo D, Forner P, Ayache C, Jijkoun V, Osenova P, Peñas A, Rocha P, Sacaleanu B, Sutcliffe RFE (2006) Overview of the clef 2006 multilingual question answering track. In: Peters et al (2007), pp 223–256

Moldovan DI, Clark C, Harabagiu SM, Hodges D (2007) Cogex: A semantically and contextually enriched logic prover for question answering. J Applied Logic 5(1):49–69

Peñas A, Rodrigo Á, Sama V, Verdejo F (2006) Overview of the answer validation exercise 2006. In: Peters et al (2007), pp 257–264

Peñas A, Rodrigo Á, Sama V, Verdejo F (2007) Testing the reasoning for question answering validation. Journal of Logic and Computation (3), DOI 10.1093/logcom/exm072

Peters C, Clough P, Gey FC, Karlgren J, Magnini B, Oard DW, de Rijke M, Stempfhuber M (eds) (2007) Evaluation of Multilingual and Multi-modal Information Retrieval, 7th Workshop of the Cross-Language Evaluation Forum, CLEF 2006, Alicante, Spain, September 20-22, 2006, Revised Selected Papers, LNCS, vol 4730, Springer

Pizzato LAS, Mollá-Aliod D (2005) Extracting exact answers using a meta question answering system. In: Proceedings of the Australasian Language Technology Workshop, Sydney, Australia, pp 105–112

Rodrigo Á, Peñas A, Herrera J, Verdejo F (2006) The effect of entity recognition on answer validation. In: Peters et al (2007), pp 483–489

Rotaru M, Litman DJ (2005) Improving question answering for reading comprehension tests by combining multiple systems. In: Proceedings of the American Association for Artificial Intelligence (AAAI) 2005 Workshop on Question Answering in Restricted Domains, Pittsburgh, PA.

Roussinov D, Chau M, Filatova E, Robles-Flores JA (2005) Building on redundancy: Factoid question answering, robust retrieval and the "other". In: Proceedings of the Thirteenth Text REtreival Conference (TREC 2005), pp 15–18

Tatu M, Iles B, Moldovan DI (2006) Automatic answer validation using cogex. In: Peters et al (2007), pp 494–501

Téllez-Valero A, Montes-y-Gómez M, Villaseñor-Pineda L (2007) A supervised learning approach to spanish answer validation. In: Peters C, Jijkoun V, Mandl T, Müller H, Oard DW, Peñas A, Petras V, Santos D (eds) CLEF, Springer, Lecture Notes in Computer Science, vol 5152, pp 391–394

Vallin A, Magnini B, Giampiccolo D, Aunimo L, Ayache C, Osenova P, Peñas A, de Rijke M, Sacaleanu B, Santos D, Sutcliffe RFE (2005) Overview of the clef 2005 multilingual question answering track. In: Peters C, Gey FC, Gonzalo J, Müller H, Jones GJF, Kluck M, Magnini B, de Rijke M (eds) CLEF, Springer, LNCS, vol 4022, pp 307–331

Vicedo JL (2001) Using semantics for paragraph selection in question answering systems. In: SPIRE, pp 220–227

Vogt CC, Cottrell GW (1999) Fusion via a linear combination of scores. Information Retrieval 1(3):151–173

Voorhees EM (2004) Overview of the TREC 2004 question answering track. In: Proceedings of the Thirteenth Text REtreival Conference (TREC 2004), pp 52–62

Witten IH, Frank E (1999) Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann

Xu J, Licuanan A, May J, Miller S, Weischedel R (2002) TREC 2002 QA at BBN: Answer selection and confidence estimation. In: TREC

Named Entity Recognition Using Appropriate Unlabeled Data, Post-processing and Voting

Asif Ekbal and Sivaji Bandyopadhyay
Department of Computer Science and Engineering
Jadavpur University
Kolkata, 700032, India
E-mail: asif.ekbal@gmail.com and sivaji_cse_ju@yahoo.com

Keywords: named entity recognition, maximum entropy, conditional random ield, support vector machine, weighted voting, Bengali

Received: January 10, 2009

This paper reports how the appropriate unlabeled data, post-processing and voting can be effective to improve the performance of a Named Entity Recognition (NER) system. The proposed method is based on a combination of the following classifiers: Maximum Entropy (ME), Conditional Random Field (CRF) and Support Vector Machine (SVM). The training set consists of approximately 272K wordforms. The proposed method is tested with Bengali. A semi-supervised learning technique has been developed that uses the unlabeled data during training of the system. We have shown that simply relying upon the use of large corpora during training for performance improvement is not in itself sufficient. We describe the measures to automatically select effective documents and sentences from the unlabeled data. In addition, we have used a number of techniques to post-process the output of each of the models in order to improve the performance. Finally, we have applied weighted voting approach to combine the models. Experimental results show the effectiveness of the proposed approach with the overall average recall, precision, and f-score values of 93.79%, 91.34%, and 92.55%, respectively, which shows an improvement of 19.4% in f-score over the least performing baseline ME based system and an improvement of 15.19% in f-score over the best performing baseline SVM based system.

Povzetek: Razvita je metoda za prepoznavanje imen, ki temelji na uteženem glasovanju več klasifikatorjev.

1 Introduction

Named Entity Recognition (NER) is an important tool in almost all Natural Language Processing (NLP) application areas such as Information Extraction [1], Machine Translation [2], Question Answering [3] etc. The objective of NER is to identify and classify every word/term in a document into some predefined categories like person name, location name, organization name, miscellaneous name (date, time, percentage and monetary expressions etc.) and "none-of-the-above". The challenge in detection of named entities (NEs) is that such expressions are hard to analyze using rule-based NLP because they belong to the open class of expressions, i.e., there is an infinite variety and new expressions are constantly being invented.

In recent years, automatic NER systems have become a popular research area in which a considerable number of studies have been addressed on developing these systems. These can be classified into three main classes [4], namely rule-based NER, machine learning-based NER and hybrid NER.

Rule-based approaches focus on extracting names using a number of hand-crafted rules. Generally, these systems consist of a set of patterns using grammatical (e.g., part of speech), syntactic (e.g., word precedence) and orthographic features (e.g., capitalization) in combination with dictionaries [5]. A NER system has been proposed in [6][7] based on carefully handcrafted regular expression called FASTUS. They divided the task into three steps: recognizing phrase, recognizing patterns and merging incidents, while [8] uses extensive specialized resources such as gazetteers, white and yellow pages. The NYU system [9] was introduced that uses handcrafted rules. A rule-based Greek NER system [10] has been developed in the context of the R&D project MITOS [1]. The NER system consists of three processing stages: linguistic pre-processing, NE identification and NE classification. The linguistic pre-processing stage involves some basic tasks: tokenisation, sentence splitting, part of speech (POS) tagging and stemming. Once the text has been annotated with POS tags, a stemmer is used. The aim of the stemmer is to reduce the size of the lexicon as well as the size and complexity of NER grammar. The NE identification phase involves the detection of their boundaries, i.e., the start and end of all the possible spans of tokens that are likely to belong to a NE. Classification involves three sub-stages: application of classification rules, gazetteer-based classification, and par-

[1]http://www.iit.demokritos.gr/skel/mitos

tial matching of classified NEs with unclassified ones. The French NER system has been implemented with the rule-based inference engine [11]. It is based on a large knowledge base including 8,000 proper names that share 10,000 forms and consists of 11,000 words. It has been used continuously since 1995 in several real-time document filtering applications [12]. Other rule-based NER systems are University Of Sheffield's LaSIE-II [13], ISOQuest's NetOwl [14] and University Of Edinburgh's LTG [15] [16] for English NER. These approaches are relying on manually coded rules and compiled corpora. These kinds of systems have better results for restricted domains and are capable of detecting complex entities that are difficult with learning models. However, rule-based systems lack the ability of portability and robustness, and furthermore the high cost of the maintenance of rules increases even though the data is slightly changed. These types of systems are often domain dependent, language specific and do not necessarily adapt well to new domains and languages.

Nowadays, machine-learning (ML) approaches are popularly used in NER because these are easily trainable, adaptable to different domains and languages as well as their maintenance are also less expensive [17]. On the other hand, rule-based approaches lack the ability of coping with the problems of robustness and portability. Each new source of text requires significant tweaking of rules to maintain optimal performance and the maintenance costs could be quite high. Some of the well-known machine-learning approaches used in NER are Hidden Markov Model (HMM)(BBN's IdentiFinder [18] [19]), Maximum Entropy (ME)(New York University's MENE in ([20]; [21]), Decision Tree (New York University's system in [22] and SRA's system in [23] and CRF [24]; [25]. Support Vector Machines (SVMs) based NER system was proposed by Yamada et al. [26] for Japanese. His system is an extension of Kudo's chunking system [27] that gave the best performance at CoNLL-2000 shared tasks. The other SVM-based NER systems can be found in [28] and [29].

Unsupervised learning method is another type of machine learning model, where an unsupervised model learns without any feedback. In unsupervised learning, the goal is to build representations from data. [30] discusses an unsupervised model for NE classification by the use of unlabeled examples of data. An unsupervised NE classification models and their ensembles have been introduced in [31] that uses a small-scale NE dictionary and an unlabeled corpus for classifying NEs. Unlike rule-based models, these types of models can be easily ported to different domains or languages.

In hybrid systems, the goal is to combine rule-based and machine learning-based methods, and develop new methods using strongest points from each method. [32] described a hybrid document centered system, called LTG system. [33] introduced a hybrid system by combining HMM, MaxEnt and handcrafted grammatical rules. Although, this approach can get better result than some other approaches, but weakness of handcraft rule-based NER surfaces when there is a need to change the domain of data. Previous works [34, 35] have also shown that combining several ML models using voting technique always performs better than any single ML model.

When applying machine-learning techniques to NLP tasks, it is time-consuming and expensive to hand-label the large amounts of training data necessary for good performance. In the literature, we can find the use of unlabeled data in improving the performance of many tasks such as name tagging [36], semantic class extraction [37] and co-reference resolution [38]. However, it is important to decide how the system should effectively select unlabeled data, and how the size and relevance of data impact the performance. A technique to automatically select documents is reported in [39].

India is a multilingual country with great cultural diversities. However, the relevant works in NER involving Indian languages have started to appear very recently. Named Entity (NE) identification in Indian languages in general and Bengali in particular is difficult and challenging as:

1. Unlike English and most of the European languages, Bengali lacks capitalization information, which plays a very important role in identifying NEs.

2. Indian person names are more diverse compared to the other languages and a lot of these words can be found in the dictionary with some other specific meanings.

3. Bengali is a highly inflectional language providing one of the richest and most challenging sets of linguistic and statistical features resulting in long and complex wordforms.

4. Bengali is a relatively free order language.

5. Bengali, like other Indian languages, is a resource poor language - annotated corpora, name dictionaries, good morphological analyzers, POS taggers etc. are not yet available in the required measure.

6. Although Indian languages have a very old and rich literary history, technological developments are of recent origin.

7. Web sources for name lists are available in English, but such lists are not available in Bengali forcing the use of transliteration for creating such lists.

A pattern directed shallow parsing approach for NER in Bengali has been reported in [40]. The paper reports about two different NER models, one using the lexical contextual patterns and the other using the linguistic features along with the same set of lexical contextual patterns. A HMM-based NER system has been reported in [41], where more contextual information has been considered during the emission probabilities and NE suffixes have been kept for handling the unknown words. More recently, the works in the area of Bengali NER can be found in [42] with ME, in [43] with CRF and in [44] with SVM approach. These systems were developed with the help of a number of features and gazetteers. The method of improving the performance of NER system using appropriate unlabeled data, post-processing and voting has been reported in [45].

Other than Bengali, the works on Hindi can be found in [46] with CRF model using feature induction technique to automatically construct the features that does a maximal increase in the conditional likelihood. A language independent method for Hindi NER has been reported in [47]. Sujan et al. [48] reported a ME based system with the hybrid feature set that includes statistical as well as linguistic features. A MEMM-based system has been reported in [49]. As part of the IJCNLP-08 NER shared task, various works of NER in Indian languages using various approaches can be found in IJCNLP-08 NER Shared Task on South and South East Asian Languages (NERSSEAL)[2]. As part of this shared task, [50] reported a CRF-based system followed by post-processing which involves using some heuristics or rules. A CRF-based system has been reported in [51], where it has been shown that the hybrid HMM model can perform better than CRF.

Srikanth and Murthy [52] developed a NER system for Telugu and tested it on several data sets from the Eenaadu and Andhra Prabha newspaper corpora. They obtained the overall f-measure between 80-97% with person, location and organization tags. For Tamil, a CRF-based NER system has been presented in [53] for the tourism domain. This approach can take care of morphological inflections of NEs and can handle nested tagging with a hierarchical tagset containing 106 tags. Shishtla et al. [54] developed a CRF-based system for English, Telugu and Hindi. They suggested that character n-gram based approach is more effective than the word based models. They described the features used and the experiments to increase the recall of NER system.

In this paper, we have reported a NER system for Bengali by combining the outputs of the classifiers, namely ME, CRF and SVM. In terms of native speakers, Bengali is the seventh most spoken language in the world, second in India and the national language of Bangladesh. We have manually annotated a portion of the Bengali news corpus, developed from the web-archive of a leading Bengali newspaper with *Person name*, *Location name*, *Organization name* and *Miscellaneous name* tags. We have also used the IJCNLP-08 NER Shared Task data that was originally annotated with a fine-grained NE tagset of twelve tags. This data has been converted into the forms to be tagged with NEP (Person name), NEL (Location name), NEO (Organization name), NEN (Number expressions), NETI (Time expressions) and NEM (Measurement expressions). The NEN, NETI and NEM tags are mapped to point to the miscellaneous entities. The system makes use of the different contextual information of the words along with the variety of orthographic word level features that are helpful in predicting the various NE classes. We have considered both language independent as well as language dependent features. Language independent features are applicable to almost all the languages including Bengali and Hindi. Language dependent features have been extracted from the language specific resources such as the part of speech (POS) taggers and gazetteers. It has been observed from the evaluation results that the use of language specific features improves the performance of the system. We also conducted a number of experiments to find out the best-suited set of features for NER in each of the languages. We have developed an unsupervised method to generate the lexical context patterns that are used as the features of the classifiers. A semi-supervised technique has been proposed to select the appropriate unlabeled documents from a large collection of unlabeled corpus. The main contribution of this work is as follows:

1. An unsupervised technique has been reported to generate the context patterns from the unlabeled corpus.
2. A semi-supervised ML technique has been developed in order to use the unlabeled data.
3. Relevant unlabeled documents are selected using CRF techniques. We have selected effective sentences to be added to the initial labeled data by applying majority voting between ME model, CRF and two different models of SVM. In the previous literature [39], the use of any single classifier was reported for selecting appropriate sentences.
4. Useful features for NER in Bengali are identified. A number of features are language independent and can be applicable to other languages also.
5. The system has been evaluated in two different ways: Without language dependent features and with language dependent features.
6. Three different post-processing techniques have been reported in order to improve the performance of the classifiers.
7. Finally, models are combined using three weighted voting techniques.

2 Named entity tagged corpus development

The rapid development of language resources and tools using machine learning techniques for less computerized languages requires appropriately tagged corpus. There is a long history of creating a standard for western language resources. The human language technology (HLT) society in Europe has been particularly zealous for the standardization of European languages. On the other hand, in spite of having great linguistic and cultural diversity, Asian language resources have received much less attention than their western counterparts. India is a multilingual country with a diverse cultural heritage. Bengali is one of the most

[2]http://ltrc.iiit.ac.in/ner-ssea-08

popular languages and predominantly spoken in the eastern part of India. In terms of native speakers, Bengali is the seventh most spoken language in the World, second in India and the national language in Bangladesh. In the literature, there has been no initiative of corpus development from the web in Indian languages and specifically in Bengali.

Newspaper is a huge source of readily available documents. Web is a great source of language data. In Bengali, there are some newspapers (like, Anandabazar Patrika, Bartaman, Dainik, Ittefaq etc.), published from Kolkata and Bangladesh, which have their internet-edition in the web and some of them provide their archive available also. A collection of documents from the archive of the newspaper, stored in the web, may be used as the corpus, which in turn can be used in many NLP applications.

We have followed the method of developing the Bengali news corpus in terms of language resource acquisition using a web crawler, language resource creation that includes HTML file cleaning, code conversion and language resource annotation that involves defining a tagset and subsequent tagging of the news corpus. A web crawler has been designed that retrieves the web pages in Hyper Text Markup Language (HTML) format from the news archive. Various types of news (International, National, State, Sports, Business etc.) are collected in the corpus and so a variety of linguistics features of Bengali are covered. The Bengali news corpus is available in UTF-8 and contains approximately 34 million wordforms.

A news corpus, whether in Bengali or in any other language has different parts like title, date, reporter, location, body etc. To identify these parts in a news corpus the tagset described in Table 1 have been defined. Detailed of this corpus development work can be found in [55].

The *date*, *location*, *reporter* and *agency* tags present in the web pages of the Bengali news corpus have been automatically named entity (NE) tagged. These tags can identify the NEs that appear in some fixed places of the newspaper. In order to achieve reasonable performance for NER, supervised machine learning approaches are more appropriate and this requires a completely tagged corpus. This requires the selection of an appropriate NE tagset.

With respect to the tagset, the main feature that concerns us is its granularity, which is directly related to the size of the tagset. If the tagset is too coarse, the tagging accuracy will be much higher, since only the important distinctions are considered, and the classification may be easier both by human manual annotators as well as the machine. But, some important information may be missed out due to the coarse grained tagset. On the other hand, a too fine-grained tagset may enrich the supplied information but the performance of the automatic named entity tagger may decrease. A much richer model is required to be designed to capture the encoded information when using a fine grained tagset and hence, it is more difficult to learn.

When we are about to design a tagset for the NE disambiguation task, the issues that need consideration include - the type of applications (some application may required more complex information whereas only category information may be sufficient for some tasks), tagging techniques to be used (statistical, rule based which can adopt large tagsets very well, supervised/unsupervised learning). Further, a large amount of annotated corpus is usually required for statistical named entity taggers. A too fine grained tagset might be difficult to use by human annotators during the development of a large annotated corpus. Hence, the availability of resources needs to be considered during the design of a tagset.

Table 2: Statistics of the NE tagged corpus

Total Number of sentences	23,181
Number of wordforms (approx.)	200K
Number of NEs	19,749
Average length of NE	2 (approx.)

During the design of the tagset for Bengali, our main aim was to build a small but clean and completely tagged corpora for Bengali. The resources can be used for the conventional usages like Information Retrieval, Information Extraction, Event Tracking System, Web People Search etc. We have used CoNLL 2003 shared task tagset as reference point for our tagset design.

We have used a NE tagset that consists of the following four tags:

1. *Person name*: Denotes the names of people. For example, *sachin*[Sachin] /*Person name*, *manmohan singh*[Manmohan Singh]/*Person name*.

2. *Location name*: Denotes the names of places. For example, *jadavpur*[Jadavpur]/*Location name*, *new delhi*[New Delhi]/*Location name*.

3. *Organization name*: Denotes the names of organizations. For example, *infosys*[Infosys]/*Organization name*, *jadavpur vishwavidyalaya*[Jadavpur University]/*Organization name*.

4. *Miscellaneous name*: Denotes the miscellaneous NEs that include date, time, number, monetary expressions, measurement expressions and percentages. For example, *15th august 1947*[15th August 1947]/*Miscellaneous name*, *11 am*[11 am]/*Miscellaneous name*, 110/*Miscellaneous name*, *1000 taka*[1000 rupees]/*Miscellaneous name*, *100%*[100%]/ *Miscellaneous name* and *100 gram*[100 gram]/ *Miscellaneous name*.

We have manually annotated approximately 200K wordforms of the Bengali news corpus.The annotation has been carried out by one expert and edited by another expert. The corpus is in the Shakti Standard Format (SSF) form [56]. Some statistics of this corpus is shown in Table 2.

We have also used the NE tagged corpus of the IJCNLP Shared Task on Named Entity Recognition for South

Table 1: News corpus tag set

Tag	Definition	Tag	Definition
header	Header of the news document	reporter	Reporter-name
title	Headline of the news document	agency	Agency providing news
t1	1st headline of the title	location	The news location
t2	2nd headline of the title	body	Body of the news document
date	Date of the news document	p	Paragraph
bd	Bengali date	table	Information in tabular form
day	Day	tc	Table Column
ed	English date	tr	Table row

Table 3: Statistics of the IJCNLP-08 NE tagged corpus

Total Number of sentences	7035
Number of wordforms (approx.)	122K
Number of NEs	5921
Average length of NE	2 (approx.)

and South East Asian Languages (NERSSEAL)[3]. A fine grained tagset of twelve tags were defined as part of this shared task. The underlying reason to adopt this finer NE tagset is to use the NER system in various NLP applications, particularly in machine translation. The IJCNLP-08 NER shared task tagset is shown in Table 4. One important aspect of the shared task was to identify and classify the maximal NEs as well as the nested NEs, i.e, the constituent part of a larger NE. But, the training data were provided with the type of the maximal NE only. For example, *mahatma gandhi road* (Mahatma Gandhi Road) was annotated as location and assigned the tag 'NEL' even if *mahatma* (Mahatma) and *gandhi*(Gandhi) are NE title person (NETP), and person name (NEP), respectively. The task was to identify *mahatma gandhi road* as a NE and classify it as NEL. In addition, *mahatma*, and *gandhi* were to be recognized as NEs of the categories NETP (Title person) and NEP (Person name) respectively. Some NE tags are hard to distinguish in some contexts. For example, it is not always clear whether something should be marked as 'Number' or as 'Measure'. Similarly, 'Time' and 'Measure' is another confusing pair of NE tags. Another difficult class is 'Technical terms' and it is often confusing whether any expression is to be tagged as the 'NETE' (NE term expression) or not. For example, it is difficult to decide whether 'Agriculture' is 'NETE', and if not then whether 'Horticulture' is 'NETE' or not. In fact, this the most difficult class to identify. Other ambiguous tags are 'NETE' and 'NETO' (NE title-objects). The corpus is in the Shakti Standard Format (SSF) form [56]. We have also manually annotated a portion of the Bengali news corpus [55] with the twelve NE tags of the shared task tagset. Some statistics of this corpus is shown in Table 3.

We have considered only those NE tags that denote person name, location name, organization name, number expression, time expression and measurement expressions. The number, time and measurement expressions are mapped to belong to the *Miscellaneous name* tag. Other tags of the shared task have been mapped to the 'other-than-NE' category. Hence, the final tagset is shown in Table 5.

In order to properly denote the boundaries of the NEs, the four NE tags are further subdivided as shown in Table 6. In the output, these sixteen NE tags are directly mapped to the four major NE tags, namely *Person name*, *Location name*, *Organization name* and *Miscellaneous name*.

[3]http://ltrc.iiit.ac.in/ner-ssea-08

3 Named entity recognition in Bengali

In terms of native speakers, Bengali is the seventh popular language in the world, second in India and the national language of Bangladesh. We have used a Bengali news corpus [55], developed from the web-archive of a widely read Bengali newspaper for NER. A portion of this corpus containing 200K wordforms has been manually annotated with the four NE tags namely, *Person name*, *Location name*, *Organization name* and *Miscellaneous name*. The data has been collected from the International, National, State and Sports domains. We have also used the annotated corpus of 122K wordforms, collected from the IJCNLP-08 NERSSEAL (http://ltrc.iiit.ac.in/ner-ssea-08). This data was a mixed one and dealt mainly with the literature, agriculture and scientific domains. Moreover, this data was originally annotated with a fine-grained NE tagset of twelve tags. An appropriate tag conversion routine has been defined as shown in Table 5 in order to convert this data into the desired forms, tagged with the four NE tags.

3.1 Approaches

NLP research around the world has taken giant leaps in the last decade with the advent of effective machine learning algorithms and the creation of large annotated corpora for various languages. However, annotated corpora and other lexical resources have started appearing only very recently in India. In this paper, we have reported a NER system by combining the outputs of the classifiers, namely ME, CRF

Table 4: Named entity tagset for Indian languages (IJCNLP-08 NER Shared Task Tagset)

NE Tag	Meaning	Example
NEP	Person name	*sachin*/NEP, *sachin ramesh tendulkar* / NEP
NEL	Location name	*kolkata*/NEL, *mahatma gandhi road*/ NEL
NEO	Organization name	*jadavpur bishbidyalya*/NEO, *bhaba eytomik risarch sentar* / NEO
NED	Designation	*chairrman*/NED, *sangsad*/NED
NEA	Abbreviation	*b a*/NEA, *c m d a*/NEA, *b j p*/NEA, *i.b.m*/ NEA
NEB	Brand	*fanta*/NEB
NETP	Title-person	*shriman*/NED, *shri*/NED, *shrimati*/NED
NETO	Title-object	*american beauty*/NETO
NEN	Number	*10*/NEN, *dash*/NEN
NEM	Measure	*tin din*/NEM, *panch keji*/NEM
NETE	Terms	*hiden markov model*/NETE, *chemical reaction*/NETE
NETI	Time	*10 i magh 1402* / NETI, *10 am*/NETI

Table 5: Tagset used in this work

IJCNLP-08 shared task tagset	Tagset used	Meaning
NEP	*Person name*	Single word/multiword person name
NEL	*Location name*	Single word/multiword location name
NEO	*Organization name*	Single word/multiword organization name
NEN, NEM, NETI	*Miscellaneous name*	Single word/ multiword miscellaneous name
NED, NEA, NEB, NETP, NETE	NNE	Other than NEs

Table 6: Named entity tagset (B-I-E format)

Named Entity Tag	Meaning	Example
PER	Single word person name	*sachin*/PER, *rabindranath*/PER
LOC	Single word location name	*kolkata*/LOC, *mumbai*/LOC
ORG	Single word organization name	*infosys*/ORG
MISC	Single word miscellaneous name	*10*/MISC, *dash*/MISC
B-PER I-PER E-PER	Beginning, Internal or the End of a multiword person name	*sachin*/B-PER *ramesh*/I-PER *tendulkar* /E-PER, *rabindranath*/B-PER *thakur*/E-PER
B-LOC I-LOC E-LOC	Beginning, Internal or the End of a multiword location name	*mahatma*/B-LOC *gandhi* /I-LOC *road* /E-LOC, *new*/B-LOC *york*/E-LOC
B-ORG I-ORG E-ORG	Beginning, Internal or the End of a multiword organization name	*jadavpur* /B-ORG *bishvidyalya*/E-ORG, *bhaba* /B-ORG *eytomik*/I-ORG *risarch*/I-ORG *sentar* /E-ORG
B-MISC I-MISC E-MISC	Beginning, Internal or the End of a multiword miscellaneous name	*10 i* /B-MISC *magh*/I-MISC *1402*/E-MISC, *10*/B-MISC *am*/E-MISC
NNE	Other than NEs	*kara*/NNE, *jal*/NNE

and SVM frameworks in order to identify NEs from a Bengali text and to classify them into *Person name*, *Location name*, *Organization name* and *Miscellaneous name*. We have developed two different systems with the SVM model, one using **forward parsing** (SVM-F) that parses from left to right and other using **backward parsing** (SVM-B) that parses from right to left. The SVM system has been developed based on [57], which perform classification by constructing a N-dimensional hyperplane that optimally separates data into two categories. We have used *YamCha* toolkit (http://chasen-org/∼taku/software/yamcha), an SVM based tool for detecting classes in documents and formulating the NER task as a sequence labeling problem. Here, the pair wise multi-class decision method and *polynomial kernel function* have been used. We have used TinySVM-0.0[4] TinySVM classifier that seems to be the best optimized among publicly available SVM toolkits. We have used the Maximum Entropy package (http://homepages.inf.ed.ac.uk/s0450736/software/ maxent/maxent-20061005.tar.bz2). We have used C++ based CRF++ package (http://crfpp.sourceforge.net) for NER.

During testing, it is possible that the classifier produces a sequence of inadmissible classes (e.g., B-PER followed by LOC). To eliminate such sequences, we define a transition probability between word classes $P(c_i|c_j)$ to be equal to 1 if the sequence is admissible, and 0 otherwise. The probability of the classes $c_1, c_2, \dots, c_n$ assigned to the words in a sentence 's' in a document 'D' is defined as follows: $P(c_1, c_2, \dots, c_n|S, D) = \prod_{i=1}^{n} P(c_1|S, D) \times P(c_i|c_{i-1})$ where $P(c_1|S, D)$ is determined by the ME/CRF/SVM classifier.

Performance of the NER models has been limited in part by the amount of labeled training data available. We have used unlabeled corpus to address this problem. Based on the original training on the labeled corpus, there will be some tags in the unlabeled corpus that the taggers will be very sure about. For example, there will be contexts that were always followed by a person name (*sri*, *mr.* etc.) in the training corpus. While a new word W is found in this context in the unlabeled corpus then it can be predicted as a person name. If any tagger can learn this fact about W, it can successfully tag W when it appears in the test corpus without any indicative context. In the similar way, if a previously unseen context appears consistently in the unlabeled corpus before known NE then the tagger should learn that this is a predicative context. We have developed a semi-supervised learning approach in order to capture this information that are used as the features in the classifiers. We have used another semi-supervised learning approach in order to select appropriate data from the available large unlabeled corpora and added to the initial training set in order to improve the performance of the taggers. The models are retrained with this new training set and this process is repeated in a bootstrapped manner.

[4]http://cl.aist-nara.ac.jp/∼taku ku/software/

We have also used a number of post-processing rules in order to improve the performance in each of the models. Finally, three models are combined together into a single system with the help of three weighted voting schemes.

In the following subsections, some of our earlier attempts in NER have been reported that form the base of our overall approach in NER.

3.1.1 Pattern directed shallow parsing approach

Two NER models, namely A and B, using a pattern directed shallow parsing approach have been reported in [40]. An unsupervised algorithm has been developed that tags the unlabeled corpus with the seed entities of *Person name*, *Location name* and *Organization name*. These seeds have been prepared by automatically extracting the words from the *reporter*, *location* and *agency* tags of the Bengali news corpus [55]. Model A uses only the seed lists to tag the training corpus whereas in model B, we have used the various gazetteers along with the seed entities for tagging. The lexical context patterns generated in such way are used to generate further patterns in a bootstrapped manner. The algorithm terminates until no new patterns can be generated. During testing, model A can not deal with the *NE classification disambiguation* problem (i.e, can not handle the situation when a particular word is tagged with more than one NE type) but model B can handle with this problem with the help of gazetteers and various language dependent features.

3.1.2 HMM based NER system

A HMM-based NER system has been reported in [41], where more context information has been considered during emission probabilities and the word suffixes have been used for handling the unknown words. A brief description of the system is given below:

In the HMM based NE tagging, the task is to find the sequence of NE tags $T = t_1, t_2, t_3, \ldots t_n$ that is optimal for a word sequence $W = w_1, w_2, w_3 \ldots w_n$. The tagging problem becomes equivalent to searching for $argmax_T P(T) * P(W|T)$, by the application of Bayes' law.

A trigram model has been used for transition probability, that is, the probability of a tag depends on two previous tags, and then we have,

$P(T) = P(t_1|\$) \times P(t_2|\$, t_1) \times P(t_3|t_1, t_2) \times P(t_4|t_2, t_3) \times \ldots \times P(t_n|t_{n-2}, t_{n-1})$

where an additional tag '\$' (dummy tag) has been introduced to represent the beginning of a sentence. Due to sparse data problem, the linear interpolation method has been used to smooth the trigram probabilities as follows:

$P'(t_n|t_{n-2}, t_{n-1}) = \lambda_1 P(t_n) + \lambda_2 P(t_n|t_{n-1}) + \lambda_3 P(t_n|t_{n-2}, t_{n-1})$

such that the λs sum to 1. The values of λs have been calculated by the method given in [58].

Additional context dependent feature has been introduced to the emission probability to make the Markov model more powerful. The probability of the current word depends on the tag of the previous word and the tag to be assigned to the current word. Now, we calculate $P(W|T)$ by the following equation:

$P(W|T) \approx P(w_1|\$, t_1) \times P(w_2|t_1, t_2) \times \ldots \times P(w_n|t_{n-1}, t_n)$.

So, the emission probability can be calculated as:

$P(w_i|t_{i-1}, t_i) = \frac{freq(t_{i-1}, t_i, w_i)}{freq(t_{i-1}, t_i)}$

Here, also the smoothing technique is applied rather than using the emission probability directly. The emission probability is calculated as:

$P'(w_i|t_{i-1}, t_i) = \theta_1 P(w_i|t_i) + \theta_2 P(w_i|t_{i-1}, t_i)$,

where θ_1, θ_2 are two constants such that all θs sum to 1. In general, the values of θs can be calculated by the same method that was adopted in calculating λs.

Handling of unknown words is an important problem in the HMM based NER system. For words which have not been seen in the training set, $P(w_i|t_i)$ is estimated based on features of the unknown words, such as whether the word contains a particular suffix. The list of suffixes has been prepared that usually appear at the end of NEs. A null suffix is also kept to take care of those words that have none of the suffixes in the list. The probability distribution of a particular suffix with respect to specific NE tag is generated from all words in the training set that share the same suffix.

Incorporating diverse features in an HMM based NE tagger is difficult and complicates the smoothing typically used in such taggers. Indian languages are morphologically very rich and contains a lot of non-independent features. A ME [20] or CRF [25] or SVM [26] based method can deal with the diverse and overlapping features of the Indian languages more efficiently than HMM.

3.1.3 Other NER sytems

A ME based NER system for Bengali has been reported in [42]. The system has been developed with the contextual information of the words along with the variety of orthographic word-level features. In addition, a number of manually developed gazetteers have been used as the features in the model. We conducted a number of experiments in order to find out the appropriate features for NER in Bengali. Detailed evaluation results have shown the best performance with a contextual word window of size three, i.e., previous word, current word and the next one word, dynamic NE tag of the previous word, POS tag of the current word, prefixes and suffixes of length up to three characters of the current word and binary valued features extracted from the gazetteers.

A CRF based NER system has been described in [43]. The system has been developed with the same set of features as that of ME. Evaluation results have demonstrated the best results with a contextual window of size five, i.e, previous two words, current word and next two words, NE tag of the previous word, POS tags of the current and the previous words, suffixes and prefixes of length up to three characters of the current word, and the various binary valued features extracted from the several gazetteers.

A SVM based NER system has been described in [44]. This model also makes use of the different contextual information of the words, orthographic word-level features along with the various gazetteers. Results have demonstrated the best results with a contextual window of size six, i.e., previous three words, current word and next two words, NE tag of the previous two words, POS tags of the current, previous word and the next words, suffixes and prefixes of length of length up to three characters of the current word, and the various binary valued features extracted from the several gazetteers.

4 Named entity features

Feature selection plays a crucial role in any statistical model. ME model does not provide a method for automatic selection of given feature sets. Usually, heuristics are used for selecting effective features and their combinations. It is not possible to add arbitrary features in a ME framework as that will result in overfitting. Unlike ME, CRF does not require careful feature selection in order to avoid overfitting. CRF has the freedom to include arbitrary features, and the ability of feature induction to automatically construct the most useful feature combinations. Since, CRFs are log-linear models, and high accuracy may require complex decision boundaries that are non-linear in the space of original features, the expressive power of the models is often increased by adding new features that are conjunctions of the original features. For example, a conjunction feature might ask if the current word is in the person name list and the next word is an action verb '*ballen*'(told). One could create arbitrary complicated features with these conjunctions. However, it is infeasible to incorporate all possible conjunctions as these might result in overflow of memory as well as overfitting. Support vector machines predict the classes depending upon the labeled word examples only. It predicts the NEs based on feature information of words collected in a predefined window size while ME or CRF predicts them based on the information of the whole sentence. So, CRF can handle the NEs with outside tokens, which SVM always tags as 'NNE'. A CRF has different characteristics from SVM, and is good at handling different kinds of data. In particular, SVMs achieve high generalization even with training data of a very high dimension. Moreover, with the use of *kernel function*, SVMs can handle non-linear feature spaces, and carry out the training considering combinations of more than one feature.

The main features for the NER task have been identified based on the different possible combination of available word and tag context. The features also include prefix and suffix for all words. The term prefix/suffix is a sequence of first/last few characters of a word, which may not be a linguistically meaningful prefix/suffix. The use of prefix/suffix information works well for the highly inflected languages as like the Indian languages. In addition to these, various gazetteer lists have been developed for use in the NER tasks. We have considered different combination from the following set for inspecting the best set of features for NER in Bengali:

F={$w_{i-m}, \ldots, w_{i-1}, w_i, w_{i+1}, \ldots w_{i+n}$, |prefix| $\leq n$, |suffix| $\leq n$, NE tag(s) of previous word(s), POS tag(s) of the current and/or the surrounding word(s), First word, Length of the word, Digit information, Infrequent word, Gazetteer lists}, where w_i is the current word; w_{i-m} is the previous *m*th word and w_{i+n} is the next *n*th word.

The set 'F' contains both language independent as well as language dependent features. The set of language independent features includes the context words, prefixes and suffixes of all the words, NE information of the previous word(s), first word, length of the word, digit information and infrequent word. Language dependent features for Bengali include the set of known suffixes that may appear with the various NEs, clue words that help in predicting the location and organization names, words that help to recognize measurement expressions, designation words that help to identify person names, various gazetteer lists that include the first names, middle names, last names, location names, organization names, function words, weekdays and month names. As part of language dependent features for Hindi, the system uses only the lists of first names, middle names, last names, weekdays, month names along with the list of words that helps to recognize measurement expressions. We have also used the part of speech (POS) information of the current and/or the surrounding word(s) for Bengali.

Language independent NE features can be applied for NER in any language without any prior knowledge of that language. Though the lists or gazetteers are not theoretically language dependent, we call it as language dependent as these require apriori knowledge of any specific language for their preparation. Also, we include the POS information in the set of language dependent features as it depends on some language specific phenomenon such as person, number, tense, gender etc. For example, gender information has a crucial role in Hindi but it is not an issue in Bengali. In Bengali, a combination of non-finite verb followed by a finite verb can have several different morphosyntactic functions. For example, '*mere phellO*' [kill+non-finite throw+finite] can mean 'threw after killing' (here, '*mere*' is a sequential participle) or just 'killed' with a completive sense (where, '*mere*' is a polar verb and '*phellO*', the vector verb of a finite verb group). On the other hand, constructs like '*henshe ballO*' [smile+non-finite say+finite] might mean 'said while smiling' ('*henshe*' is functioning as an adverbial participle). Similarly, it is hard to distinguish between the adjectival participle and verbal nouns. The use of language specific features is helpful to improve the performance of the NER system. In the resource-constrained Indian language environment, the non-availability of language specific resources such as POS taggers, gazetteers, morphological analyzers etc. forces the development of such resources to use in NER systems. This leads to the necessity of apriori knowledge of the language.

4.1 Language independent features

We have considered different combinations from the set of language independent features for inspecting the best set of features for NER in Bengali. Following are the details of the features:

- Context word feature: Preceding and following words of a particular word can be used as the features. This is based on the observation that the surrounding words are very effective in the identification of NEs.

- Word suffix: Word suffix information is helpful to identify NEs. This is based on the observation that the NEs share some common suffixes. This feature can be used in two different ways. The first and the naïve one is, a fixed length (say, n) word suffix of the current and/or the surrounding word(s) can be treated as feature. If the length of the corresponding word is less than or equal to $n - 1$ then the feature values are not defined and denoted by ND. The feature value is also not defined (ND) if the token itself is a punctuation symbol or contains any special symbol or digit. The value of ND is set to 0. The second and the more helpful approach is to modify the feature as binary valued. Variable length suffixes of a word can be matched with predefined lists of useful suffixes for different classes of NEs. Various length suffixes belong to the category of language dependent features as they require language specific knowledge for their development.

- Word prefix: Word prefixes are also helpful and based on the observation that NEs share some common prefix strings. This feature has been defined in a similar way as that of the fixed length suffixes.

- Named Entity Information: The NE tag(s) of the previous word(s) has been used as the only dynamic feature in the experiment.

- First word: This is used to check whether the current token is the first word of the sentence or not. Though Bengali is a relatively free order language, the first word of the sentence is most likely a NE as it appears in the subject position most of the time.

- Digit features: Several binary valued digit features have been defined depending upon the presence and/or the number of digits in a token (e.g., CntDgt [token contains digits], FourDgt [four digit token], TwoDgt [two digit token]), combination of digits and punctuation symbols (e.g., CntDgtCma [token consists of digits and comma], CntDgtPrd [token consists of digits and periods]), combination of digits and symbols (e.g., CntDgtSlsh [token consists of digit and slash], CntDgtHph [token consists of digits and hyphen], CntDgtPrctg [token consists of digits and percentages]). These binary valued features are helpful in recognizing miscellaneous NEs, such as time expressions, measurement expressions and numerical numbers etc.

- Infrequent word: The frequencies of the words in the training corpus have been calculated. A cut off frequency has been chosen in order to consider the words that occur with more than the cut off frequency in the training corpus. The cut off frequency is set to 10. A binary valued feature 'Infrequent' is defined to check whether the current token appears in this list or not.

- Length of a word: This binary valued feature is used to check whether the length of the current word is less than three or not. This is based on the observation that very short words are rarely NEs.

The above set of language independent features along with their descriptions are shown in Table 7. The *baseline* models have been developed with the language independent features.

4.2 Language dependent features

Language dependent features for Bengali have been identified based on the earlier experiments [40] on NER. Additional NE features have been identified from the Bengali news corpus [55]. Various gazetteers used in the experiment are presented in Table 8. Some of the gazetteers are briefly described as below:

- NE Suffix list (variable length suffixes): Variable length suffixes of a word are matched with the predefined lists of useful suffixes that are helpful to detect person (e.g., *-babu*, *-da*, *-di* etc.) and location (e.g., *-land*, *-pur*, *-liya* etc.) names.

- Organization suffix word list: This list contains the words that are helpful to identify organization names (e.g., *kong*, *limited* etc.). These are also part of organization names.

- Person prefix word list: This is useful for detecting person names (e.g., *shriman*, *shri*, *shrimati* etc.).

- Common location word list: This list contains the words (e.g., *sarani*, *road*, *lane* etc.) that are part of the multiword location names and usually appear at their end.

- Action verb list: A set of action verbs like *balen*, *balalen*, *ballo*, *sunllO*, *hanslo* etc. often determine the presence of person names. Person names generally appear before the action verbs.

- Designation words: A list of common designation words (e.g., *neta*, *sangsad*, *kheloar* etc.) has been prepared. This helps to identify the position of person names.

Table 7: Descriptions of the language independent features. Here, *i* represents the position of the current word and w_i represents the current word

Feature	Description
ContexT	ContexT$_i = w_{i-m}, \ldots, w_{i-1}, w_i, w_{i+1}, w_{i+n}$, where w_{i-m}, and w_{i+n} are the previous m-th, and the next n-th word
Suf	$\text{Suf}_i(n) = \begin{cases} \text{Suffix string of length } n \text{ of } w_i & \text{if } \lvert w_i\rvert \geq n \\ ND(=0) \text{ if } \lvert w_i\rvert \leq (n-1) \\ \quad \text{or } w_i \text{ is a punctuation symbol} \\ \quad \text{or } w_i \text{ contains any special symbol or digit} \end{cases}$
Pre	$\text{Pre}_i(n) = \begin{cases} \text{Prefix string of length } n \text{ of } w_i & \text{if } \lvert w_i\rvert \geq n \\ ND(=0) \text{ if } \lvert w_i\rvert \leq (n-1) \\ \quad \text{or } w_i \text{ is a punctuation symbol} \\ \quad \text{or } w_i \text{ contains any special symbol or digit} \end{cases}$
NE	$NE_i =$ NE tag of w_i
FirstWord	$\text{FirstWord}_i = \begin{cases} 1, \text{if } w_i \text{ is the first word of a sentence} \\ 0, \text{otherwise} \end{cases}$
CntDgt	$\text{CntDgt}_i = \begin{cases} 1, \text{if } w_i \text{ contains digit} \\ 0, \text{otherwise} \end{cases}$
FourDgt	$\text{FourDgt}_i = \begin{cases} 1, \text{if } w_i \text{ consists of four digits} \\ 0, \text{otherwise} \end{cases}$
TwoDgt	$\text{TwoDgt}_i = \begin{cases} 1, \text{if } w_i \text{ consists of two digits} \\ 0, \text{otherwise} \end{cases}$
CntDgtCma	$\text{CntDgtCma}_i = \begin{cases} 1, \text{if } w_i \text{ contains digit and comma} \\ 0, \text{otherwise} \end{cases}$
CntDgtPrd	$\text{CntDgtPrd}_i = \begin{cases} 1, \text{if } w_i \text{ contains digit and period} \\ 0, \text{otherwise} \end{cases}$
CntDgtSlsh	$\text{CntDgtSlsh}_i = \begin{cases} 1, \text{if } w_i \text{ contains digit and slash} \\ 0, \text{otherwise} \end{cases}$
CntDgtHph	$\text{CntDgtHph}_i = \begin{cases} 1, \text{if } w_i \text{ contains digit and hyphen} \\ 0, \text{otherwise} \end{cases}$
CntDgtPrctg	$\text{CntDgtPrctg}_i = \begin{cases} 1, \text{if } w_i \text{ contains digit} \\ \quad \text{and percentage} \\ 0, \text{otherwise} \end{cases}$
Infrequent	$\text{Infrequent}_i = I_{\{\text{Infrequent word list}\}}(w_i)$
Length	$\text{Length}_i = \begin{cases} 1, \text{if } w_i \geq 3 \\ 0, \text{otherwise} \end{cases}$

– Part of Speech information: For POS tagging, we have used a CRF-based POS tagger [59], which has been developed with the help of a tagset of 26 different POS tags [5], defined for the Indian languages. We have used the inflection lists that can appear with the different wordforms of noun, verb and adjectives, a lexicon [60] that has been developed in an unsupervised way from the Bengali news corpus, and the NE tags using a NER system [44] as the features of POS tagging in Bengali. This POS tagger has an accuracy of 90.2%.

The language dependent features are represented in Table 9.

5 Use of unlabeled data

We have developed two different techniques that use the large collection of unlabeled corpus [55] in NER. The first one is an unsupervised learning technique used to generate lexical context patterns for use as the features of the classifiers. The second one is a semi-supervised learning technique that is used to select the appropriate data from the large collection of documents. In the literature, unsupervised algorithms (bootstrapping from seed examples and unlabeled data) have been discussed in [61], [47], and [62]. Using a parsed corpus, the proper names that appear in certain syntactic contents were identified and classified in [61].The procedures to identify and classify proper names in seven languages, learning character-based contextual, internal, and morphological patterns are reported in [62]. This algorithm does not strictly require capitalization but recall was much lower for the languages that do not have case distinctions. Others such as [63] relied on structures such as appositives and compound nouns. Contextual patterns that predict the semantic class of the subject, direct object, or prepositional phrase object are reported in [64] and [65]. The technique to use the windows of tokens to learn contextual and internal patterns without parsing is described in [66] and [67]. The technique reported in [67] enable discovery of generalized names embedded in larger noun groups. An algorithm for unsupervised learning and semantic classification of names and terms is reported in [67]. They considered the *positive example* and *negative example* for a particular name class. We have developed an unsupervised algorithm that can generate the lexical context patterns from the unlabeled corpus. This work differs from the previous works in the sense that here we have also considered the patterns that yield *negative examples*. These *negative examples* can be effective to generate new patterns. Apart from *accuracy*, we have considered the *relative frequency* of a pattern in order to decide its inclusion into the final set of patterns. The final lexical context patterns have been used as features of the classifiers. Here, we have used a portion of the Bengali news corpus [55] that has been classified on geographic domain (International, National, State, District, Metro [Kolkata]) as well as on topic domain (Politics, Sports, Business). Statistics of this corpus is shown in Table 10.

[5]http://shiva.iiit.ac.in/SPSAL2007/iiit_tagset_guidelines.pdf

5.1 Lexical context pattern learning

Lexical context patterns are generated from the unlabeled corpus of approximately 10 million wordforms, as shown in Table 10. Given a small seed examples and an unlabeled corpus, the algorithm can generate the lexical context patterns through bootstrapping. The seed name serves as a *positive example* for its own NE class, *negative example* for other NE classes and *error example* for non-NEs.

1. Seed list preparation: We have collected frequently occurring words from the Bengali news corpus and the annotated training set of 272K wordforms to use as the seeds. There are 123, 87, and 32 entries in the person, location, and organization seed lists, respectively.

2. Lexical pattern generation: The unlabeled corpus is tagged with the elements from the seed lists. For example,
<Person> *sonia gandhi* < /Person>, <Location> *kolkata* < /Location> and <Organization> *jadavpur viswavidyalya*< /Organization>.

   For each tag T inserted in the training corpus, the algorithm generates a lexical pattern p using a context window of maximum width 6 (excluding the tagged NE) around the left and the right tags, e.g., $p = [l_{-3}l_{-2}l_{-1} < T > \ldots < /T > l_{+1}l_{+2}l_{+3}]$, where, l_i are the context of p. Any of l_i may be a punctuation symbol. In such cases, the width of the lexical patterns will vary. We also generate the lexical context patterns by considering the left and right contexts of the labeled examples of the annotated corpus of 272K wordforms. All these patterns, derived from the different tags of the labeled and unlabeled training corpora, are stored in a Pattern Table (or, set P), which has four different fields namely, pattern *id* (identifies any particular pattern), pattern *example* (pattern), pattern *type* (*Person name*/*Location name*/*Organization name*) and *relative frequency* (indicates the number of times any pattern of a particular *type* appears in the entire training corpus relative to the total number of patterns generated of that *type*). This table has 38,198 entries, out of which 27,123 patterns are distinct. Labeled training data contributes to 15,488 patterns and the rest is generated from the unlabeled corpus.

3. Evaluation of patterns: Every pattern p in the set P is matched against the same unlabeled corpus. In a place, where the context of p matches, p predicts the occurrence of the left or right boundary of name. POS information of the words as well as some linguistic rules and/or length of the entity have been used in detecting the other boundary. The extracted entity may fall in one of the following categories:

Table 8: Gazetteers used in the experiment

Gazetteer	Number of entries	Source
NE suffix	115	Manually prepared
Organization suffix	94	Manually created from the news corpus
Person prefix	245	Manually created from the news corpus
Middle name	1491	Semi-automatically from the news corpus
Surname	5,288	Semi-automatically from the news corpus
Common Location	547	Manually developed
Action verb	221	Manually prepared
Designation words	947	Semi-automatically prepared from news corpus
First names	72,206	Semi-automatically prepared from the news corpus
Location name	5,125	Semi-automatically prepared from the news corpus
Organization name	2,225	Manually prepared
Month name	24	Manually prepared
Weekdays	14	Manually prepared
Measurement expressions	52	Manually prepared

Table 9: Descriptions of the language dependent features. Here, *i* represents the position of the current word and w_i represents the current word

Feature	Description
FirstName	$FirstName_i = I_{\{\text{First name list}\}}(w_i)$
MidName	$MidName_i = I_{\{\text{Middle name list}\}}(w_i)$
SurName	$SurName_i = I_{\{\text{Sur name list}\}}(w_i) \bigvee I_{\{\text{Sur name list}\}}(w_{i+1})$
Funct	$Funct_i = I_{\{\text{Function word list}\}}(w_i)$
MonthName	$MonthName_i = I_{\{\text{Month name list}\}}(w_i)$
WeekDay	$WeekDay_i = I_{\{\text{Week day list}\}}(w_i)$
MeasureMent	$Measurement_i = I_{\{\text{Measurement word list}\}}(w_{i+1})$ $\bigvee I_{\{\text{Measurement list}\}}(w_{i+1})$
POS	POS_i=POS tag of the current word
NESuf	$NESuf_i = I_{\{\text{NE suffix list}\}}(w_i)$
OrgSuf	$OrgSuf_i = I_{\{\text{Organization suffix word list}\}}(w_i)$ $\bigvee I_{\{\text{Organization suffix word list}\}}(w_{i+1})$
ComLoc	$ComLoc_i = I_{\{\text{Common location list}\}}(w_i)$
ActVerb	$ActVerb_i = I_{\{\text{Action verb list}\}}(w_i)$ $\bigvee I_{\{\text{Action verb ist}\}}(w_{i+1})$
DesG	$DesG_i = I_{\{\text{Designation word list}\}}(w_{i-1})$
PerPre	$PerPre_i = I_{\{\text{Person prefix word list}\}}(w_{i-1})$
LocName	$LocName_i = I_{\{\text{Location name list}\}}(w_i)$
OrgName	$OrgName_i = I_{\{\text{Organization name list}\}}(w_i)$

Table 10: Corpus statistics

Total number of news documents in the corpus	35, 143
Total number of sentences in the corpus	940, 927
Average number of sentences in a document	27
Total number of wordforms in the corpus	9, 998, 972
Average number of wordforms in a document	285
Total number of distinct wordforms in the corpus	152, 617

(a) *positive example*: The extracted entity is of the same NE type as that of the pattern.

(b) *negative example*: The extracted entity is of the different NE type as that of the pattern.

(c) *error example*: The extracted entity is not at all a NE.

4. Candidate pattern acquisition: For each pattern p, we have maintained three different lists for the *positive*, *negative* and *error* examples. The *type* of the extracted entity is determined by checking whether it appears in any of the seed lists (person/location/organization); otherwise, its *type* is determined manually. The *positive* and *negative* examples are then added to the appropriate seed lists. We then compute the pattern's *accuracy* as follows: $accuracy(p) = \frac{|positive(p)|}{[|positive(p)|+|negative(p)|+|error(p)|]}$

   A threshold value of *accuracy* has been chosen in order to discard the patterns below this threshold. A pattern is also discarded if its total *positive count* is less than a predetermined threshold value. The remaining patterns are ranked by their *relative frequency* values. The n top high frequent patterns are retained in the pattern set P and this set is denoted as *Accept Pattern*.

5. Generation of new patterns: All the positive and negative examples extracted by a pattern p in Step 4 can be used to generate further patterns from the same training corpus. Each new *positive* or *negative* instance (not appearing in the seed lists) is used to further tag the training corpus. We repeat steps 2-4 for each new NE until no new patterns can be generated. The threshold values of *accuracy, positive count* and *relative frequency* are chosen in such a way that in each iteration of the algorithm at least 5% new patterns are added to the set P. A newly generated pattern may be identical to a pattern that is already in the set P. In such a case, the *type* and *relative frequency* fields in the set P are updated accordingly. Otherwise, the newly generated pattern is added to the set with the *type* and *relative frequency* fields set properly. The algorithm terminates after 23 iterations and there are 34,298 distinct entries in the set P.

5.2 Unlabeled document and sentence selection using bootstrapping

We have divided the unlabeled 35,143 news documents based on news sources/types, i.e., International, National, State, District, Metro [Kolkata], Politics, Sports, Business etc. in order to create segments of manageable size. This helps us to separately evaluate the contribution of each segment using a gold standard development test set and reject those that are not helpful and to apply the latest updated best model to each subsequent segment. We have observed that the use of unlabeled data becomes effective if it is related to the target problem, i.e., the test set. So, appropriate unlabeled document selection is very essential. After selecting the documents, it is necessary to select the tagged sentences that are useful to improve the system performance. Appropriate sentences are selected based on majority voting and depending upon the structure and/or the contents of the sentences.

- Unlabeled Document Selection: The unlabeled data supports the acquisition of new names and contexts to provide new evidences to be incorporated in ME, CRF and SVM classifiers. Old estimates of the models may be worsened by the unlabeled data if it adds too many names whose tags are incorrect, or at least are incorrect in the context of the labeled training data and the test data. Unlabeled data can degrade rather than improve the classifier's performance on the test set if it is irrelevant to the test document. So, it is necessary to measure the relevance of the unlabeled data to our target test set.

  We construct a set of key words from the test set T to check whether unlabeled document d is useful or not.

  - We do not use all the words in test set T as the key words since we are only concerned about the distribution of name candidates. So, each document is tested with the CRF model that is developed with the language independent features (i.e, *baseline*), context features and gazetteers.
  - It is insufficient to take only the name candidates in the top one hypothesis for each sentence.

  Thus, we take all the name candidates in the top N best hypotheses ($N = 10$) for each sentence of the test set T to construct a query set Q. Using this query set, we find all the relevant documents that include three (heuristically set) names belonging to the set Q. In addition, the documents are not considered if they contain fewer than seven (heuristic) names.

– Sentence Selection: All the tagged sentences of a relevant document are not added to training corpus as incorrectly tagged or irrelevant sentences can lead to the degradation in model performance. We are actually concerned on how much new information is extracted from each sentence of the unlabeled data compared to the training corpus that already we have in our hand.

  We have used majority voting approach to select the relevant sentences. All the relevant documents are tagged with the ME, CRF, SVM-F and SVM-B models. If the majority of models agree to the same output for at least 80% of the words in a sentence then that sentence is selected to be added to the training corpus. This criterion often selects some sentences which are too short or do not include any name. These words may make the model worse if added to the training data. For example, the distribution of non-names may increase significantly leading to degradation of model performance. In this experiment, we have not included the sentences that include fewer than five words or do not include any names.

The bootstrapping procedure is given as follows:

1. Select a relevant document *RelatedD* from a large corpus of unlabeled data with respect to the test set T using the document selection method described earlier.

2. Split *RelatedD* into n subsets and mark them $C_1, C_2, \ldots, C_n$.

3. Call the development set *DevT*.

4. For $I = 1$ to n

   (a) Run initial ME, CRF, SVM-F and SVM-B on C_i.

   (b) For each tagged sentence S in C_i , if at least 80% of the words agree with the same outputs by the majority of models then keep S; otherwise, remove S.

   (c) Assign outputs to the remaining words from the SVM-F model.

   (d) If the length of S is less than five words or it does not contain any name then discard S.

   (e) Add C_i to the training data and retrain each model. This produces the updated models.

   (f) Run the updated models on *DevT*; if the performance gets reduced then do not use C_i and use the old models.

5. Repeat steps 1-4 until performance of each model becomes identical in two consecutive iterations.

Table 11: Statistics of the training, development and test sets

	Training	Development	Test
# of sentences	21,340	3,367	2,501
#of wordforms	272,000	50,000	35,000
#of NEs	22,488	3,665	3,178
#Avg. length of NE	1.5138	1.6341	1.6202

6 Evaluation results and discussions

We have manually annotated approximately 200K wordforms of the Bengali news corpus [55] with *Person name*, *Location name*, *Organization name* and *Miscellaneous name* NE tags with the help of *Sanchay Editor* [6] , a text editor for the Indian languages. Out of 200K wordforms, 150K wordforms along with the IJCNLP-08 shared task data has been used for training the models. Out of 200K wordforms, 50K wordforms have been used as the development data. The system has been tested with a gold standard test set of 35K wordforms. Statistics of the training, development and test sets are given in Table 11.

A number of experiments have been carried out taking the different combinations of the available words, context and orthographic word level features to identify the best-suited set of features in the ME, CRF and SVM frameworks for NER in Bengali. Evaluation results of the development set for the *baseline* models are presented in Table 12. The *baseline* ME based system performs best for the context word window of size three, dynamic NE tag of the previous word, suffixes and prefixes of length upto three characters of the current word, POS tag of the current word and other word-level language independent features. The system has demonstrated the overall f-score value of 72.49%. The *baseline* CRF model has shown best performance with the f-score of 75.71% for the context window of size five, dynamic NE information of the previous word, POS information of the current and previous words, prefixes and suffixes of length upto three characters of the current word along with other features. The SVM-F based *baseline* system has performed best among the three models and has demonstrated the f-score value of 76.3% for the context window of size six, NE information of the previous two words, POS information of the current, previous and the next words along with the other set of features as like CRF. The SVM-B has shown the f-score value of 76.1% with the same set of features used in SVM-F. In SVM models, we have conducted experiments with the different *polynomial kernel* functions and observed the highest f-score value with degree 2.

The language dependent features as described in Table 9 are included into the *baseline* models and the evaluation results are reported in Table 13. We have observed that all the gazetteers are not equally important to improve the per-

[6]Sourceforge.net/project/nlp-sanchay

Table 12: Results of the *baseline* models

Model	R (in %)	P (in %)	FS (in %)
ME	73.55	71.45	72.49
CRF	75.97	75.45	75.71
SVM-F	77.14	75.48	**76.30**
SVM-B	77.09	75.14	76.10

Table 13: Results including language dependent features

Model	R (in %)	P (in %)	FS (in %)
ME	75.26	74.91	74.41
CRF	79.03	80.62	79.82
SVM-F	81.37	80.14	**80.75**
SVM-B	81.29	79.16	80.21

formance of the classifiers. The use of gazetteers increases the performance by 2.43%, 4.11%, 4.45%, and 4.11% in the ME, CRF, SVM-F, and SVM-B classifiers, respectively. Results show that the effect of language dependent features is not very impressive in ME model. Thus, it can be decided that the use of all available features can not always improve the performance in a ME model and careful feature selection is very important.

6.1 Use of context patterns as features

High ranked patterns in the *Accept Pattern set* can be used as the features of the individual classifier. Words in the left and/or the right contexts of person, location and organization names carry effective information that could be helpful in their identification. A feature 'ContextInformation' is defined by observing the words in the window $[-3, 3]$ (three words spanning to left and right) of the current word in the following way:
•Feature value is 1 if the window contains any word of the pattern type *Person name*.
• Feature value is 2 if the window contains any word of the pattern type *Location name*.
•Feature value is 3 if the window contains any word of the pattern type *Organization name*.
• Feature value is 4 if the window contains any word that appears with more than one type.
• Feature value is 0 for those if the window does not contain any word of any pattern.

Experimental results of the system for the development set are presented in Table 14 by including the context features. Evaluation results show the effectiveness of context features with the improvement of f-scores by **3.17%**, **3.08%, 2.82%**, and **3.28%** in the ME, CRF, SVM-F, and SVM-B models, respectively. So, the context features are effective in improving the performance of all the models.

Table 14: Results using context features

Model	R (in %)	P (in %)	FS (in %)
ME	78.26	76.91	77.58
CRF	82.07	83.75	82.90
SVM-F	84.56	82.60	83.57
SVM-B	84.42	82.58	83.49

6.2 Post-processing techniques

We have conducted error analysis for all the classifiers with the help of confusion matrices. Several post-processing techniques have been adopted in order to improve the performance of each of the classifiers. It has been observed that the SVM models have the highest tendency of assigning NE tags to the words that are actually not NEs. In ME model, a lot of NEs are not identified at all. CRF model also suffers from this problem. The most confusing pairs of classes in these two models are LOC vs NNE, MISC vs NNE, PER vs NNE, E-ORG vs NNE and B-MISC vs MISC. On the other hand the most confusing pairs are LOC vs NNE, PER vs NNE, MISC vs NNE and E-ORG vs NNE. Depending upon the errors involved in the models, we have developed various mechanisms to improve the recall and precision values of the classifiers.

– Class decomposition technique for SVM: Unlike CRF, SVM model does not predict the NE tags to the constituent words depending upon the sentence. SVM predicts the class depending upon the labeled word examples only. If target classes are equally distributed, the *pairwise* method can reduce the training cost. Here, we have a very unlabeled class distribution with a large number of samples belonging to the class 'NNE' (other than NEs) (Table 11). This leads to the same situation like *one-vs-rest* strategy. One solution to this unbalanced class distribution is to decompose the 'NNE' class into several subclasses effectively. Here, we have decomposed the 'NNE' class according to the POS information of the word. That is, given a POS tagset POS, we produce new $|POS|$ classes, '$NNE - C'|C \in POS$. So, we have 26 subclasses which correspond to non-NE regions such as 'NNE-NN' (common noun), 'NNE-VFM' (verb finite main) etc. Experimental results have shown the recall, precision, and f-score values of 87.09%, 86.73%, and 86.91%, respectively, in the SVM-F model and 87.03%, 85.98%, and 86.5%, respectively, in SVM-B model. We have also conducted similar experiments in the CRF models and observed the lower f-score values.

– Post-processing with the n-best outputs for CRF: There are inconsistent results in the CRF model in some cases. We have performed a post-processing step to correct these errors. The post-processing tries to assign the correct tag according to the n-best results for every sentence of the test set. We have considered the top 15 labeled sequences for each sentence with the confidence scores. Initially, we collect the NEs from the high confident results and then we re-assign the tags for low confident results using this NE list. The procedure is given below: S is the set of sentences in the test set, i.e, $S = \{s_1, s_2, \ldots, s_n\}$; R is set of n-best result ($n = 15$) of S, i.e, $R = \{r_1, r_2, \ldots, r_n\}$, where r_i is a set of n-best results of s_i; c_{ij} is the confidence score of r_{ij}, that is the jth result in r_i.

Creation of NE set from the high confident tags:

```
for $i = 1$ to $n$ {if ($r_{i0} \geq 0.6$) then collect all NEs
from $r_{i0}$ and add to the set NESet }.
```

Replacement:

```
for $i = 1$ to $n$ {if ($r_{i0} \geq 0.6$) then Result($s_i$ )=$r_{i0}$ ;
else { TempResult($s_i$ )=$r_{i0}$ ;
for $j = 1$ to $m$ {if ( NEs of $r_{ij}$ are included in NESet)
then Replace the NE tags
of TempResult with these new tags}.
Result($s_i$) =TempResult($s_i$ )}}.
```

Evaluation results have demonstrated the recall, precision, and f-score values of 86.75%, 85.91%, and 86.33%, respectively, in the CRF model. Thus, there is an improvement of **4.43%** f-score in the CRF model.

– Post-processing the output of ME model: We have used the following heuristics to further improve the performance of the ME model. Some of the rules are useful to improve the recall values, whereas some are effective to increase the precisions. Many of the heuristics are also helpful to identify the boundaries properly. Following are the set of heuristics.

1. The NNE tag of a particular word is replaced by the appropriate NE tag, if that word appears somewhere in the output with that NE.
2. If any word is tagged as B-XXX/I-XXX/E-XXX (XXX: PER/LOC/ORG/MISC) and the previous and next words are tagged as NNE then that word is assigned the NE tag of type XXX.
3. The NNE tag of a word is replaced by the E-XXX if the previous word is already tagged as B-XXX.
4. NNE tag of a word is replaced by B-XXX, if the next word is already tagged as E-XXX.
5. If there is sequence B-XXX/I-XXX followed by XXX in the output, then the tag XXX is replaced by the E-XXX.
6. If the sequence of tags is of the form XXX B-XXX1/I-XXX1/E-XXX1 NNE (XXX#XXX1) for three consecutive words in the output, then the tag B-XXX1/I-XXX1/E-XXX1 is replaced by the XXX1.
7. If current word is not tagged as B-XXX/I-XXX/NNE but the following word is tagged as B-XXX/I-XXX/E-XXX then the current word is assigned the tag B-XXX.
8. If the words, tagged as NNE, contain the variable length NE suffixes (used as the feature in the *baseline* models) then the words are assigned the NE tags. The types of the NE tags are determined by the types of the suffixes (e.g., Person tag is assigned if matches with the person name suffix).

Evaluation results have demonstrated the recall, precision, and f-score values of 81.55%, 78.67%, and 80.8%, respectively.

6.3 Impact of unlabeled data selection

In order to investigate the contribution of document selection in bootstrapping, we run the post-processed models on 35,143 news documents. This yields the gradually improving performance for the models as shown in Table 15.

We have also carried out experiments with the same unlabeled data in order to observe the effectiveness of document selection and sentence selection separately. Results are reported in Table 16. Row 2 of the table represents results of the post-processed models that are used to tag the unlabeled documents to be included into the initial training set in a bootstrapped manner. This presents the results by using the majority voting selection criterion only. Comparing row 2 with row 3, we find that not using document selection, even though it multiplies the size of the training corpus, results in 1.04%, 1.36%, 1.02%, and 0.83% lower performance in the ME, CRF, SVM-B, and SVM-F models, respectively. This leads us to conclude that simply relying upon large corpus is not in itself sufficient. Effective use of large corpus demands good selection criterion of documents to remove off-topic materials. The system has demonstrated the f-score values of 83.87%, 89.34%, 89.55%, and 89.37% in the ME, CRF, SVM-F, and SVM-B models, respectively, by adding the sentence selection method.

6.4 Voting techniques

Voting scheme is effective in order to improve the overall performance of any multi-engine system. Here, we have combined four models using three different voting mechanisms. But before applying weighted voting, we need to decide the weights to be given to the individual system. We can obtain the best weights if we could obtain the accuracy for the 'true' test data. However, it is impossible to estimate them. Thus, we have used following weighting methods in our experiments:

1. Uniform weights (Majority voting): We have assigned the same voting weight to all the systems. The com-

Table 15: Incremental improvement of performance

Itera tion	Sentences added	FS (in %)			
		ME	CRF	SVM-F	SVM-B
0	0	80.8	86.33	86.91	86.5
1	107	81.2	86.9	87.27	87.13
2	213	81.67	87.35	87.53	87.41
3	311	81.94	87.93	88.12	87.99
4	398	82.32	88.11	88.25	88.18
5	469	82.78	88.66	88.83	88.71
6	563	82.94	89.03	89.17	89.08
7	619	83.56	89.12	89.27	89.15
8	664	83.79	89.28	89.35	89.22
9	691	83.85	89.34	89.51	89.37
10	701	83.87	89.34	89.55	89.37
11	722	83.87	89.34	89.55	89.37

Table 16: Incremental improvement of performance

Model		ME	CRF	SVM-F	SVM-B
1	Post-processed	80.8	86.33	86.91	86.50
2	(1)+ Bootstrapping	82.01	87.36	88.05	87.81
3	(2) + Document selection	83.05	88.72	88.88	88.83
4	(3) + Sentence selection	83.87	89.34	89.55	89.37

bined system selects the classifications, which are proposed by the majority of the models. If four outputs are different, then the output of the SVM-F system is selected.

2. Cross validation f-score values: The training data is divided into N portions. We employ the training by using $N - 1$ portions, and then evaluate the remaining portion. This is repeated N times. In each iteration, we have evaluated the individual system following the similar methodology, i.e., by including the various gazetteers and the same set of post-processing techniques. At the end, we get N f-score values for each of the system. Final voting weight for a system is given by the average of these N f-score values. Here, we set the value of N to be 10. We have defined two different types of weights depending on the cross validation f-score as follows:

    – Total F-Score: In the first method, we have assigned the overall average f-score of any classifier as the weight for it.

    – Tag F-Score: In the second method, we have assigned the average f-score value of the individual tag as the weight for that model.

Experimental results of the voted system are presented in Table 17. Evaluation results show that the system achieves the highest performance for the voting scheme 'Tag F-Score'. Voting shows (Tables 16-17) an overall improvement of **8.84%** over the least performing ME based system and **3.16%** over the best performing SVM-F system in terms of f-score values.

Table 17: Results of the voted system (development set)

Voting	R (in %)	P (in %)	FS (in %)
Majority	93.19	89.35	91.23
Total F-Score	93.85	89.97	92.17
Tag F-Score	93.98	91.46	92.71

6.5 Experimental results of the test set

The systems have been tested with a gold standard test set of 35K wordforms. Approximately, 25% of the NEs are unknown in the test set. Experimental results of the test set for the *baseline* models have shown the f-score values of 73.15%, 76.35%, 77.36%, and 77.23% in the ME, CRF, SVM-F, and SVM-B based systems, respectively. Results have demonstrated the improvement in f-scores by 8.35%, 9.67%, 8.82% and 8.83% in the ME, CRF, SVM-B, and SVM-F models, respectively, by including the language specific features, context features and post-processing techniques. Appropriate unlabeled sentences are then selected by the document and sentence selection methods to be included into the training set. Models have shown the f-scores of 83.77%, 89.02%, 89.17%, and 89.11% in the ME, CRF, SVM-F, and SVM-B models, respectively. Experi-

Table 18: Results of the voted system (test set)

Voting	R (in %)	P (in %)	FS (in %)
Majority	92.91	89.77	91.31
Total F-Score	93.55	90.16	91.82
Tag F-Score	93.79	91.34	92.55

Table 19: Comparison with other Bengali NER systems

Model	R(%)	P (%)	FS (%)
A ([40])	66.53	63.45	64.95
B ([40])	69.32	65.11	67.15
HMM ([41])	74.02	72.55	73.28
ME ([42])	78.64	76.89	77.75
CRF ([43])	80.02	80.21	80.15
SVM ([68])	81.57	79.05	80.29
Proposed system	93.79	91.34	92.55

mental results of the voted system are presented in Table 18. Results show that the voting scheme that considers the f-score value of the individual NE tag as the weight of a particular classifier, i.e., 'Tag F-Score' gives the best result among the three voting methods. The voted system has demonstrated the improvement in the f-scores by 8.78%, 3.53%, 3.38%, 3.44%, in the ME, CRF, SVM-F, and SVM-B systems, respectively.

The existing Bengali NER systems based on the pattern directed shallow parsing approach[40], HMM [41], ME [59], CRF [43], and SVM [68] have been evaluated with the same datasets. Comparative evaluation results are presented in Table 19. Comparisons with the works reported in the IJCNLP-08 shared task are out of scope because of the following reasons:

– The shared task was involved with a fine-grained tagset of twelve NE tags. In this work, we have considered only the tags that denote person name, location name, organization name, date, time and number expressions.

– The main challenge of the shared task was to identify and classify the nested NEs (i.e, the constituent parts of a bigger NE). Here, we are not concerned with the nested NEs.

Results show the effectiveness of the proposed NER system that outperforms other existing systems by the impressive margins. Thus, it can be decided that contextual information of the words, several post-processing methods and the use of appropriate unlabeled data can yield a reasonably good performance. Results also suggest that combination of several classifiers is more effective than any single classifier.

7 Conclusion

In this paper, we have reported a NER system by combining the classifiers, namely ME, CRF and SVM with the help of weighted voting techniques. We have manually annotated a portion of the Bengali news corpus, developed from the web archive of a leading Bengali newspaper. In addition, we have also used the IJCNLP-08 NER shared task data tagged with a fine-grained NE tag set of twelve tags. We have converted this data with the NE tags denoting person name, location name, organization name and miscellaneous name. The individual models make use of the different contextual information of the words, several orthographic word-level features and the binary valued features extracted from the various gazetteers that are helpful to predict the NE classes. A number of features are language independent in nature. We have used an unsupervised learning technique to generate lexical context patterns to be used as features of the classifiers. We have described the method of selecting appropriate unlabeled documents and sentences from a large collection of unlabeled data. This eliminates the necessity of manual annotation for preparing the NE annotated corpus. We have also shown how several heuristics for ME, n-best output of CRF and the class splitting technique of SVM are effective in improving the performance of the corresponding model. Finally, the outputs of the classifiers have been combined with the three different weighted voting techniques. It has been shown that combination of several models performs better than any single one.

References

[1] Cunningham, H.: GATE, a General Architecture for Text Engineering. Computers and the Humanities **36** (2002) 223–254

[2] Babych, B., Hartley, A.: Improving Machine Translation Quality with Automatic Named Entity Recognition. In: Proceedings of EAMT/EACL 2003 Workshop on MT and other Language Technology Tools. (2003) 1–8

[3] Moldovan, D., Harabagiu, S., Girju, R., Morarescu, P., Lacatusu, F., Novischi, A., Badulescu, A., Bolohan, O.: LCC Tools for Question Answering. In: Text REtrieval Conference (TREC) 2002. (2002)

[4] Y.C. Wu, T.K. Fan, Y.L., Yen, S.: Extracting Named Entities using Support Vector Machines. In: Springer-Verlag. (2006)

[5] I. Budi, S.B.: Association Rules Mining for Name Entity Recognition. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering. (2003)

[6] Appelt, D.E., Hobbs, J.R., Bear, J., Israel, D.J., Tyson, M.: Fastus: A finite-state processor for information extraction from real-world text. In: IJCAI. (1993) 1172–1178

[7] Appelt, D.E., Hobbs, J.R., Bear, J., Israel, D., Kameyama, M., Kehler, A., Martín, D., Myers, K., Tyson, M. Proceedings of the 6th Messsage Understanding Conference. In: SRI International FASTUS System MUC-6 Test Results and Analysis. Morgan Kaufmann Publishers, Inc., Columbia, Maryland (1995) 237–248

[8] Iwanska, L., Croll, M., Yoon, T., Adams, M.: Wayne State University: Description of the UNO Processing System as used for MUC-6Language Independent Named Entity Recognition. In: Proceedings of the MUC-6, Morgan-Kauffman Publisher (1995)

[9] Grishman, R.: The NYU System for MUC-6 or Where's the Syntax. In: Proceedings of the MUC-6, Morgan-Kauffman Publisher (1995)

[10] Farmakiotou D., Karkaletsis V., K.J.S.G.S.C., P., S.: Rule-based Named Entity Recognition for Greek Financial Text. In: Proceedings of the Workshop on Computational Lexicography and Multimedia Dictionaries (COMLEX 2000). (2000) 75–78

[11] Wolinski F., V.F., B., D.: Automatic Processing of Proper Names in Texts. In: In Proceedings of the European Chapter of the Association for Computer Linguistics (EACL), Dublin, Ireland (1995) 23–30

[12] Wolinski F., V.F., M., S.: Using Learning-based Filters to Detect Rule-based Filtering Obsolescence. In: In Recherche d' Information Assistee par Ordinateur (RIAO), Paris, France (2000) 1208–1220

[13] Humphreys, K., Gaizauskas, R., Azzam, S., Huyck, C., Mitchell, B., Cunnigham, H., Wilks, Y.: Univ. Of Sheffield: Description of the LaSIE-II System as Used for MUC-7. In: MUC-7, Fairfax, Virginia (1998)

[14] Aone, C., Halverson, L., Hampton, T., Ramos-Santacruz, M.: SRA: Description of the IE2 system used for MUC-7. In: MUC-7, Fairfax, Virginia (1998)

[15] Mikheev, A., Grover, C., Moens, M.: Description of the LTG system used for MUC-7. In: MUC-7, Fairfax, Virginia (1998)

[16] Mikheev, A., Grover, C., Moens, M.: Named Entity Recognition without Gazeteers. In: Proceedings of EACL, Bergen, Norway (1999) 1–8

[17] Zhou, G., Su, J.: Named Entity Recognition using an HMM-based Chunk Tagger. In: Proceedings of ACL, Philadelphia (2002) 473–480

[18] Miller, S., Crystal, M., Fox, H., Ramshaw, L., Schawartz, R., Stone, R., Weischedel, R., the Annotation Group: BBN: Description of the SIFT System as Used for MUC-7. In: MUC-7, Fairfax, Virginia (1998)

[19] Bikel, D.M., Schwartz, R.L., Weischedel, R.M.: An Algorithm that Learns What's in a Name. Machine Learning **34** (1999) 211–231

[20] Borthwick, A.: Maximum Entropy Approach to Named Entity Recognition. PhD thesis, New York University (1999)

[21] Borthwick, A., Sterling, J., Agichtein, E., Grishman, R.: NYU:Description of the MENE Named Entity System as Used in MUC-7. In: MUC-7, Fairfax (1998)

[22] Sekine, S.: Description of the Japanese NE System used for MET-2. In: MUC-7, Fairfax, Virginia (1998)

[23] Bennet, S.W., Aone, C., Lovell, C.: Learning to Tag Multilingual Texts Through Observation. In: Proceedings of Empirical Methods of Natural Language Processing, Providence, Rhode Island (1997) 109–116

[24] McCallum, A., Li, W.: Early results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons. In: Proceedings of CoNLL, Canada (2003) 188–191

[25] Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: ICML. (2001) 282–289

[26] Yamada, H., Kudo, T., Matsumoto, Y.: Japanese Named Entity Extraction using Support Vector Machine. In Transactions of IPSJ **43** (2001) 44–53

[27] Kudo, T., Matsumoto, Y.: Chunking with Support Vector Machines. In: Proceed-ings of NAACL. (2001) 192–199

[28] Takeuchi, K., Collier, N.: Use of Support Vector Machines in Extended Named Entity Recognition. In: Proceedings of the 6th Conference on Natural Language Learning, (CoNLL-2002). (2002) 119–125

[29] Masayuki, A., Matsumoto, Y.: Japanese Named Entity Extraction with Redundant Morphological Analysis. In: NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, Morristown, NJ, USA, Association for Computational Linguistics (2003) 8–15

[30] Collins, M., Singer, Y.: Unsupervised Models for Named Entity Classification. In: Proceedings of the

Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora. (1999)

[31] J. Kim, I.K., Choi, K.: Unsupervised Named Entity Classification Models and their Ensembles. In: Proceedings of the 19th Conference on Computational Linguistics. (2002)

[32] A. Mikheev, C.G., Moens, M.: Description of the LTG System for MUC-7. In: Proceedings of Message Understanding Conference (MUC-7). (1998)

[33] R. Srihari, C.N., Li, W.: A Hybrid Approach for Named Entity and Sub-type Tagging. In: Proceedings of Sixth Conference on Applied Natural Language Processing. (2000) 247–254

[34] Wu, D., Ngai, G., Carpuat, M.: A stacked, voted, stacked model for named entity recognition. In: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003, Morristown, NJ, USA, Association for Computational Linguistics (2003) 200–203

[35] Florian, R., Ittycheriah, A., Jing, H., Zhang, T.: Named entity recognition through classifier combination. In: Proceedings of CoNLL-2003, Edmonton, Canada (2003) 168–171

[36] Miller, S., Guinness, J., Zamanian, A.: Name tagging with word clusters and discrimina-tive training. In: Proc of HLT-NAACL 2004, Boston, USA (2004) 337–342

[37] Lin, W., Yangarber, R., Grishman, R.: Bootstrapped learning of semantic classes from positive and negative examples. In: In Proceedings of ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data. (2003) 103–111

[38] Bean, D., Rilof, E.: Unsupervised learning of contextual role knowledge for coreference resolution. In: Proc. of HLT-NAACL 2004. (2004) 297–304

[39] Ji, H., Grishman, R.: Data selection in semi-supervised learning for name tagging. In: Proc. of the Workshop on Information Extraction Beyond the Document. (2006)

[40] Ekbal, A., Bandyopadhyay, S.: Lexical Pattern Learning from Corpus Data for Named Entity Recognition. In: Proceedings of ICON, India (2007) 123–128

[41] Ekbal, A., Naskar, S., Bandyopadhyay, S.: Named Entity Recognition and Transliteration in Bengali. Named Entities: Recognition, Classification and Use, Special Issue of Lingvisticae Investigationes Journal **30** (2007) 95–114

[42] Ekbal, A., Bandyopadhyay, S.: Maximum entropy approach for named entity recognition in bengali. In: Proceedings of the 7th International Symposium on Natural Language Processing (SNLP-08), Thailand (2007)

[43] Ekbal, A., R.Haque, Bandyopadhyay, S.: Named Entity Recognition in Bengali: A Conditional Random Field Approach . In: Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP 2008). (2008) 589–594

[44] Ekbal, A., Bandyopadhyay, S.: Bengali Named Entity Recognition using Support Vector Machine. In: Proceedings of Workshop on NER for South and South East Asian Languages, 3rd International Joint Conference on Natural Languge Processing (IJCNLP), India (2008) 51–58

[45] Ekbal, A., Bandyopadhyay, S.: Appropriate Unlabeled Data, Post-processing and Voting can Improve the Performance of a NER System. In: Proceedings of the 6th International Conference on Natural Language Processing (ICON), Pune, India, Macmillan Publishers (2008) 234–239

[46] Li, W., McCallum, A.: Rapid Development of Hindi Named Entity Recognition using Conditional Random Fields and Feature Induction. ACM Transactions on Asian Languages Information Processing **2** (2004) 290–294

[47] Cucerzon, S., Yarowsky, D.: Language independent named entity recognition combining morphological and contextual evidence. In: Proceedings of the 1999 Joint SIGDAT conference on EMNLP and VLC, Washington, D.C. (1999)

[48] Saha, S., Sarkar, S., Mitra, P.: A Hybrid Feature set based Maximum Entropy Hindi Named Entiy Recognition. In: Proceedings of the 3rd International Joint Conference in Natural Langauge Processing (IJCNLP 2008). (2008) 343–350

[49] Kumar, N., Bhattacharyya, P.: Named entity recognition in hindi using memm. Technical report, IIT Bombay, India (2006)

[50] Gali, K., Sharma, H., Vaidya, A., Shisthla, P., Sharma, D.M.: Aggregrating Machine Learning and Rule-based Heuristics for Named Entity Recognition. In: Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian Languages. (2008) 25–32

[51] Kumar, P.P., Kiran, V.R.: A Hybrid Named Entity Recognition System for South Asian Languages. In: Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian Languages. (2008) 83–88

[52] Srikanth, P., Murthy, K.N.: Named Entity Recognition for Telugu. In: Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian Languages. (2008) 41–50

[53] Vijayakrishna, R., Sobha, L.: Domain Focused Named Entity Recognizer for Tamil using Conditional Random Fields. In: Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian Languages. (2008) 93–100

[54] Shishtla, P.M., Pingali, P., Varma, V.: A Character n-gram Based Approach for Improved Recall in Indian Language ner. In: Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian Languages. (2008) 101–108

[55] Ekbal, A., Bandyopadhyay, S.: A Web-based Bengali News Corpus for Named Entity Recognition. Language Resources and Evaluation Journal **42** (2008) 173–182

[56] Bharati, A., R.S., Sharma, D.M.: Shakti Analyser: SSF Representation. In: http://shiva.iiit.ac.in/SPSAL2007/ssf-analysisrepresentation.pdf. (2005)

[57] Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag New York, Inc., New York, NY, USA (1995)

[58] Brants, T.: TnT a Statistical Parts-of-Speech Tagger. In: Proceedings of the Sixth Conference on Applied Natural Language Processing ANLP-2000. (2000) 224–231

[59] Ekbal, A., Haque, R., Bandyopadhyay, S.: Bengali Part of Speech Tagging using Conditional Random Field. In: Proceedings of Seventh International Symposium on Natural Language Processing, SNLP2007, Thailand (2007)

[60] Ekbal, A., Bandyopadhyay, S.: Web-based bengali News Corpus for Lexicon Development and POS Tagging. Polibits (ISSN 1870 9044) **37** (2008) 20–29

[61] Collins, M., Singer, Y.: Unsupervised models for named entity classification. In: Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora. (1999)

[62] Cucerzan, S., Yarowsky, D.: Language independent NER using a unified model of internal and contextual evidence. In: Proceedings of CoNLL 2002. (2002) 171–175

[63] Phillips, W., Riloff, E.: Exploiting strong syntactic heuristics and co-training to learn semantic lexicons. In: EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing, Morristown, NJ, USA, Association for Computational Linguistics (2002) 125–132

[64] Riloff, E., Jones, R.: Learning dictionaries for information extraction by multi-level bootstrapping. In: AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, Menlo Park, CA, USA, American Association for Artificial Intelligence (1999) 474–479

[65] Thelen, M., Riloff, E.: A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In: EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing, Morristown, NJ, USA, Association for Computational Linguistics (2002) 214–221

[66] Strzalkowski, T., Wang, J.: A self-learning universal concept spotter. In: Proceedings of the 16th conference on Computational linguistics, Morristown, NJ, USA, Association for Computational Linguistics (1996) 931–936

[67] Yangarber, R., Lin, W., Grishman, R.: Unsupervised learning of generalized names. In: Proceedings of the 19th international conference on Computational linguistics, Morristown, NJ, USA, Association for Computational Linguistics (2002) 1–7

[68] Ekbal, A., Bandyopadhyay, S.: Bengali Named Entity Recognition using Support Vector Machine. In: Proceedings of NERSSEAL, IJCNLP-08. (2008) 51–58

Assigning Library of Congress Classification Codes to Books Based Only on their Titles

Ricardo Ávila-Argüelles[1], Hiram Calvo[1,2], Alexander Gelbukh[1] and Salvador Godoy-Calderón[1]

[1] Center for Computing Research, National Polytechnic Institute
Mexico City, 07738, Mexico
E-mail: ravila06@sagitario.cic.ipn.mx; hcalvo@cic.ipn.mx; www.gelbukh.com; sgodoyc@cic.ipn.mx

[2] Nara Institute of Science and Technology
Takayama, Ikoma, Nara 630-0192, Japan
E-mail: calvo@is.naist.jp

Keywords: library classification, LCC, scarce information classification, logical-combinatorial methods

Received: February 4, 2009

Many publishers follow the Library of Congress Classification (LCC) scheme to indicate a classification code on the first pages of their books. This is useful for many libraries worldwide because it makes possible to search and retrieve books by content type, and this scheme has become a de facto standard. However, not every book has been pre-classified by the publisher; in particular, in many universities, new dissertations have to be classified manually. Although there are many systems available for automatic text classification, all of them use extensive information which is not always available, such as the index, abstract, or even the whole content of the work. In this work, we present our experiments on supervised classification of books by using only their title, which would allow massive automatic indexing. We propose a new text comparison measure, which mixes two well-known text classification techniques: the Lesk voting scheme and the Term Frequency (TF). In addition, we experiment with different weighing as well as logical-combinatorial methods such as ALVOT in order to determine the contribution of the title in the correct classification. We found this contribution to be approximately one third, as we correctly classified 36% (on average by each branch) of 122,431 previously unseen titles (in total) upon training with 489,726 samples (in total) of one major branch (Q) of the LCC catalogue.

Povzetek: Opisan je postopek klasifikacije knjig na osnovi naslovov v ameriški kongresni knjižnici.

1 Introduction

One of the most important tasks of librarians is book classification. A classification system designed to meet their requirements is the Library of Congress Classification (LCC) scheme, which is widely known and used by many important libraries in the world [6], being the system with the widest coverage of books. Besides using the previously assigned LCC code for each book, librarians need to classify other works such as dissertations, articles, magazines, which in most cases lack a previously assigned LCC code [7].

Given the size of the LCC list, manual assignment of an LCC category is a tedious and error-prone process. There exist systems that facilitate this process using automatic text classification techniques. However, such systems require extensive information about the book in machine-readable form, for example, an abstract, table of contents, or the complete text of the work. Providing such information when it is not available beforehand is costly and impractical.

Our motivation for this work was to develop an algorithm that is able to automatically assign a classification code based only on the most basic piece of information available: the title of the publication. We explore the level of attainment that it is possible to obtain given this strong restriction. On this way, we faced several problems, such as similar titles in different classes and a noisy data set, among others. We conducted tests using five supervised classification algorithms; some of them are rather simple, while other, such as those based on Logical-Combinatorial methods, are more sophisticated. In this paper we report on the results of these experiments and compare the methods that we considered.

Table 1: Comparison between our system and similar systems

Features \ Systems*	**1**	**2**	**3**	**4**	**Our system**
Uses *LCC*	✓	✓	✓	✓	✓
Uses book title	✓	✓	✓	✓	✓
Uses whole book contents		✓			
Uses LCSH thesaurus*	✓		✓	✓	
Uses MARC*			✓	✓	
Uses Text Categorization and IE techniques**		✓			
Uses Machine Learning	✓				
Uses Logical-Combinatorial methods					✓
Training set	800,000	19,000	800,000	1,500,000	489,726
Test set	50,000	1,029	50,000	7,200	122,431
Precision	55.00%	80.99%	86.00%	16.90%	**86.89%**

* See below. ** Information Extraction

In the next section, we present a review of existing related works. In Section 3, we describe different algorithms that we considered. In Section 4, we explain our experiments with the LCC catalog and present the experimental results. Finally, in Section 5 we draw our conclusions and outline the future work.

2 Related work

Table 1 summarizes previous works on book classification and compares them with our work as to the information and resources used and the resulting precision achieved. The systems compared in the table are as follows:

1. Predicting Library of Congress Classification from Library of Congress Subject Headings [3].
2. The Utility of Information Extraction in the classification of books [4].
3. Experiments in Automatic Library of congress Classification [5].
4. Challenges in automated classification using library classification schemes [2].

LCSH and MARC are lexical resources frequently used in similar works. They can be summarized as follows:

– LCSH (*Library of Congress Subject Headings*) is a collection of synonyms and antonyms of some terms related to book contents. This collection is updated by the Library of Congress. LCSH is widely used for book search where queries such as “body temperature regulation” should point to a title with the word “thermoregulation” [8].

– MARC (*Machine Readable Cataloguing*) defines a bibliographic data format. It provides a protocol for computers to exchange, use, and interpret bibliographic information [10], [13].

These resources are available only in English.

3 Classification algorithms we used

We implement a supervised learning technique. We assume that there is a collection of previously classified titles, and assign the new title to a category where the titles most similar to it belong in the training collection. The main difference between supervised learning techniques is in the definition of similarity. Accordingly, we have tested several methods of judging similarity between texts.

We tested several basic algorithms based on the simple classifier, or simple term matching. These basic algorithms and their variations, as well as more complex algorithms, are described in the following sections.

3.1 Algorithm 0: Frequency term voting

In this algorithm we first remove stopwords (function words, such as determiners and prepositions). Each title is compared with other titles from each class. For example, let Title 1 be compared with every other title in the collection (stop words appear strikethrough, because they are removed from consideration):

Title 1: ANATOMY ~~from~~ ~~the~~ GREEKS ~~to~~ HARVEY.

And for example, let Title 2 to be:

Title 2: SHORT HISTORY ~~of~~ ANATOMY ~~from~~ ~~the~~ GREEKS ~~to~~ HARVEY.

We count then the number of words intersected from the two titles, being in this example the similarity = 3:

Title A $\cap$ Title B = {ANATOMY, GREEKS, HARVEY}

Using Simple Term Matching technique, we take the terms contained in the title to be classified, and then we measure their frequency in each one of the classes that contain them. The selected class is chosen by being the one with the highest calculated frequency.

Consider a very simple example, on which we will illustrate different algorithms. Suppose we have a title with 4 terms and 4 subclasses from QA. Let A, F, D and C be four consecutive terms (words) from the title to be classified. 5 of these terms are present in class QA1, 7 in class QA103, 2 in QA242.5 and 4 in QA247. QA103 is the one with more votes; see Figure 1 and Table 3.

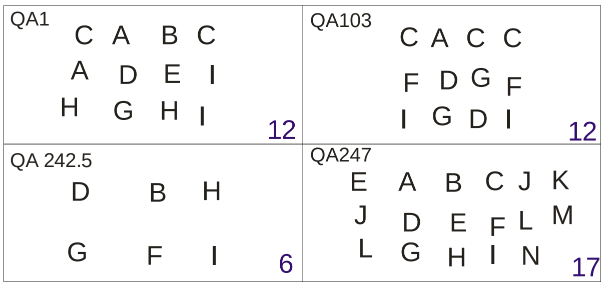


Figure 1: Example of terms ("words", such as A) belonging to titles for each classification (such as QA1)

Table 3: Frequency counts for the intersection of title AFDC and classes of Figure 1

	QA1	**QA103**	**QA242.5**	**QA247**
A	2	1	0	1
F	0	1	1	1
D	1	2	1	1
C	2	3	0	1
Frequency	**5**	**7**	**2**	**4**

The algorithm can be summarized as follows:

Algorithm 0. Frequency Term Voting

1. Extract title terms.
2. Remove stopwords (articles, prepositions, etc.).
3. Calculate the frequency of the terms in the class
4. Apply solution rule: the title belongs to the class with the greatest number of coincident terms.

3.2 Algorithm 1: Weighted term frequency voting

This algorithm considers the existence or absence of the terms in the title to be classified with regard to the classes where it should be classified, i.e., if the term is present in the class it will be counted as 1 and if not, as 0.

Following the same example as with Algorithm 0, we show the calculated Term Presence in Table 4.

Table 4: Presence counts for the intersection of title A F D C and classes of Figure 1

	QA1	**QA103**	**QA242.5**	**QA247**
A	1	1	0	1
F	0	1	1	1
D	1	1	1	1
C	1	1	0	1
Presence	**3**	**4**	**2**	**4**

It is common to have similar values, as it is shown in previous table (A F D C would be classified in QA103 as well as in QA247). To avoid this, we add a Term Frequency factor to the Term Presence. See Table 2.

The algorithm can be summarized as Algorithm 1.

Algorithm 1. Weighted Term Frequency Voting.

1. Extract the terms from Title *T*.
2. Remove stopwords.
3. Calculate Term Frequency of title *T* in all the classes, weighted by the total number of elements of each class.
4. Calculate Term Presence for all the classes.
5. Calculate *S*(*Title*, *Class*)= *Term Frequency for all classes + Term Presence in Class*.
6. Apply solution rule: the selected class for title *T* is the one with the highest *S*(*T*, *Class*).

3.3 Algorithm 2: Term frequency weighted by TF/IDF

Following the same example, first we calculate the Term Frequency. Then we calculate IDF for each row using the following formula [1]. Results are shown in Table 5.

$$IDF = \log\left(\frac{|\text{classes}|}{|\{C \mid C \text{ is a class and } w \in C\}|}\right)$$

Finally we multiply each row by its corresponding IDF value (for example, (2/12) × 0.124939 = 0.20823), which yields the data shown in Table 6. The selected class is the one with the greatest TF/IDF, in this case, QA103.

We can see from this table that the term D has zeroes in all columns, because it is present in all classes. Because of IDF, in general, any term present in every class has no effect in classification. On the contrary, a particular term is present mostly in a set of classes, and

Table 2: Term Frequency and Term Presence

Word \ Class:	QA1	QA103	QA242.5	QA247
A	2/12	1/12	0/6	1/17
F	0/12	1/12	1/6	1/17
D	1/12	2/12	1/6	1/17
C	2/12	3/12	0/6	1/17
Term Frequency	5/12 ≈ 0.4167	7/12 ≈ 0.5833	2/6 ≈ 0.3333	4/17 ≈ 0.2353
Term Presence in the class	3	4	2	4
Term Frequency + Term Presence	3.4167	4.5833	2.3333	4.2353

then it contributes to them proportionally to its lesser presence in other classes.

Table 5: TF and IDF calculation for the title A F D C

Class / Term	QA1	QA103	QA242.5	QA247	IDF
A	2/12	1/12	0/6	1/17	0.124939
F	0/12	1/12	1/6	1/17	0.124939
D	1/12	2/12	1/6	1/17	0
C	2/12	3/12	0/6	1/17	0.124939
TF	5/12	7/12	2/6	4/17	

Table 6: TF·IDF for the intersection of title A F D C and classes of Fig. 1

Class / Word	QA1	QA103	QA242.5	QA247	Total
A	0.020	0.010	0.000	0.007	0.038
F	0.000	0.010	0.020	0.007	0.038
D	0.000	0.000	0.000	0.000	0.000
C	0.020	0.031	0.000	0.007	0.059
Total	0.041	0.052	0.020	0.022	

Algorithm 2: Term Frequency weighted by TF·IDF.

```
1. Extract the terms from the title $T$.
2. Remove stopwords.
3. Use Algorithm 1 to calculate $TF$.
4. Calculate $IDF$.
5. Classify title T as class C for the class with the highest $TF·IDF$.
```

3.4 Algorithm 3: Term presence discrimination

Based on our observations of the previous algorithms, we propose this one. It is derived from Algorithm 1, but we use only the classes with the greatest presence of terms, while the other classes are discarded. Then we apply the TF·IDF classification from Algorithm 2 to the remaining classes.

Algorithm 3: Term presence discrimination.

```
1. Extract the terms from the title $T$.
2. Eliminate stopwords.
3. Calculate term presence for each term $w$ of title $T$ for each class $C$.
4. Calculate $M = max$ (term presence).
5. Remove classes with term presence $< M$.
6. If only one class is left then
     • Classify title $T$ in this class,
7. otherwise:
     • Calculate TF·IDF for the remaining classes.
     • Classify $T$ as member of the class with the highest TF·IDF.
```

For example, consider Table 4. We can see that only QA103 and QA247 are possible classifications. Then, using only them, we calculate their TF·IDF values.

3.5 Algorithms 4 and 4′: Title classification using logical-combinatorial methods

We experimented with the algorithm known as ALVOT. It has its origin in 1965 approximately [12]. It was developed by Yu. I. Zhuravliov and his group. ALVOT uses feature subsets called support groups or omega groups. For our analysis, we used the total set of terms from a title.

The model for voting algorithms has five components [11]:

1. Feature sets
2. Comparison criterion
3. Similarity function
4. Object evaluation (row) given a feature set
5. Class evaluation (column) for all feature sets
6. Rule of Solution

Feature sets: a non-empty set of features in terms of which all objects will be analysed.

Comparison criterion: A function with two descriptive features as input, from the same domain, and defining how they should be compared, giving a result within the range [0,1]: $Cc_i(A,B) \rightarrow [0,1]$, where A and B are descriptive features within the same domain.

Similarity function. It is a function that performs calculations using the defined comparison criteria for each feature comprising the object. The similarity function is normalized to the range [0, 1]. Its formal description is:

$$f = (M_1 \times M_2 \times ... \times M_r) \times (M_1 \times M_2 \times ... \times M_r) \rightarrow [0,1],$$

where M is the set of all features comprising the objects of a covering.

Evaluation by object (row) given a fixed feature set is performed once the feature set and the similarity function are defined. In this one a process of vote counting is performed, related to the similarity measure of the different features of the previously classified objects, with regard to those which are to be classified. Each row corresponding to one object is compared to the object to be classified by using the similarity measure.

Evaluation by class (column) for all the features set. It is the sum of the obtained evaluations of each one of the objects with regard to the object to be classified. This sum is a function from the evaluations by object obtained previously. That is, the belonging of the object is calculated with regard to the different classes of coverings.

Solution rule is a criterion for making a decision. Within this, the final vote is defined. The class of the object to be classified is decided, as well as its degree of belonging to this class.

In our specific case these concepts have the following meaning:

1. The objects to classify are titles. Each title has terms to be used by the similarity measure.

2. We compared each title to be classified, using a similarity measure, with every other title from the sample. The result of this is a matrix with the results of all comparisons. We call this matrix *similarity matrix*.
3. We created 2 similarity matrices based on two different similarity measures, that we will describe shortly.
4. Our *solution rule*:
   a. Defines the class to which each one of the titles belong.
   b. The degree of belonging of each title to the class. As we are using hard classes (it belongs, or it does not belong), then this value will be 0 or 1.

Each title with all of its terms (not separated, as in the previous algorithms) is assigned to the class where it belongs, and we compare the new title to be classified with all the previously classified titles.

We define two different similarity functions, explained in the following two sections.

3.5.1 Similarity between titles (Algorithm 4)

For this case the measure is expressed by the following formula:

$$f_{(T_i,T_j)} = \frac{STM}{\max\left(\left|T_i\right|,\left|T_j\right|\right)}$$

where

STM is the number of terms identical in the patterns,
T_i is the title to be classified,
T_j is the title previously classified,
$|\ T_i\ |$ and $|\ T_j\ |$ are the number of terms in T_i and T_j.

This means that the title will be compared with every other title. The class which contains the title with the greatest similarity will be selected.

3.5.2 Similarity of the title to be classified with all the titles in a class (Algorithm 4′)

For this case the measure is expressed by the following formula:

$$f\,(T_i, Q_j) = \frac{\sum_{T_p \in Q_j} f\left(T_i, T_p\right)}{\left|Q_j\right|}$$

where:

T_i is the title to be classified,
Q_j is a class to be evaluated for similarity with T_i.

The title to be classified will be compared with all the titles from each class, so that the class which in average has the greatest similarity will be chosen.

Consider the following example:

Title to classify: PRACTICAL MATHEMATICS, Length = 2.

From Table 7 we can see that the selected class would be the title with greatest similarity with the title to be classified, i.e., QA39. Figure 2 shows a screenshot of the system.

Algorithms 4 and 4′: Title Classification Using Logical-Combinatorial methods.

1. Extract the terms from Title *T*.
2. Eliminate stop words.
3. Calculate similarity of Title *T* with all of the other titles with at least one similar term.
4. Calculate *per* class average similarity.
5. Solution rule:
   Algorithm 4: The selected class is the one which contains the most similar title to Title *T*.
   Algorithm 4': The selected class is the one with greatest average similarity with Title *T*.

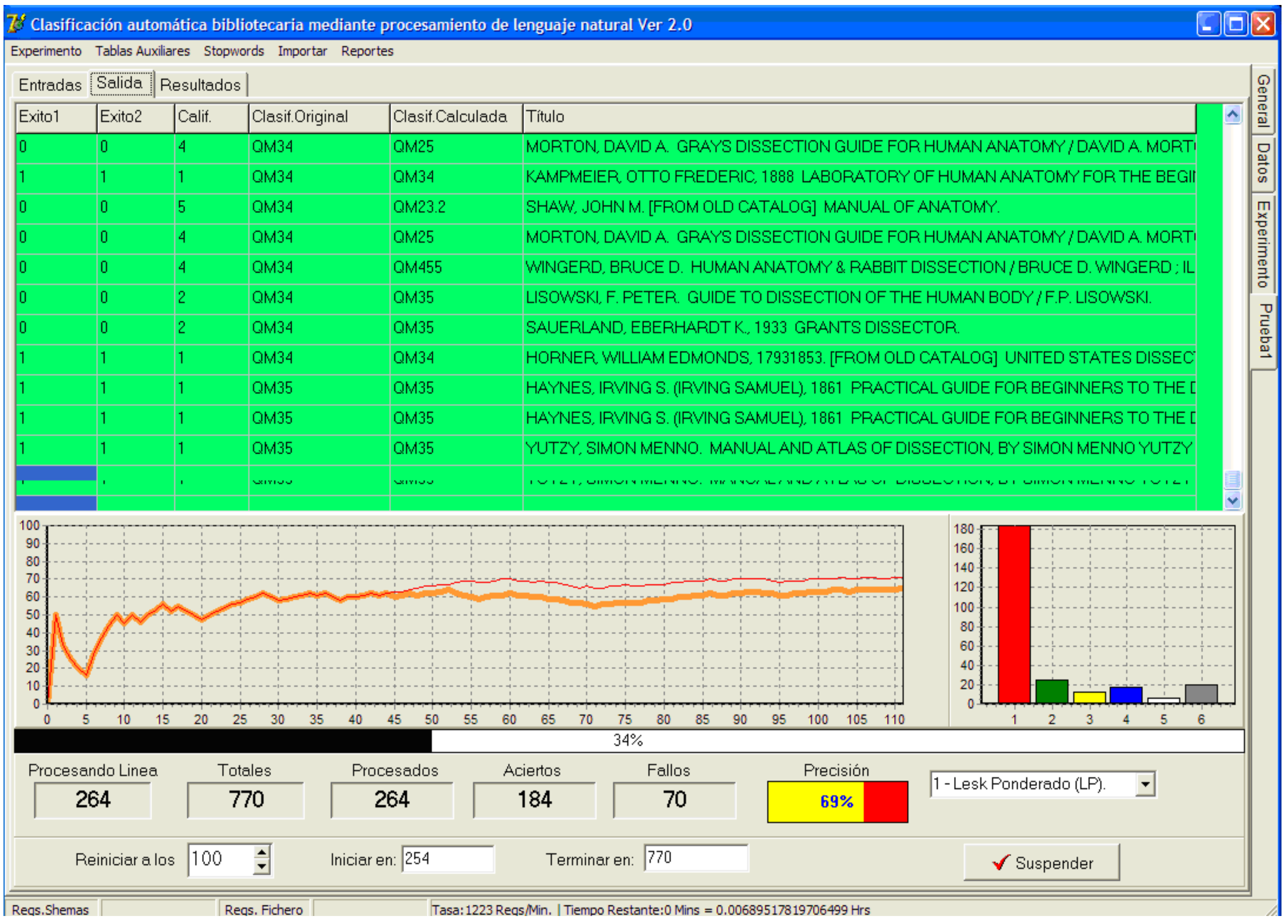
4 Evaluation and results

We experimented with class Q (Sciences) from the Library of Congress (LCC). The Q class comprises the following subclasses: QA: Math, QB: Astronomy, QC: Physics, QD: Chemistry, QE: Geology, QH: Natural History, QK: Botany, QL: Zoology, QM: Human Anatomy, QP: Physiology, QR: Microbiology.

We performed 11 experiments, each of them trained with 80% of the records, and evaluated with 20% of them from each branch using the previously described algorithms, namely:

Table 7: Example of similarity measures of title "PRACTICAL MATHEMATICS" with other titles from all classes

Similarity	STM	Length	Sub-Class	Title
0.00	0	1	QA37	BIOMATHEMATICS
0.50	**1**	**2**	**QA39**	**MATHEMATICS USE**
0.33	1	3	QA303	PURE MATHEMATICS COURSE
0.17	1	6	QA5	MATHEMATICS JA GLENN GH LITTLER DICTIONARY
0.20	1	5	QA501	PRACTICAL DESCRIPTIVE GEOMETRY, GRANT
0.25	1	4	QA76	INTERNATIONAL JOURNAL COMPUTER MATHEMATICS
0.20	1	5	QA76.58	PRACTICAL PARALLEL COMPUTING STEPHEN MORSE
0.17	1	6	QA37	MATHEMATICS MEASUREMENTS MERRILL RASSWEILER MERLE HARRIS
0.14	1	7	QA37.2	APPLIED FINITE MATHEMATICS RICHARD COPPINS PAUL UMBERGER
0.14	1	7	QA37.2	EUCLIDEAN SPACES PREPARED LINEAR MATHEMATICS COURSE TEAM
0.17	1	6	QA37.2	FOUNDATIONS MATHEMATICS KENNETH BERNARD HENRY WELLENZOHN
0.17	1	6	QA37.2	MATHEMATICS APPLICATIONS LAURENCE HOFFMANN MICHAEL ORKIN



Exito1	Exito2	Calif.	Clasif.Original	Clasif.Calculada	Titulo
0	0	4	QM34	QM25	MORTON, DAVID A. GRAYS DISSECTION GUIDE FOR HUMAN ANATOMY / DAVID A. MORT
1	1	1	QM34	QM34	KAMPMEIER, OTTO FREDERIC, 1888 LABORATORY OF HUMAN ANATOMY FOR THE BEGI
0	0	5	QM34	QM23.2	SHAW, JOHN M. [FROM OLD CATALOG] MANUAL OF ANATOMY.
0	0	4	QM34	QM25	MORTON, DAVID A. GRAYS DISSECTION GUIDE FOR HUMAN ANATOMY / DAVID A. MORT
0	0	4	QM34	QM455	WINGERD, BRUCE D. HUMAN ANATOMY & RABBIT DISSECTION / BRUCE D. WINGERD ; IL
0	0	2	QM34	QM35	LISOWSKI, F. PETER. GUIDE TO DISSECTION OF THE HUMAN BODY / F.P. LISOWSKI.
0	0	2	QM34	QM35	SAUERLAND, EBERHARDT K., 1933 GRANTS DISSECTOR.
1	1	1	QM34	QM34	HORNER, WILLIAM EDMONDS, 17931853. [FROM OLD CATALOG] UNITED STATES DISSEC
1	1	1	QM35	QM35	HAYNES, IRVING S. (IRVING SAMUEL), 1861 PRACTICAL GUIDE FOR BEGINNERS TO THE D
1	1	1	QM35	QM35	HAYNES, IRVING S. (IRVING SAMUEL), 1861 PRACTICAL GUIDE FOR BEGINNERS TO THE D
1	1	1	QM35	QM35	YUTZY, SIMON MENNO. MANUAL AND ATLAS OF DISSECTION, BY SIMON MENNO YUTZY

34%

Procesando Linea	Totales	Procesados	Aciertos	Fallos	Precisión
264	770	264	184	70	69%

1 - Lesk Ponderado (LP).

Reiniciar a los: 100 Iniciar en: 254 Terminar en: 770

Suspender

Regs.Shemas | Regs. Fichero | Tasa:1223 Regs/Min. | Tiempo Restante:0 Mins = 0.00689517819706499 Hrs

Figure 2: Screenshot of the experiment.

0. Frequency Term Voting.
1. Weighted Term Frequency Voting.
2. Term Frequency weighted by TF·IDF.
3. Term Presence Discrimination.
4 and 4′. Title Classification using Logical-Combinatorial methods.

In the following sections we present the average performance of each experiment for each branch.

4.1 Learning rate (training and test sets are the same)

The corresponding data are shown in Table 8 and Table 9.

Table 8: Experimental data description (1)

Experiment	Training records	Subclasses	Keywords	Test records
0 to 3	515,721	8,243	1,454,615	515,721
4, 4′	8,837	402	28,398	8,387

Table 9: Experimental evaluation (1)

Algorithm	**0**	**1**	**2**	**3**	**4**	**4’**
Uncovered	0	228	0	5,35	0	0
Covered	515,721	515,493	515,721	510,286	8,837	8,837
Success	178,654	433,861	177,945	396,689	8,214	7,822
Failure	337,067	81,860	337,776	119,032	623	1,015
Precision	34.64%	84.16%	34.50%	77.74%	**92.95%**	88.51%

4.2 Evaluation for unseen titles (training 80%, test 20%)

The corresponding data are shown in Table 10 and Table 11.

Table 10: Experimental data description (2)

Training records	Subclasses	Keywords	Test records
489,726	8,377	1,441,220	122,431

Coverage was 100% for all tests.

Table 11: Experimental evaluation (2)

Algorithm	**0**	**1**	**2**	**3**	**4**	**4′**
Success	32,869	41,763	32,507	**42,305**	41,537	30,223
Failure	84,222	75,328	84,584	74,786	75,554	86,868
Recall	28.07%	35.67%	27.76%	**36.13%**	35.47%	25.81%

4.2.1 Evaluation up to decimal point

In this section a less strict evaluation is presented. A complete classification would be QA237.6, being the *.6* part more specific. If the decimal part is not considered, QA237.13 would be correct as well.

Table 12: Experimental evaluation (3)

Algorithm	0	1	2	3
Success	39,482	48,857	39,023	48,274
Failure	77,609	68,234	78,068	68,817
Recall	33.72%	**41.73%**	33.33%	41.23%

We did not perform this test for algorithm 4 and 4′.

4.2.2 Evaluation by position

In Table 13, it can be seen that the top 5 suggestions given by our system comprise more than 62% of the correct classification for Algorithm 1. This suggest that this method could be used for suggesting classifications for a librarian reducing the number of classes he or she has to consider for classifying a title.

5 Conclusions and future work

We experimented with the branch Q of the LCC database, which comprises 612,157 titles in several languages. We achieved to classify books using only their title, when using the LCC classification up to its first decimal point (*v.gr. QA237.15*). Our evaluation was based on 8,377 subclasses of class Q, separately for each main branch (QA, QB, etc.). The best two algorithms proved to be Algorithms 1 and 3. These are simple algorithms that run approximately in half the time that the basic logical-combinatorial algorithms took. Algorithm 1 presents the correct answer within the top 5 answers with slightly more than a 60% precision. The highest learning rate was from the ALVOT algorithm (Algorithms 4 and 4′) that achieve more than 92.95% accuracy. For the unseen titles test we obtained 37.74% accuracy using the ALVOT algorithm (the version used in Algorithm 4) based on logical-combinatorial methods.

These experiments show that the title of a book is contributing at least one third of the information for its correct classification, as can be shown by comparing our results with those using more resources such as the table of contents, the complete text contents, MARC and LCSH (for example 36 with regard to 90) for sub-class comparison.

In the future we plan to explore more developed NLP methods for improving the performance of classification based only on the title of the work. Among other methods, we plan on involving thesauri and stemming, as well as using more sophisticated algorithms within the logical-combinatorial approach.

Acknowledgements

The work was done under partial support of the Mexican Government (CONACYT grants 50206-H and 60557, SIP-IPN 20091587, and PIFI-IPN) and the Japanese Government. The second author is a JSPS fellow, and the second, third, and fourth authors are National Researchers of Mexico (SNI).

Table 13: Evaluation by position

	0	1	2	3	4	4’
	VFT	VFTP	VFTP-TF·IDF	DPT	VT-MLC	VT-MLC
1	28.07%	35.67%	27.76%	36.13%	35.47%	25.81%
2	12.37%	12.28%	12.08%	10.15%	8.20%	9.71%
3	7.60%	6.88%	7.51%	3.83%	4.62%	6.01%
4	5.51%	4.65%	5.42%	2.05%	3.31%	4.49%
5	4.18%	3.40%	4.11%	1.19%	2.51%	3.54%
Total	57.73%	**62.88%**	56.88%	53.35%	54.11%	49.56%

References

[1] Manning, C. Shütze, H, *Foundations of statistical natural language Processing*, MIT Press, ISBN 0262133601, Cambridge, May, 620 p., 1999.

[2] Kwan, Yi, Challenges in automated classification using library classification schemes, *Proceedings of the 97 Information Technology with Audiovisual and Multimedia and National Libraries IFLA 2006*, Seoul, Korea, 2006.

[3] Frank, Ebie, Gordon W. Paynter, Predicting Library of Congress classifications from Library of Congress subject headings, *Journal of the American Society for Information Science and Technology*, Volume 55, Issue 3 , pp 214-227, 2004.

[4] Betts, Tom, Maria Milosavljevic, and Jon Oberlander. The utility of information extraction in the classification of books. In *Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007)*, Rome, Italy, 2004.

[5] Larson, Ray R., Experiments in automatic library of congress classification, *Journal of the American Society for Information Science and Technology*, Volume 43, Issur 2, pp 130-148, January 1999.

[6] Matthis, Raimund, *Adopting the library of congress classification system. A manual methods and techniques for application or conversion*, New York: R. R. Bowker, USA, 209 p.

[7] Savage Helen, Droste Kathleen D., Runchock Rita, *Class Q science: Library of Congress classification*

schedules combined with additions and changes through 1987, Library of Congress. Subject Cataloguing Division, Detroit, Michigan: Gale research: Book Tower, USA, 862 p.

[8] Immroth, John Phillip, *A guide to Library of Congress classification*, Rochester libraries unlimited, USA, 356 p.

[9] *Library of Congress/Decimal Classification Office, Guide to use of dewey decimal classification. Based on the practice of the practice of the decimal classification office at the library of congress*, Forest, New york, USA, 133 p.

[10] Furrie, Betty, *Conociendo MARC bibliográfico: catalogación legible por máquina*, Rojas Eberhard, Bogotá, Colombia, 30 pp.

[11] A.N. Dmitriev, Yu.I. Zhuravliov; F.P. Kredelev, *Acerca de los principios matemáticos de la clasificación de objetos y fenómenos*, Novosibirsk, Rusia, Tomo 7, pp. 3-15

[12] Ruiz Shulcloper, José; Guzmán Arenas, Adolfo; Martínez Trinidad J. Francisco, *Enfoque lógico combinatorio al reconocimiento de patrones, selección de variables y clasificación supervisada*, IPN, Mexico, pp. 69–75.

[13] Taylor, Arlene G., *The Organization of Information*, page 77. Libraries Unlimited, 2004

Automatic Identification of Lexical Units

Vidas Daudaravicius
Vytautas Magnus University, Faculty of Informatics
Vileikos 8, Kaunas, Lithuania
E-mail: vidas@donelaitis.vdu.lt

Keywords: lexical unit, lexical unit identification, token/type ratio, dice score, corpus size, average minimum law

Received: March 3, 2009

Lexical unit is a word or collocation. Extracting lexical knowledge is an essential and difficult task in NLP. The methods of extracting of lexical units are discussed. We present a method for the identification of lexical boundaries. The problem of necessity of large corpora for training is discussed. The advantage of identification of lexical boundaries within a text over traditional window method or full parsing approach allows to reduce human judgment significantly.

Povzetek: Opisana je metoda za avtomatično identifikacijo leksikalnih enot.

1 Introduction

Identification of a lexical unit is an important problem in many natural language processing tasks and refers to the process of extracting of meaningful word chains. The Lexical unit is a fuzzy term embracing a great variety of notions. The definition of the lexical unit differs according to the researcherŠs interests and standpoint. It also depends on the methods of extraction that provide researchers with lists of lexical items. Most lexical units are usually single words or constructed as binary items consisting of a node and its collocates found within a previously selected span. The lexical unit can be: (1) a single word, (2) the habitual co–occurrence of two words and (3) also a frequent recurrent uninterrupted string of words. Second and third notion refers to the definition of a collocation or a multi–word unit. It is common to consider a single word as a lexical unit. A big variety of the definition of the collocation is presented in Violeta Seretan work [12]. Fragments of corpus or strings of words consisting of collocating words are called collocational chains [7]. For many years the final agreed definition of the collocation is not made. Many syntactical, statistical and hybrid methods have been proposed for collocation extraction [13], [1], [5], [4]. In [10], it is shown that MWEs are far more diverse and interesting than is standardly appreciated. MWEs constitute a key problem that must be resolved in order for linguistically precise NLP to succeed. Although traditionally seen as a language independent task, collocation extraction relies nowadays more and more on the linguistic preprocessing of texts prior to the application of statistical measures. In [14] it is provided a language-oriented review of the existing extraction work.

In our work we compare Dice and Gravity Counts methods for the identification of lexical units by applying them under the same conditions. The definition of what is a Lexical Unit in a linguistic sence is not discussed in this paper. New presented technique extracts collocations like 'in the' that do not have meaning and have functional purpose. A question of keeping such collocations as lexical units is left open. At the same time, it is interesting to see that the frequency lists of such lexical units for English and Lithuanian (memeber of Balto-Slavonic language group) are now comparable.

2 Extracting vs. abstracting

Most of the collocation definitions refer to the collocation, which is constructed in an abstracting way. The collocations are not gathered directly from the text but rather constructed using syntactic and statistical information. The abstracted collocation is constructed using statistical information extracted from the corpus. The extraction of statistical information from a corpus is only the first step for constructing collocations. The process of constructing the collocation is called as a collocation extraction in many research works. In this paper we make difference between the extraction of collocation and the abstraction of colocation. The major difference between abstracting and extracting of collocation is the use of lexical boudaries. The extractive technique for the identification of lexical units takes a linear approach of consecutive counts of words in a text and of all the texts in a corpus. Thus calculations of combinability are applied to the continuous chain of words. The first step is to detect the strength of combinability for pairs of words in the corpus, the second step is to detect the boundaries of the lexical units. The Extractive technique marks lexical boundaries in the text and a word or a word chain between these boundaries is a lexical unit. A clear idea about the boundaries of the lexical units allows to determine the exact size of a corpus lexicon. Abstractive technique uses a statistical information extracted from a corpus and a definition of a threshold for a good lexical unit. The threshold in many cases is frequency. Tagging and parsing are also used for filtering out invalid results [8]. Both, abstractive and extractive, techniques use associative measures to evaluate the combinability of two items. A new technique is presented to solve the problem of identification of (uni-)multiword lexical units without any linguistic knowlegde when full automatization is necessary. Extractive technique is very practical, easy to implement, and could improve quality of results in many IR and IE tasks. Nevertheless the results can be used for lexicografical tasks.

3 Combinability measures

Two different statistical calculations of collocability counts are applied (Dice and Gravity Counts)in this work. A good overview of combinability methods is presented in [3].

3.1 Dice score

The Dice coefficient can be used to calculate the co-occurrence of words or word groups. This ratio is used, for instance, in the collocation compiler XTract [11] and in the lexicon extraction system Champollion [6]. It is defined as follows [11]:

$$Dice(x,y) = \frac{2 * f(x,y)}{f(x) + f(y)}$$

$f(x,y)$ being the frequency of co-occurrence of x and y, and $f(x)$ and $f(y)$ the frequencies of occurrence of x and y anywhere in the text. If x and y tend to occur in conjunction, their Dice score will be high. The logarithm is added in order to discern small numbers. Thus the formula is slightly modified. The combinability of the each pair of words using this method was measured on the basis of the formula:

$$Dice'(x,y) = log_2\left(\frac{2 * f(x,y)}{f(x) + f(y)}\right)$$

The human have to set the level of collocability manualy. We set the level of collocability at the Dice minus 8 in our experiments. This decision was based on the shape of the curve found in [3].

3.2 Gravity counts

Gravity Counts are based on the evaluation of the combinability of two words in a text that takes into account a variety of frequency features, such as individual frequencies of words, the frequency of pairs of words and the number of types in the selected span. Token/type ratio is used slightly different. Usually this ratio is used for the whole document. The difference is that the token/type ratio is calculated not for a document or a corpus but for a word within a selected span only. In our experiments we used the span equal to 1. The expression of Gravity Counts is as follows:

$$G(x,y) = log\left(\frac{f(x,y) * n(x)}{f(x)}\right) + $$
$$+ log\left(\frac{f(x,y) * n'(y)}{f(y)}\right)$$

(x,y) is the frequency of the pair of words x and y in the corpus; $n(x)$ is a number of types to the right of x; $f(x)$ is the frequency of x in the corpus; $n'(y)$ is the number of types to the left of y; $f(y)$ is the frequency of y in the corpus. We set the level of collocability at the Gravity Counts 1 in our experiments. This decision was based on the shape of the curve found in [3].

4 Identifying the boundaries of lexical units

There were attempts to extract recurrent uninterrupted strings of unlemmatized word-forms [7]. The chains were identified purely on the ground of row frequency and consisted of chains up to five words in length. However, the applied method did not reveal whether the extracted chains are made of collocating words. In our case, the detection of the boundaries of a lexical unit is based on a full text approach. The idea behind this approach is that the corpus is used as a very long chain of words to calculate the combinability of adjacent word pairs. The counts starts with the first and ends with the last word of the corpus. Thus, the corpus is seen as a changing curve of the lexical combinability. Peaks, appearing above the point of a selected value are taken as collocability points that can form lexical units (see Figure 1 for an example of a sentence). Using a text as the basis for the identification of lexical units with the help of the collocability points allows detecting the boundaries of each lexical unit. A lexical unit is defined as a segment of text where the combinability of constituent adjacent word pairs is above the arbitrarily chosen point of collocability. The lower combinability of word pairs preceding and following the segment marks the boundaries of a lexical unit. The list of all such segments from the corpus is the list of its lexical units. Moreover, we introduce two new additional definitions of the boundary of lexical unit. We call them absolute minimum and average minimum laws.

4.1 Average minimum law

In addition to the collocability requirement the average minimum law can be applied. This law is applied to the three adjacent collocability points. The law can be expressed as follows: if $\frac{x_{-2}+x_0}{2} > x_{-1}$ then the boundary of a lexical unit is set at x_{-1} . The boundary of a lexical item is set, if the average of values of collocability points on both sides are higher. This law allows making additional boundaries of lexical units when collocability points are set. The identified lexical units are shorter and more clearcut (see Figure 2 for an example of a sentence).

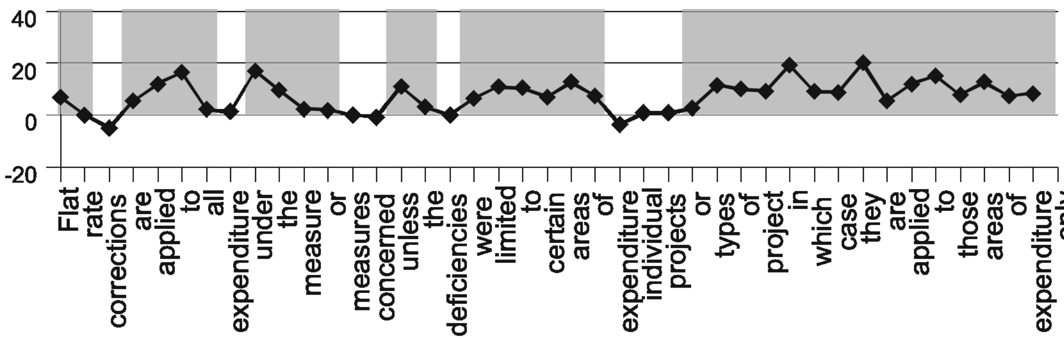


Figure 1: Identified lexical units of an example sentence, combinability values and collocability level at value 1. / Flat rate / corrections / are applied to all expenditure / under / the measure or measures / concerned / unless the deficiencies / were limited to certain areas of expenditure / individual / projects / or types of project in which case they are applied to those areas of expenditure only/

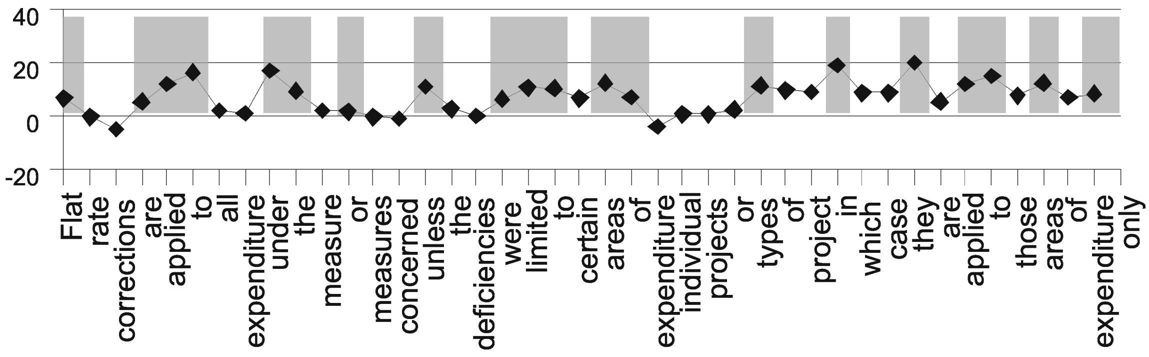


Figure 2: Identified lexical units of an example sentence, combinability values, collocability level at value 1 and average minimum law applied. / Flat rate / corrections / are applied to / all / expenditure / under the measure / or measures / concerned / unless the / deficiencies / were limited to certain / areas of expenditure / individual / projects / or / types of / project / in which / case / they are / applied to those / areas of / expenditure only /

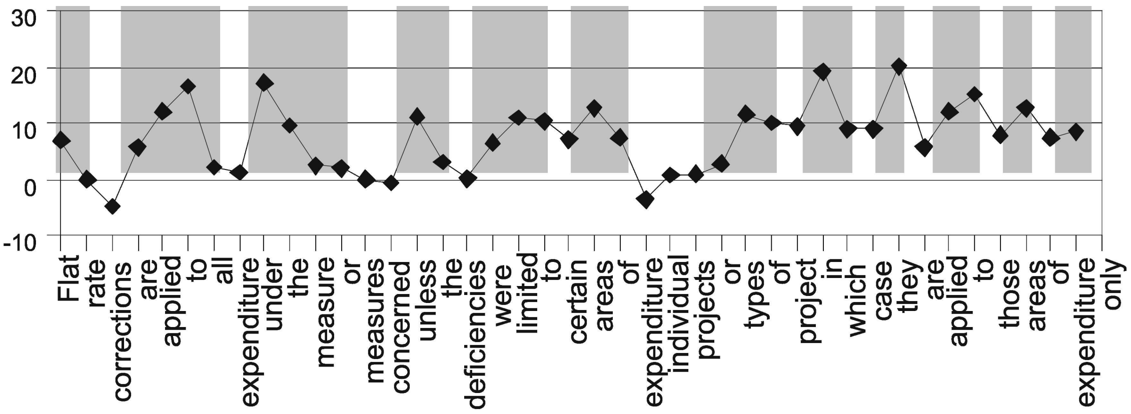


Figure 3: Identified lexical units of an example sentence, combinability values, collocability level at value 1 and absolute minimum law applied. / Flat rate / corrections / are applied to all expenditure / under the measure or measures / concerned / unless the / deficiencies / were limited to certain / areas of expenditure / individual / projects / or / types of project / in which case / they are / applied to those / areas of / expenditure only /

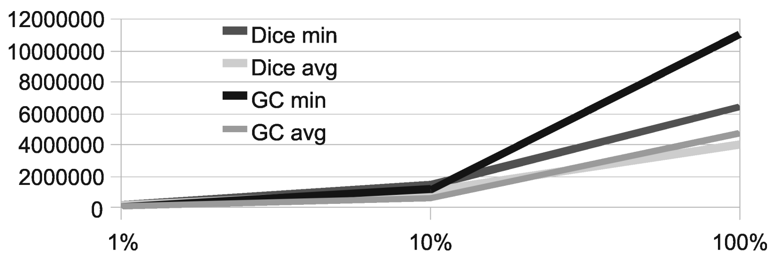


Figure 4: The number of lexical units (types) in the selected corpus (*x-axis has logarithmic scale*)

Gravity Counts average minimum law			Dice average minimum law		
100%	10%	1%	100%	10%	1%
of the	the	and	and	and	and
in the	and	the	the	the	the
and	in the	of	of the	of the	of
to the	the	of the	of	of	of the
on the	of	in the	in the	in the	in the
the	to the	to	in	in	to
at the	in	in	to	to	in
for the	on the	a	a	a	a
and the	to	for	to the	to the	to the
to be	or	’s	’s	on the	that
of	’s	to the	on the	’s	for
or	for	that	for	for	on the
in	a	is	that	that	to be
by the	that	by	or	or	and the
of a	and the	was	for the	for the	’s
from the	is	with	is	and the	or
with the	for the	or	and the	is	for the
that	with	on	to be	to be	was
to	at the	on the	was	at the	with
it was	by	from	at the	with	is

Table 1: The top 20 lexical units for different size of corpus and scores

100%	10%	1%
she might	she might	she might
have been	have been the	have been
the	headmistress	the
headmistress	of a	headmistress
of a	certain type of	of a
certain	girls ’	certain type of
type of	school	girls ’
girls ’	, now	school
school	almost extinct	, now
, now almost extinct	, or a	almost extinct
, or a	mother superior	, or a
mother superior	in an	mother superior
in an	enclosed order	in an
enclosed	.	enclosed order
order		.
.		
at any rate	at any rate	at any rate
there could be	there could be	there could be
no doubt	no doubt	no doubt
that she had	that she had	that she had found
found	found	the
the	the	temptation
temptation	temptation	of the
of the	of the	flesh resistible
flesh resistible	flesh resistible	.
.	.	

Table 2: The boundaries of lexical units identified by Dice

4.2 Absolute minimum law

In addition to the collocability requirement the average minimum law can be applied. This law is applied to the three adjacent collocability points. The law can be expressed as follows: if $x_{-2} > x_{-1} \wedge x_0 > x_{-1}$ then the boundary is set at x_{-1}. Informally, the boundary of a lexical item is set, if the values of collocability points on both sides are higher. The identified lexical units are wider compared to the average minimum low (see Figure 3 for an example of a sentence).

5 Experiments and results

We used whole British National Corpus for experiments. Three corpora sizes were used in experiments: whole, 10% and 1% of the corpus. We used row text without any tagged information.

5.1 Dictionary size

The number of lexical units identified in the corpus using the respective methods is presented in Figure 4. The number of lexical units extracted with the help of Dice and Gravity Counts scores using average minimum law is similar. The absolute minimum low yields to the different size of the dictionary. The result of the number of lexical units shows the trend line of possible total number of lexical units. We can expect maximum of about 5-6 million of lexical units in English using Dice score and average minimum law. This number is comparable to different languages, e. g., Lithuanian with rich morphology and almost free word order. In [9] the number of word types in corpus comparable to BNC is 1.8 million. In [8] the number of extracted collocations using similar method from Lithuanian

100%	10%	1%
she might have been the	she might have been the	she might have been the
headmistress	headmistress	headmistress
of a certain	of a certain	of a certain
type of	type of	type of
girls ’	girls ’	girls
school , now	school , now	’
almost	almost	school , now
extinct	extinct	almost
, or a	, or a mother	extinct
mother	superior	, or a
superior	in an	mother
in an	enclosed	superior
enclosed	order .	in an
order .		enclosed
		order
		.
at any rate	at any rate	at any rate
there could be	there could be	there
no doubt	no doubt	could be
that she had	that she had	no doubt
found the temptation	found the temptation	that she had
of the	of the	found the
flesh	flesh	temptation
resistible	resistible	of the
	.	flesh
		resistible
		.

Table 3: The boundaries of lexical units identified by Gravity Counts

corpus is 20 millions. We used new laws of minimum in our experiments. It is obvious that if the law of average minimum would be applied in [8] work then the number of collocations would drop to 6-7 millions or more for Lithuanian. Thus we are able to speak about the similar number of lexical units which could be applied for any language. For instance, the machine translation system ATLAS-II v.13 by Fujitsu has 5.44M terminological entries and about 1M to 1.5M general entries [2].

5.2 Top 20 lexical units

Another goal of our research is to discover which score is less sensitive to the corpus size. The size of corpus differs in applications. In [3] is shown that Mutual Information score heavily depends on the corpus size and it is very difficult to set the level of collocability. Dice and Gravity Counts scores do not consider corpus size. We performed several experiments to compare method dependability on the size of the corpus. We used Dice and Gravity Counts score together with the average minimum law on the different corpus sizes. We took 1%, 10% and full corpus of BNC. We built the dictionaries of lexical units and took top 20 lexical units for every case. The results are shown in Table 1. The list of top 20 lexical units identified using Dice score almost does not change. While the list of lexical units identified using Gravity Counts changes. This is sufficient to state that Dice score is stable, not sensitive to the corpus size and is reliable in many NLP applications. This statement is confirmed by the examples in Table 2 and Table 3. The same two sentences are taken and the boundaries of lexical units are identified. The law of average minimum is used. We can see that the identified boundaries of lexical units using Dice score do not change considerably. While in case of Gravity Counts the change of boundaries is observable often.

6 Conclusions

The numbers of lexical units in most languages is comparable and amounts to 6-7 millions. It should be applicable for the most of indoeuropean languages. The lexical unit is very important in NLP and is applied widely. But the notion of lexical unit is not clear and hard to define. We propose a definition of a lexical unit as a sequence of wordforms extracted from row text by using collocability feature and setting boundaries of lexical units. This approach is more clear compared to a widely used n-gram definition of a lexical unit. The boundaries are predictable and easier controlled compared to n-gram model. The result of setting lexical boundaries for the small and large corpora are stable using Dice score. Thus Dice score is reliable in many NLP applications. The average minimum law allows making additional boundaries of lexical units when collocability points are set. Identified lexical units are shorter and more clearcut. Human judgment on the boundaries of lexical unit is reduced considerably as the setting of collocability level is not so sensitive when the average minimum law is applied.

References

[1] Brigitte Orliac, Mike Dillinger (2003) Collocation extraction for machine translation, *MT Summit IX*, New Orleans, USA, 23-27 September 2003, pp.292-298

[2] Christian Boitet, Youcef Bey, Mutsuko Tomokio, Wenjie Cao, Hervé Blanchon (2006) IWSLT-06: experiments with commercial MT systems and lessons from subjective evaluations,ă *International Workshop on Spoken Language Translation: Evaluation Campaign on Spoken Language Translation [IWSLT 2006]*, November, 27-28, 2006, Kyoto, Japan, pp.23–30.

[3] Daudaravicius V., Marcinkeviciene R. (2004) Gravity Counts for the Boundaries of Collocations, *International Journal of Corpus Linguistics, 9(2)*, pp. 321Ű-348.

[4] Dekang Lin (1998) Extracting collocations from text corpora, *In First Workshop on Computational Terminology*, Montreal, 1998, pp. 57Ű-63.

[5] Gael Dias (2003) Multiword unit hybrid extraction, *In Proceedings of the ACL Workshop on Multiword Expressions*, Sapporo, Japan, pp. 41Ű-48.

[6] Marcinkeviciene R., Grigonyte G. (2005) Lexicogrammatical Patterns of Lithuanian Phrases, *The Second Baltic Conference on Human Language Technologies proceedings*, Tallinn, pp. 299Ű-305.

[7] Rimkute E., Daudaravicius V., Utka A. (2007) Morphological Annotation of the Lithuanian Corpus, *45th Annual Meeting of the Association for Computational Linguistics; Workshop Balto-Slavonic Natural Language Processing*, Praha, pp. 94–99.

[8] Smadja, F. (1993) Retrieving Collocations from Text: XTRACT, *Computational Linguistics, 19(1)*, pp. 143-177.

[9] Smadja, F., McKeown, K. R. and V. Hatzivassiloglou (1996) Translation Collocations for Bilingual Lexicons: A Statistical Approach, *Computational Linguistics, 22(1)*, pp. 1-38.

[10] Sag, Ivan A., Timothy Baldwin, Francis Bond, Ann Copestake and Dan Flickinger (2002) Multiword Expressions: A Pain in the Neck for NLP, *In Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2002)*, Mexico City, Mexico, pp. 1–15

[11] Stubbs, M. (2002) Two quantitative methods of studying phraseology in English, *International Journal of Corpus Linguistics, 7(2)*, pp. 215-244.

[12] Violeta Seretan (2008) Collocation Extraction Based on Syntactic Parsing, *Ph.D. thesis*, University of Geneva.

[13] Violeta Seretan and Eric Wehrli (2006) Accurate collocation extraction using a multilingual parser, *In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*, Sydney, Australia, pp. 953-Ű960.

[14] Violeta Seretan and Eric Wehrli (2006) Multilingual collocation extraction: Issues and solutions. *In Proceedings of COLING/ACL Workshop on Multilingual Language Resources and Interoperability*, 2006, Sydney, Australia, pp. 40-Ű49.

Appendix 1. *Risk* Lexical units extracted using Gravity Counts and average minimum law

1% of BNC (1 million words)	
Lexical unit	Frequency
increase the risk of	1
risk	51
risk of	8
the risk	9
the risk of	5
the risk of another	1

Appendix 2. *Risk* Lexical units extracted using Dice and average minimum law

1% of BNC (1 million words)	
Lexical unit	Frequency
at risk	9
calculated risk	1
currency risk	1
environmental risk	1
health risk	1
particular risk	2
primary risk	1
real risk	1
reducing risk	1
risk	13
risk being	1
risk being disappointed	1
risk being considered	1
risk losing	2
risk undermining	1
risk using	1
risk of	7
risk than being overweight	1
risk than an asset	1
risk factor	1
risk slipping	1
risk arguing	1
risk assessment	1
risk her	1
risk her rage	1
risk damage	1
risk missing	1
risk cream	1
risk element	1
serious potential risk	1
serious risk	1
safety risk	1
the risk	5
the risk of	6
the risk compared with	1
the great risk of	1
were at risk	1

Finding Maximal Sequential Patterns in Text Document Collections and Single Documents

René Arnulfo García-Hernández
Autonomous University of the State of Mexico
Tianguistenco Professional Academic Unit
Paraje el Tejocote, San Pedro Tlaltizapan, Estado de México
E-mail: renearnulfo@hotmail.com, http://scfi.uaemex.mx/~ragarcia/

J. Fco. Martínez-Trinidad and J. Ariel Carrasco-Ochoa
National Institute of Astrophysics, Optics and Electronics
E-mail: fmartine@inaoep.mx, ariel@inaoep.mx

Keywords: text mining, maximal sequential patterns

Received: April 1, 2009

In this paper, two algorithms for discovering all the Maximal Sequential Patterns (MSP) in a document collection and in a single document are presented. The proposed algorithms follow the "pattern-growth strategy" where small frequent sequences are found first with the goal of growing them to obtain MSP. Our algorithms process the documents in an incremental way avoiding re-computing all the MSP when new documents are added. Experiments showing the performance of our algorithms and comparing against GSP, DELISP, GenPrefixSpan and cSPADE algorithms over public standard databases are also presented.

Povzetek: Predstavljena sta dva algoritma za iskanje najdaljših zaporedij v besedilu.

1 Introduction

Frequent pattern mining is a task into the datamining area that has been intensively studied in the last years [Jiawei Han et al. 2007]. Frequent patterns are itemsets, subsequences, or substructures that appear in a data set with frequency no less than a user-specified threshold. Frequent pattern mining plays an important role in mining associations, correlations, finding interesting relationships among data, data indexing, classification, clustering, and other data mining tasks as well. Besides, frequent patterns are useful for solving more complex problems of data analysis. Therefore, frequent pattern mining has become an important area in data mining research.

Frequent pattern mining was first proposed by [Agrawal et al. 1993] for market basket analysis finding associations between the different items that customers place in their "shopping baskets". Since this first proposal there have been many research publications proposing efficient mining algorithms, most of them, for mining frequent patterns in transactional databases.

Mining frequent patterns in document databases is a problem which has been less studied. Sequential pattern mining in document databases has the goal of finding all the subsequences that are contained at least β times in a collection of documents or in a single document, where β is a user-specified support threshold. This discovered set of frequent sequences contains the maximal frequent sequences which are not a subsequence of any other frequent sequence (from now on we will use the term Maximal Sequential Patterns, MSP), that is, the MSPs are a compact representation of the whole set of frequent sequences. Therefore, in the same way as occurs in transactional databases, the sequential pattern mining in document databases plays an important role, because it allows identifying valid, novel, potentially useful and ultimately understandable patterns. In this paper, we will focus in the extraction of this kind of patterns from textual or text document databases. Since maximal sequential patterns can be extracted from documents independently of the language without losing their sequential nature they can be used to solve more complex problems (all of them related to text mining) as question answering [Denicia-Carral et al. 2006; Juárez-González et al. 2007; Aceves-Pérez et al. 2007], authorship attribution [Coyotl-Morales et al. 2006], automatic text summarization [Ledeneva et al. 2008], document clustering [Hernandez-Reyes et al. 2006], and extraction of hyponyms [Ortega-Mendoza et al. 2007], among others.

In this article, we present two pattern-growth based algorithms, DIMASP-C and DIMASP-D, to **Di**scover all the **Ma**ximal **S**equential **P**atterns in a document collection and in a single document respectively. The rest of this article is organized in four sections: (2) related work, (3) problem definition, (4) algorithms for mining frequent patterns in documents (in this section experimental results are also given) and (5) concluding remarks.

2 Related work

Most of the algorithms for sequential pattern mining have been developed for vertical databases, this is, databases with short sequences but with a large amount of sequences. A document database can be considered as horizontal because it could have long sequences. Therefore, most of the algorithms for sequential pattern mining are not efficient for mining a document database. Furthermore, most of the sequential pattern mining approaches assume a short alphabet; that is, the set of different items in the database. Thus, the characteristics of textual patterns make the problem intractable for most of the *a priori*-like candidate-generation-and-test approaches. For example, if the longest MSP has a length of 100 items then GSP [Srikant et al., 1996] will generate $\sum_{i=1}^{100}\binom{100}{i}\approx 10^{30}$ candidate sequences where each one must be tested over the DB in order to verify its frequency. This is the cost of candidate generation, no matter what implementation technique would be applied. For the candidate generation step, GSP generates candidate sequences of size $k+1$ by joining two frequent sequences of size k when the prefix k-1 of one sequence is equal to the suffix k-1 of another one. Then a candidate sequence is pruned if it is non-frequent. Even though, GSP reduces the number of candidate sequences, it still is inefficient for mining long sequences.

As related work, we can mention those pattern-growth algorithms that speed up the sequential pattern mining [Jiawei Han et al. 2000; Antunes et al. 2003; Jian Pei et al. 2004; Lin et al. 2005] when there are long sequences. According to the empirical performance evaluations of pattern-growth algorithms like PrefixSpan [Jian Pei et al. 2004], GenPrefixSpan [Antunes et al. 2003], cSPADE [Zaki 2000], and DELISP[Lin et al. 2005], they outperform GSP specially when the database contains long sequences, therefore in this paper we will use them in our experiments. The basic idea in these algorithms is to avoid the cost of the candidate generation step and to focus the search on sub-databases generating projected databases. An α-projected database is the set of subsequences in the database that are suffixes of the sequences with prefix α. In each step, the algorithm looks for frequent sequences with prefix α in the corresponding projected database. In this sense, pattern-growth methods try to find the sequential patterns more directly, growing frequent sequences, beginning with sequences of size one. Even though, these methods are faster than apriori-like methods, some of them were designed to find all the frequent sequences, instead of only finding the MSP. Furthermore, none of them is incremental.

In this paper, we present two pattern-growth based algorithms, DIMASP-C and DIMASP-D, to **Di**scover all the **Ma**ximal **S**equential **P**atterns in a document collection and in a single document respectively. First, DIMASP algorithms build a novel data structure from the document database which is relatively easy to extract. Once DIMASP algorithms have built the data structure, they can discover all the MSP according to a threshold specified by the user.

In contrast with PrefixSpan, GenPrefixSpan and DELISP; if the user specify a new threshold our algorithms avoid rebuilding the data structure for mining with the new threshold. In addition, when the document database is increased, DIMASP algorithms update the last discovered MSP by processing only the new added documents.

3 Problem definition

The problem of finding patterns in documents can be formulated following the same idea as in transactional databases, i.e., assuming that each document of the collection is a transaction in the database, in this way, a sequence of items in a document will be a pattern in the collection if it appears in a certain number of documents. We have denominated to this formulation as the problem of finding all the maximal sequential patterns in a document collection.

3.1 Finding all the MSP in a document collection

A *sequence S,* denoted by $<s_1,s_2,...,s_k>$, is an ordered list of k elements called *items*. The number of elements in a sequence S is called the *length* of the sequence denoted by $|S|$. A *k-sequence* denotes a sequence of length k. Let $P=<p_1p_2...p_n>$ and $S=<s_1s_2...s_m>$ be sequences, P is a *subsequence* of S, denoted $P\subseteq S$, if there exists an integer $i\geq 1$, such that $p_1=s_i$, $p_2=s_{i+1}$, $p_3=s_{i+2}$,…,$p_n=s_{i+(n-1)}$. The *frequency* of a sequence S, denoted by S_f or $<s_1,s_2,...,s_n>_f$, is the number of documents in the collection where S is a subsequence. A sequence S is *β-frequent* in the collection if $S_f\geq\beta$, a β-frequent sequence is also called a *sequential pattern in the document collection*. A sequential pattern S is *maximal* if S is not a subsequence of any other sequential pattern in the collection.

Given a document collection, the problem consists in finding all the maximal sequential patterns in the document collection.

Another formulation of the problem is finding patterns in a single document. At first glance, this problem could be solved by dividing the document into sections or paragraphs, and by applying algorithms for finding all the MSP in a document collection. However, the result would depend on the way the document was divided.

In addition, a sequence of items will be a pattern in the document if it appears in many sections or paragraphs without taking account the number of times the sequence appears inside each section or paragraph. This situation makes the problem different, therefore we will consider that a sequence of items in a document will be a pattern if it is frequent or appears many times inside the whole document. We have denominated to this formulation as the problem of finding all the maximal sequential patterns in a single document.

3.2 Finding all the MSP in a single document

Following the notation used in the section 3.1. Let $X \subseteq S$ and $Y \subseteq S$ then X and Y are *mutually excluded sequences* if X and Y do not share items i.e., if ($x_n=s_i$ and $y_1=s_j$) or ($y_n=s_i$ and $x_1=s_j$) then $i<j$. A sequence S is *β-frequent* in a document T, if it is contained at least β times in T in a mutually excluded way. *A β-frequent* sequence is also called a *sequential pattern in a document*. A sequential pattern S is *maximal* if S is not a subsequence of any other sequential pattern in the document.

Given a document, the problem consists in finding all the maximal sequential patterns in the document.

4 Algorithms for mining sequential patterns in documents

In this section, two algorithms one for mining maximal sequential patterns in a document collection (DIMASP-C) and another for mining maximal sequential patterns in a single document (DIMASP-D), are introduced. Both of them build a data structure containing all the different pairs of contiguous words in the document or in the collection and the relations among them. Then the maximal sequential patterns are searched into this structure, following the pattern-growth strategy.

4.1 DIMASP-C

The basic idea of DIMASP-C consists in finding all the sequential patterns in a data structure, which is built from the document database (DDB). The data structure stores all the different pairs of contiguous words that appear in the documents, without losing their sequential order. Given a threshold β specified by the user, DIMASP-C reviews if a pair is β-frequent. In this case, DIMASP-C grows the sequence in order to determine all the possible maximal sequential patterns containing such pair as a prefix. A possible maximal sequential pattern (PMSP) will be a maximal sequential pattern (MSP) if it is not a subsequence of any previous MSP. This implies that all the stored MSP which are subsequence of the new PMSP must be deleted. The proposed algorithm is composed of three steps described as follows:

In the first step, for each different word (item) in the DDB, DIMASP-C assigns an integer number as identifier. Also, for each identifier, the frequency is stored, i.e., the number of documents where it appears. These identifiers are used in the algorithm instead of the words. Table 1 shows an example for a DDB containing 4 documents.

In the second step (Fig. 1), DIMASP-C builds a data structure from the DDB storing all the pairs of contiguous words $<w_i,w_{i+1}>$ that appear in a document and some additional information to preserve the sequential order. The data structure is an *array* which contains in each cell a pair of words $C=<w_i,w_{i+1}>$, the *frequency* of the pair (C_f), a Boolean *mark* and a list Δ of nodes δ where a *node* δ stores a document identifier ($\delta.Id$), an *index* ($\delta.Index$) of the cell where the pair appears in the array, a link ($\delta.NextDoc$) to maintain the list Δ and a link ($\delta.NextNode$) to preserve the sequential order of the pairs with respect to the document, where they appear. Therefore, the number of different documents presented in the list Δ is C_f. This step works as follows: for each pair of words $<w_i,w_{i+1}>$ in the document D_J, if $<w_i,w_{i+1}>$ does not appear in the *array,* it is added, and its *index* is gotten. In the position *index* of the array, add a node δ at the beginning of the list Δ. The added node δ has J as $\delta.Id$, *index* as $\delta.index$, $\delta.NextDoc$ is linked to the first node of the list Δ and $\delta.NextNode$ is linked to the next node δ corresponding to $<w_{i+1},w_{i+2}>$ of the document D_J. If the document identifier ($\delta.Id$) is new in the list Δ, then the frequency of the cell (C_f) is increased. In Fig. 2 the data structure built for the document database of table 1 is shown.

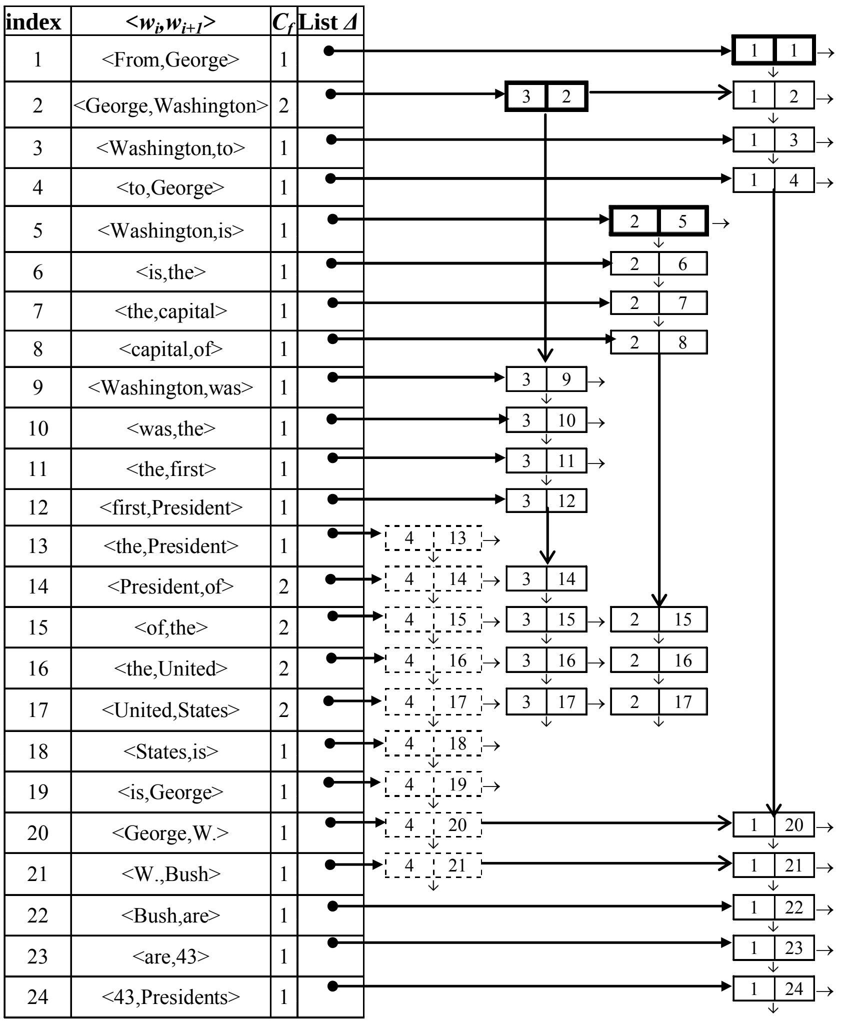
Table 1: Example of a document database and its identifier representation

D_J	**Document database**
1	From George Washington to George W. Bush are 43 Presidents
2	Washington is the capital of the United States
3	George Washington was the first President of the United States
4	*the President of the United States is George W. Bush*
	Document database (words by integer identifiers)
1	<1,2,3,4,2,5,6,7,8,9>
2	<3,10,11,12,13,11,14,15>
3	<2,3,16,11,17,18,13,11,14,15>
4	<11,18,13,11,14,15,10,2,5,6>

```
Step 2: Algorithm to construct the data structure from the DDB
Input: A document database (DDB)    Output: The Array
For all the documents $D_J \in DDB$ do
    Array ← Add a document ( $D_J$ ) to the array
end-for
Step 2.1: Algorithm to add a document
Input: A document $D_J$  Output: The Array
For all the pairs $\langle w_i, w_{i+1} \rangle \in D_J$ do
    $\delta_i$ ←Create a new Pair $\delta$
    $\delta_i$.Id ← J //Assign the document identifier to the node $\delta$
    index←Array[ $\langle w_i, w_{i+1} \rangle$ ] //Get the index of the cell
    $\delta_i$.index ← index   //Assign the index to the node $\delta$
    $\alpha$ ← Get the first node of the list $\Delta$
  If $\delta_i$.Id ≠ $\alpha$.Id then the document identifier is new to the list $\Delta$
      Increment $C_f$    //increment the frequency
      $\delta_i$.NextDoc ← $\alpha$ //link the node $\alpha$ at the beginning of list $\Delta$
      List $\Delta$← Add $\delta_i$ as the first node //link it at the beginning
    $\delta_{i-1}$.NextNode ← $\delta_i$ //do not lose the sequential order
  end-if
end-for
```

Figure 1: Steps 2 and 2.1 of DIMASP-C.

In the last step (Fig. 3), given a threshold β, DIMASP-C uses the constructed structure for mining all the maximal sequential patterns in the collection. For each pair of words stored in the structure, DIMASP-C verifies if this



index	$<w_i,w_{i+1}>$	C_f	List Δ
1	<From,George>	1	
2	<George,Washington>	2	
3	<Washington,to>	1	
4	<to,George>	1	
5	<Washington,is>	1	
6	<is,the>	1	
7	<the,capital>	1	
8	<capital,of>	1	
9	<Washington,was>	1	
10	<was,the>	1	
11	<the,first>	1	
12	<first,President>	1	
13	<the,President>	1	
14	<President,of>	2	
15	<of,the>	2	
16	<the,United>	2	
17	<United,States>	2	
18	<States,is>	1	
19	<is,George>	1	
20	<George,W.>	1	
21	<W.,Bush>	1	
22	<Bush,are>	1	
23	<are,43>	1	
24	<43,Presidents>	1	

Figure 2: Data structure built by DIMASP-C for the database of the table 1.

pair is β-frequent, in such case DIMASP-C, based on the structure, grows the pattern while its frequency (the number of documents where the pattern can grow) remains greater or equal than β. When a pattern cannot grow, it is a possible maximal sequential pattern (PMSP), and it is used to update the final maximal sequential pattern set. Since DIMASP-C starts finding 3-MSP or longer, then at the end, all the β-frequent pairs that were not used for any PMSP and all the β-frequent words that were not used for any β-frequent pair are added as maximal sequential patterns.

In order to be efficient, it is needed to reduce the number of comparisons when a PMSP is added to the MSP set (Fig. 4). For such reason, a k-MSP is stored according to its length k, it means, there is a k-MSP set for each k. In this way, before adding a k-PMSP as a k-MSP, the k-PMSP must not be in the k-MSP set and must not be subsequence of any longer k-MSP. When a PMSP is added, all its subsequences are eliminated.

For avoiding repeating all the work for discovering all the MSP when new documents are added to the database, DIMASP-C only preprocesses the part corresponding to these new documents. First the identifiers of these new documents are defined in step 1, then DIMASP-C would only use the step 2.1 (Fig. 1) to add them to the data structure. Finally, the step 3.1 (Fig. 3) is applied on the new documents using the old MSP set, to discover the new MSP set, for example, Fig. 2 shows with dotted lines the new part of the data structure when D_4 of table 1 is added as a new document. This strategy works only if the same β is used, however for a different β only the discovery step (step 3, Fig. 3) must be applied, without rebuilding the data structure.

The experiments were done using the well-known reuters-21578[1] document collection. After pruning 400 stop-words, this collection has 21578 documents with around 38,565 different words from 1.36 million words used in the whole collection. The average length of the documents was 63 words. In all the experiments the first 5000, 10000, 15000 and 20000 documents were used. DIMASP-C was compared against GSP [Srikant et al.,

[1] http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

1996], an apriori-like candidate-generation-and-test algorithm, and cSPADE [Zaki 2000], GenPrefixSpan [Antunes et al. 2003] and DELISP [Lin et al. 2005], three pattern-growth algorithms. Excepting for GSP, the original programs provided by the authors were used. All the experiments with DIMASP-C were done in a computer with an Intel Pentium 4 running at 3 GHz with 1GB RAM.

```
Step 3: Algorithm to find all MSP
Input: Structure from step 2 and $\beta$ threshold
Output: MSP set
For all the documents $D_{J\cdots(\beta-1)} \in DDB$ do
    MSP set ← Find all MSP w.r.t. the document ( $D_J$ )
Step 3.1: Algorithm to find all MSP with respect to the document $D_J$
Input: A $D_J$ from the data structure and a $\beta$ threshold
Output: The MSP set w.r.t. to $D_J$
For all the nodes $\delta_{i=1\cdots n} \in D_J$ i.e. $\langle w_i, w_{i+1}\rangle \in D_J$ do
  If Array [ $\delta_i$.index ].frequency $\geq \beta$ then
    PMSP←Array[ $\delta_i$.index ].$\langle w_i, w_{i+1}\rangle$ //the initial pair
    $\Delta'$ ←Copy the rest of the list of $\Delta$ beginning from
        $\delta_i$.NextDoc
    $\Delta'_f$ ← Number of different documents in $\Delta'$
    $\delta'_i \leftarrow \delta_i$
    While $\Delta'_f \geq \beta$ do the growth the PMSP
        $\Delta''$ ←Array[ $\delta'_{i+1}$.index ].list
  $\Delta' \leftarrow \Delta' \,\&\, \Delta''$ i.e. $\{\alpha \in \Delta' \mid (\alpha.\text{index} = \delta'_{i+1}) \wedge (\delta'_i.\text{NextNode} = \alpha)\}$
        $\Delta'_f$ ←Number of different documents in $\Delta'$
        If $\Delta'_f \geq \beta$ then to grow the PMSP
            Array [ $\delta'_{i+1}$.index ].mark ← “used”
            PMSP ← PMSP + Array [ $\delta'_{i+1}$.index ].$\langle w_{i+1}\rangle$
                $\delta'_i \leftarrow \delta'_{i+1}$ i.e. $\delta'_i$.NextNode
    end-while
    If |PMSP| ≥ 3 then add the PMSP to the MSP set
        MSP set ← add a k-PMSP to the MSP set //step 3.1.1
end-for
For all the cells $C \in$ Array do the addition of the 2-MSP
  If $C_f \geq \beta$ and C.mark = “not used” then add it as 2-MSP
    2-MSP set ← add $C.\langle w_i, w_{i+1}\rangle$
```

Figure 2: Step 3 of DIMASP-C.

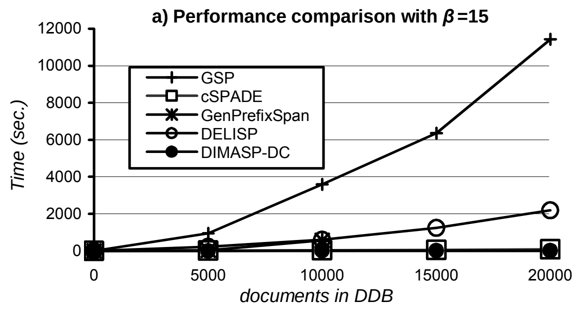
```
Step 3.1.1: Algorithm to add a PMSP to the MSP set
Input: A k-PMSP, MSP set                Output: MSP set
If (k-PMSP ∈ k-MSP set) or (k-PMSP is subsequence of some
longer k-MSP) then // do not add anything
    return MSP set
else //add as a MSP
    k-MSP set ← add k-PMSP
    {delete S ∈ MSP set | S ⊆ k-PMSP }
    return MSP set
```

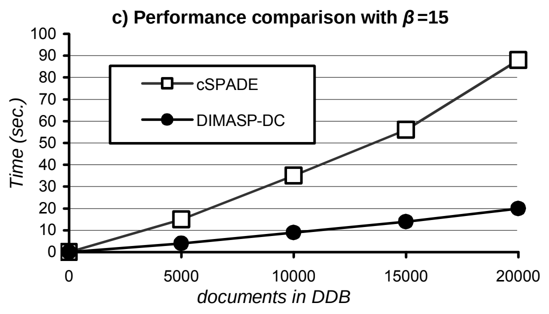
Figure 3: Algorithm to add a PMSP to the MSP set.

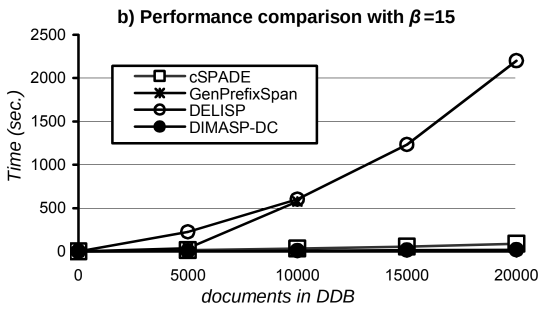
4.1.1 Experiments with DIMASP-C

In Fig. 5a the performance comparison of DIMASP-C (with all the steps), cSPADE, GenPrefixSpan, DELISP and GSP algorithms with β=15 is shown. Fig. 5b shows the same comparison of Fig. 5a but the worst algorithm (GSP) was eliminated, here it is possible to see that DELISP is not as good as it seems to be in Fig. 5a. In this experiment GenPrefixSpan had memory problems; therefore it was only tested with the first 5000 and 10000 documents. Fig. 5c compares only DIMASP-C against the fastest algorithm cSPADE with respect to *β=15*. Fig. 5d draws a linear scalability of DIMASP-C with respect to *β=15*. An additional experiment with the lowest β=2 was performed, in this experiment DIMASP-C found a MSP of length 398 because there is a duplicated document in the collection, Fig. 5e shows these results.

In order to evaluate the incremental scalability of DIMASP-C, 4000, 9000 14000 and 19000 documents were processed, and 1000 documents were added in each experiment. Fig. 5f shows the results and compares them against cSPADE which needs to recompute all the MSP. Fig. 5g shows the distribution of the MSP for β=15 according to their length. Finally, Fig. 5h shows the number of MSP when β=1% of the documents in the collection was used.







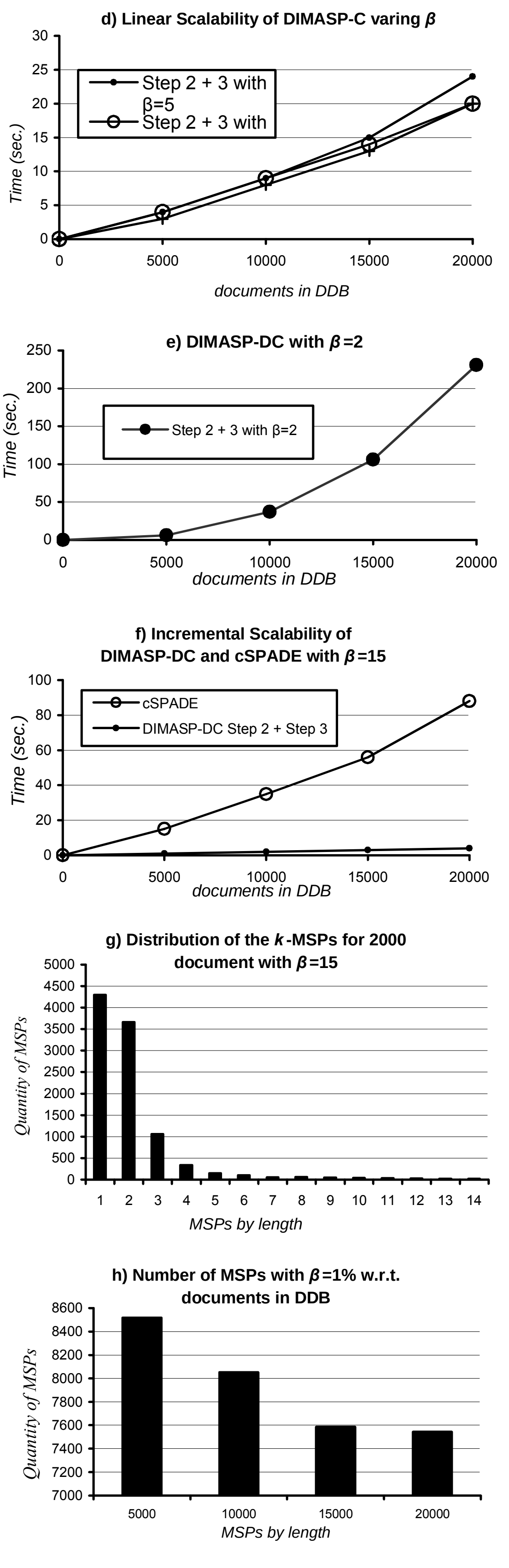


Figure 4: Performance results using Reuters-2157 collection.

4.2 DIMASP-D

Similar to DIMASP-C, the idea of DIMASP-D consists in finding all the sequential patterns in a data structure, which is built, in this case, from the single document to be analyzed. This structure stores all the different pairs of contiguous words that appear in the document, without losing their sequential order. Given a threshold *β* specified by the user, DIMASP-D reviews if a pair is *β*-frequent. In this case, DIMASP-D grows the pattern in order to determine all the possible maximal sequential patterns containing such pair as a prefix. The proposed algorithm has three steps described as follows:

In the first step, the algorithm assigns an *id* for each different word in the document.

The second step (fig. 6) consists in building the data structure. DIMASP-D will construct a data structure similar to the structure used for the document collection, but in this case containing only a single document. Since only one document is stored in the structure, the document index is not needed, instead of it, the position of the pair inside the document is stored, as it is shown in Fig. 7, this position is used to avoid overlapping among the instances of a maximal sequential pattern that appear into the document.

```
Input: A document $T$      Output: The data structure
For all the pairs $[t_i,t_{i+1}] \in T$ do
// if $[t_i,t_{i+1}]$ it is not in Array, add it
        PositionNode.Pos ← index ← array $[t_i,t_{i+1}]$;
        Array[index].Positions ← New PositionNode
        Array[index].Freq ← array[index].Freq+ 1
        Array[LastIndex].Positions.NextIndex ← index;
        Array[LastIndex].Positions.NextPos ← PositionNode;
        LastIndex ← index;
End-for
```

Figure 5: Step 2 of DIMASP-D.

In the last step (Fig. 8), DIMASP-D finds all the maximal sequential patterns in similar way as DIMASP-C, but now the *β*-frequency is verified inside the document, counting how many times a pattern appears without overlapping.

4.2.1 Experiments with DIMASP-D

For the experiments, we chose from the collection Alex[2] the document "Autobiography" by Thomas Jefferson with around 243,115 chars corresponding to: 31,517 words (approx. 100 pages); and the document "LETTERS" by Thomas Jefferson with around 1,812,428 chars and 241,735 words (approx. 800 pages). In both documents the stop words were not removed and only the numbers and punctuation symbols were omitted. In order to show the behavior of the processing time against the number of words in the document, we computed the MSP using DIMASP-D with the minimum threshold value, *β*=2.

[2] Public domain documents from American and English literature as well as Western philosophy. http://www.infomotions.com/alex/

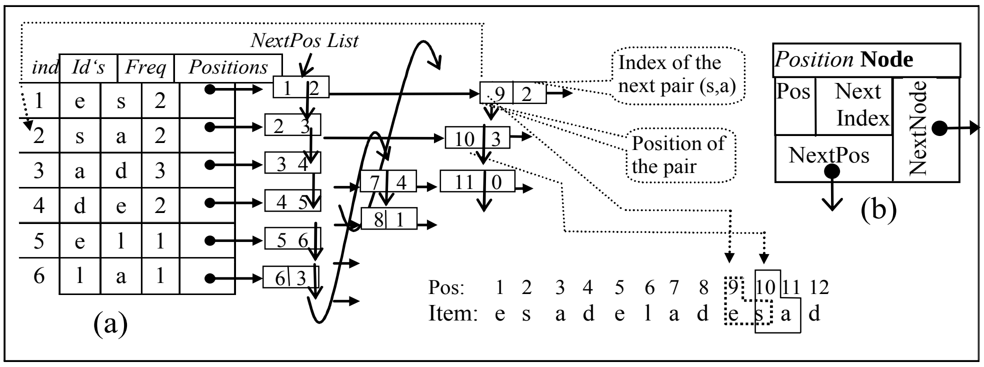
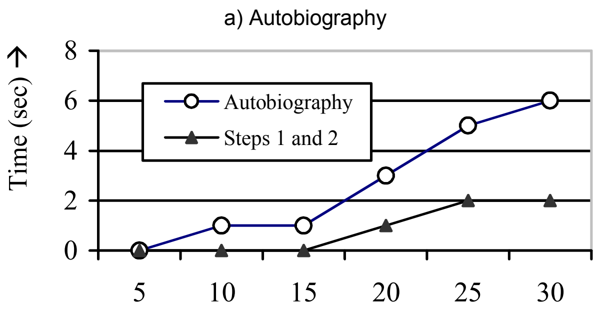


Figure 6: a) Data structure built by DIMASP-D for the text: "esadeladesad"
b) Node for positions list

```
Input: Data structure from phase 2        Output: MSP list
  Actual ← 1 //First element of NextPos List
while Actual ≠ 0 do
  if Array[Actual].Frequency ≥ $\beta$, then
    temporal ← Copy ( Array[Actual].Positions)
    PMS ← Array[Actual].$Id_1$ + Array [Actual].$Id_2$
    aux ← Array [Actual].NextIndex
      while aux ≠ 0 do  //expand the 2-sequence
    temporal ← Match((temporal.Pos + 1) AND (Array[aux].Positions.Pos)
    If | temporal | ≥ $\beta$, then
      if aux = Array, then there is a cycle,
           PMSP ← Cycle( $\beta$, temporal, Array, Actual, aux )
           If the PMSP cannot grow then exit from the while
      else PMSP ← PMSP + Array[aux].$Id_2$
    aux ← Array[Actual].NextIndex
  end-while
  delete all the MSP $\subseteq$ PMSP
  if (PMSP $\not\subset$ MSP) then  MSP←Add(PMSP)
  Actual ← Array[Actual].NextIndex
End-while
```

Figure 7: Step 3 of DIMASP-D.

Each graph in Fig. 9 corresponds to one document, processing different quantities of words. In the fig. 9a, we started with 5,000 words and used an increment of 5,000, in order to see how the processing time grows when the number of processed words is increased in the same document. In the fig. 9b, an increment of 40,000 words was used in order to see how the processing time grows for a big document. By the way, in both graphs the time for steps (1 and 2) is shown.



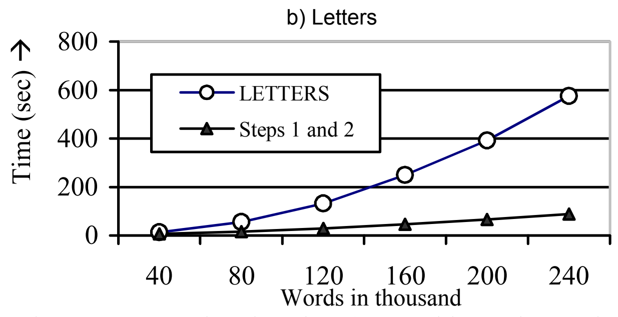


Figure 8: Processing time for: a) "Autobiography" and, b) "LETTERS".

For the same documents, the whole document was processed to find all the MSP, in order to appreciate (Fig. 10) how the performance of our algorithm is affected for different values of the threshold β. Fig. 10a shows the time in seconds for "Autobiography" and Fig. 10b for "LETTERS".

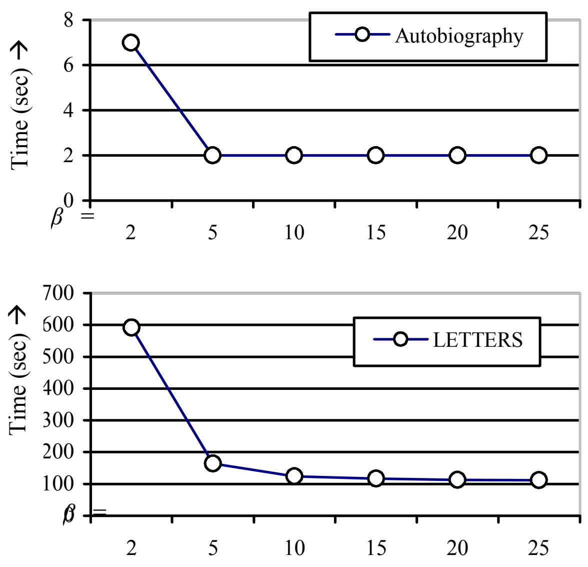


Figure 10: Time performance for different values of β for: a) "Autobiography" and b) "LETTERS".

Furthermore, we have included in Fig. 11 an analysis of the number of MSP obtained from the same documents for different values for *β*.

Additionally to these experiments, we processed the biggest document from the collection Alex, “An Inquiry into the nature …” by Adam Smith with 2,266,784 chars corresponding to 306,156 words (approx. 1000 pages) with *β*=2, all MSP were obtained in 1,223 seconds (approx. 20 min). All the experiments with DIMASP-D were done in a computer with an Intel Centrino Duo processor running at 1.6 GHz with 1GB RAM.

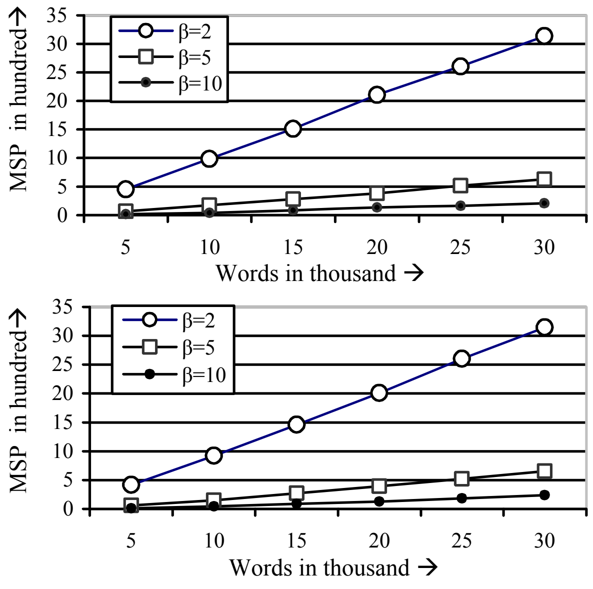


Figure 11: Amount of MSP generated for different values of β for a) “Autobiography” and b) “LETTERS”.

5 Concluding remarks

In this work, we have introduced two new algorithms for mining maximal sequential patterns into a document collection (DIMASP-C) and into a single document (DIMASP-D).

According to our experiments, DIMASP-C is faster that all the previous algorithms. In addition, our algorithm allows processing the document collection in an incremental way; therefore if some documents are added to the collection, DIMASP-C only needs to process the new documents, which allows updating the maximal sequential patterns much faster than mining them over the whole modified collection by applying any of the previous algorithms.

DIMASP-D is a first approach for mining maximal sequential patterns into a single document, which allows processing large documents in a short time.

Since our proposed algorithms were designed for processing textual databases, they are faster than those proposed for transactional databases, therefore our algorithms are more suitable to apply maximal sequential patterns for solving more complex problems and applications in text mining, for example: question answering [Denicia-Carral et al. 2006; Juárez-González et al. 2007; Aceves-Pérez et al. 2007], authorship attribution [Coyotl-Morales et al. 2006], automatic text summarization [Ledeneva et al. 2008], document clustering [Hernandez-Reyes et al. 2006], and extraction of hyponyms [Ortega-Mendoza et al. 2007], among others.

As future work, we are going to adapt DIMASP-C and DIMASP-D for mining Maximal Sequential Patterns, allowing a bounded gap between words.

References

[1] Aceves-Pérez Rita Marina, Montes-y-Gómez Manuel, Villaseñor Pineda Luis, “Enhancing Cross-Language Question Answering by Combining Multiple Question Translations”, *8th Intelligent Text Processing and Computational Linguistics (CICLing’2007), LNCS 4394*, Springer-Verlag, 2007, pp. 485–493.

[2] Agrawal R, Imielinski T, Swami A, “Mining association rules between sets of items in large databases”, *Proceedings of the 1993ACM-SIGMOD international conference on management of data (SIGMOD’93),* Washington, DC, 1993, pp 207–216.

[3] Antunes, C., Oliveira A., “Generalization of Pattern-growth Methods for Sequential Pattern Mining with Gap Constraints”, *Third IAPR Workshop on Machine Learning and Data Mining MLDM2003 LNCS 2734*, 2003, pp. 239–251.

[4] Coyotl-Morales Rosa Maria, Villaseñor-Pineda Luis, Montes y Gómez Manuel, Rosso Paolo, “Authorship Attribution Using Word Sequences”, *11th Iberoamerican Congress on Pattern Recognition (CIARP’2006), LNCS 4225*, Springer Verlag, 2006, pp. 844–853.

[5] Denicia-Carral Claudia, Montes-y-Gómez Manuel, Villaseñor-Pineda Luis, García Hernández René, “A Text Mining Approach for DefinitionQuestion Answering”, *5th International Conference on NLP (Fintal 2006), LNAI 4139*, Springer-Verlag, 2006, pp. 76–86.

[6] Hernandez-Reyes E, Garcia-Hernandez RA, Carrasco-Ochoa JA, J. Fco. Martínez-Trinidad, “Document clustering based on maximal frequent sequences”, *5th International Conference on NLP (Fintal 2006), LNAI 4139*, Springer-Verlag, 2006, pp. 257-267.

[7] Jian Pei, Jiawei Han, et. al. “Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach”, *IEEE Transactions on Knowledge and Data Engineering*, 16(10), 2004, pp. 1424–1440.

[8] Jiawei Han and Micheline Kamber, “*Data Mining: Concepts and Techniques*”, Morgan Kaufmann Publishers, 2000.

[9] Jiawei Han, Hong Cheng, Dong Xin, Xifeng Yan, “Frequent pattern mining: current status and future directions”, *Data Min Knowl Disc* 15, 2007, pp. 55–86.

[10] Juárez-González Antonio, Téllez-Valero Alberto, Delicia-Carral Claudia, Montes-y-Gómez Manuel and Villaseñor-Pineda Luis, “Using Machine

Learning and Text Mining in Question Answering", *7th Workshop of the Cross-Language Evaluation Forum, CLEF 2006, LNCS 4730*, Springer-Verlag, 2007, pp. 415–423.

[11] Ledeneva Yulia, Gelbukh Alexander, and García-Hernández René Arnulfo, "Terms Derived from Frequent Sequences for Extractive Text Summarization", *LNCS 4919*, Springer-Verlag, 2008, pp. 593–604.

[12] Lin, M. Y., and S. Y. Lee, "Efficient Mining of Sequential Patterns with Time Constraints by Delimited Pattern-Growth", *Knowledge and Information Systems*, 7(4), 2005, pp. 499-514.

[13] Ortega-Mendoza Rosa M., Villaseñor-Pineda Luis and Montes-y-Gómez Manuel, "Using Lexical Patterns for Extracting Hyponyms from the Web", *Mexican International Conference on Artificial Intelligence MICAI 2007, LNAI 4827*, Springer-Verlag, 2007, pp. 904-911.

[14] Srikant, R., and Agrawal, R., "Mining sequential patterns: Generalizations and performance improvements", *5th Intl. Conf. Extending Database Discovery and Data Mining, LNCS 1057*, Springer-Verlag, 1996, pp. 3–17.

[15] Zaki, Mohammed, "SPADE: An Efficient Algorithm for Mining Frequent Sequences", *Machine Learning*, Kluwer Academic Publishers, 42, 2000, pp. 31–60.

Grammar of ReALIS and the Implementation of its Dynamic Interpretation

Gábor Alberti and Judit Kleiber
Department of Linguistics, University of Pécs, 7624 Pécs, Ifjúság 6, Hungary
E-mail: realis@btk.pte.hu

Keywords: lexical and discourse semantics, Hungarian, total lexicalism

Received: February 20, 2009

ReALIS, REciprocal And Lifelong Interpretation System, is a new "post-Montagovian" theory concerning the formal interpretation of sentences constituting coherent discourses, with a lifelong model of lexical, interpersonal and encyclopaedic knowledge of interpreters in its centre including their reciprocal knowledge on each other. First we provide a 2 page long summary of its 40 page long mathematical definition. Then we show the process of dynamic interpretation of a Hungarian sentence (Hungarian is a "challenge" because of its rich morphology, free word order and sophisticated information structure). We show how an interpreter can anchor to each other in the course of dynamic interpretation the different types of referents occurring in copies of lexical items retrieved by the interpreter on the basis (of the morphemes, word order, case and agreement markers) of the sentence performed by the speaker. Finally, the computational implementation of ReALIS is demonstrated.

Povzetek: Predstavljen je sistem ReALIS za dinamično interpretacijo zapletenih stavkov.

1 Introduction

$\Re$eALIS [2] [4], *REciprocal And Lifelong Interpretation System*, is a new "post-Montagovian" [15] [17] theory concerning the formal interpretation of sentences constituting coherent discourses [9], with a *lifelong* model [1] of lexical, interpersonal and cultural/encyclopaedic knowledge of interpreters in its centre including their *reciprocal* knowledge on each other. The decisive theoretical feature of $\Re$eALIS lies in a peculiar reconciliation of three objectives which are all worth accomplishing in formal semantics but could not be reconciled so far.

The first aim concerns the exact formal basis itself ("Montague's Thesis" [20]): human languages can be described as interpreted *formal* systems. The second aim concerns *compositionality*: the meaning of a whole is a function of the meaning of its parts, practically postulating the existence of a *homomorphism* from syntax to semantics, i.e. a rule-to-rule correspondence between the two sides of grammar.

In Montague's interpretation systems a traditional logical representation played the role of an intermediate level between the syntactic representation and the world model, but Montague argued that this intermediate level of representation can, and should, be eliminated. (If α is a compositional mapping from syntax to discourse representation and β is a compositional mapping from discourse to the representation of the world model, then $\gamma=\alpha^\circ\beta$ must be a compositional mapping directly from syntax to model.) The post-Montagovian history of formal semantics [17] [9], however, seems to have proven the opposite, some principle of "discourse representationalism": "some level of [intermediate] representation is indispensable in modelling the interpretation of natural language" [14].

The Thesis of $\Re$eALIS is that the two fundamental Montagovian objectives *can* be reconciled with the principle of "discourse representationalism" – by embedding discourse representations in the world model, getting rid of an intermediate level of representation in this way while preserving its content and relevant structural characteristics. This idea can be carried out in the larger-scale framework of embedding discourse representations in the world model *not directly* but as parts of the representations of interpreters' minds, i.e. that of their (permanently changing) information states [3].

2 Definition

The frame of the mathematical definition of $\Re$eALIS (whose 40 page long complete version is available in [4] (Sections 3-4)) is summarized in this section. As interpreters' mind representations are part of the WORLD MODEL, the definition of this model $\Re = \langle U, W_0, W\rangle$ is a quite complex structure where

- U is a countably infinite set: the UNIVERSE
- $W_0 = \langle U_0, T, S, I, D, \Omega, A\rangle$: the EXTERNAL WORLD
- W is a partial function from set I×Tm where W[i,t] is a quintuple $\langle U[i], \sigma[i,t]^{\Pi}, \alpha[i,t]^{\Psi}, \lambda[i,t]^{\Lambda}, \kappa[i,t]^{K}\rangle$: the INTERNAL-WORLD FUNCTION.

The external world consists of the following components:

- U_0 is the external universe ($U_0 \subset U$), whose elements are called entities
- $T = \langle T, \Theta \rangle$ is a structured set of temporal intervals ($T \subset U_0$)
- $S = \langle S, \Xi \rangle$ is a structured set of spatial entities ($S \subset U_0$)
- $I = \langle I, Y \rangle$ is a structured set of interpreters ($I \subset U_0$)
- $D = \langle D, \Delta \rangle$ is a structured set of linguistic signs (practically morph-like entities and bigger chunks of discourses) ($D \subset U_0$)
- $\Omega \subset T{\times}U_0{}^*$ is the set of core relations (with time intervals as the first argument of all core relations)
- A is the information structure of the external world (which is nothing else but relation structure Ω reformulated as a standard simple information structure, as is defined in [22: 245]; its basic elements are called the infons of the external world
- T, S, I and D are pairwise disjoint, infinite, proper subsets of the external universe U_0 which meet further requirements that cannot be elaborated here.

The above mentioned *internal-world function* W is defined as follows:

- The relation structure W[i,t] is called the internal world (or information state) of interpreter i at moment t
- $U[i] \subset U$ is an infinite set: interpreter i's internal universe (or the set of i's referents, or internal entities); U[i'] and U[i"] are disjoint sets if i' and i" are two different interpreters
- in our approach what changes during a given interpreter's lifespan is not his/her referent set U[i] but only the four relations among the (peg-like [12]) referents, listed below, which are called i's internal functions:
- $\sigma[i,t]\Pi : \Pi{\times}U[i] \rightarrow U[i]$ is a partial function: the eventuality function (where Π is a complex label characterizing argument types of predicates)[1]
- $\alpha[i,t]\Psi : \Psi{\times}U[i] \rightarrow U[i]{\cup}U_0$ is another partial function: the anchoring function (α practically identifies referents, and Ψ contains complex labels referring to the grammatical factors legitimizing these identifications)
- $\lambda[i,t]\Lambda : \Lambda{\times}U[i] \rightarrow U[i]$ is a third partial function: the level function (elements of Λ are called level labels); the level function is practically intended to capture something similar to the "box hierarchy" among referents in complex Kampian DRS boxes [10] enriched with some rhetorical hierarchy in the style of SDRT [2]
- $\kappa[i,t]K : K \rightarrow U[i]$ is also a partial function: the cursor, which points to certain temporary reference points prominently relevant to the interpreter such as "Now", "Here", "Ego", "Then", "There", "You" etc.

[1] The DRS condition [e: p t r_1 ... r_K] [10] (e.g. [e: resemble now Peter Paul]) can be formulated with the aid of this function as follows (with i and t fixed):
$\sigma(\langle Pred, \pi \rangle, e) = p$, $\sigma(\langle Temp, \tau \rangle, e) = t$, $\sigma(\langle Arg, \psi_1 \rangle, e) = r_1$, ..., $\sigma(\langle Arg, \psi_K \rangle, e) = r_K$.

The temporary states of these four internal functions above an interpreter's internal universe (which meet further requirements that cannot be elaborated here) serve as his/her "agent model" [11] in the process of (static and dynamic) interpretation.

Suppose the information structure A of the external world (defined above as a part of model $\mathfrak{R} = \langle U, W_0, W \rangle$) contains the following infon: $\iota = \langle$PERCEIVE, t, i, j, d, s$\rangle$, where i and j are interpreters, t is a point of time, s is a spatial entity, d is a discourse (chunk), and PERCEIVE is a distinguished core relation (i.e. an element of Ω). The INTERPRETATION of this "perceived" discourse d can be defined in our model relative to an external world W_0 and internal world W[i,t].

The DYNAMIC INTERPRETATION of discourse d is essentially a mapping from W[i,t], which is a temporary information state of interpreter i, to another (potential) information state of the same interpreter that is an *extension* of W[i,t]; which practically means that the above mentioned four *internal functions* (σ, α, λ, κ) are to be developed monotonically by *simultaneous recursion*, expressing the addition of the information stored by discourse d to that stored in W[i,t].

The new value of eventuality function σ chiefly depends on the *lexical items* retrieved from the interpreter's internal mental lexicon as a result of the perception and recognition of the words / morphemes of the interpreter's mother tongue in discourse d. This process of the unification of lexical items can be regarded as the first phase of the dynamic interpretation of (a sentence of) d. In our $\mathfrak{R}$eALIS framework, as will be shown in the next section, extending function σ corresponds to the process of accumulating DRS condition rows [17] containing referents which are all – still – regarded as different from each other.

It will be the next phase of dynamic interpretation to *anchor* these referents to each other (by function α) on the basis of different grammatical relations which can be established due to the recognized *order* of morphs / words in discourse d and the *case*, *agreement* and other markers it contains. In our approach two referents will never have been *identified* (or deleted), they will only be anchored to each other; but this anchoring essentially corresponds to the identification of referents in DRSs.

The third phase in this simplified description of the process of dynamic interpretation concerns the third internal function, λ, the level function. This function is responsible for the expression of intra- and inter-sentential scope hierarchy [21] / information structure [23] / rhetorical structure [9], including the embedding of sentences, one after the other, in the currently given information state by means of rhetorical relations more or less in the way suggested in SDRT [9].

It is to be mentioned at this point that the information-state changing dynamic interpretation and the truth-value calculating *static interpretation* are mutually based upon each other. On the one hand, static interpretation operates on the *representation* of sentences (of discourses) which is nothing else but the output result of dynamic interpretation. On the other hand, however, the above discussed phases of dynamic interpretation (and chiefly the third phase) include subprocesses requiring static interpretation: certain *presuppositions* are to be verified [17].

The interpreter's fourth internal function, cursor κ, plays certain roles during the whole process of dynamic interpretation. *Aspect*, for instance, can be captured in our approach as the resetting or retaining of the *temporal* cursor value as a result of the interpretation of a sentence (→ *non-progressive / progressive* aspect). It can be said in general that the input cursor values have a considerable effect on the embedding of the "new information" carried by a sentence in the interpreter's current information state and then this embedding will affect the output cursor values.

DYNAMIC INTERPRETATION in a ℜeALIS model ℜ=⟨U, W_0, W⟩, thus, is a partial function Dyn which maps a (potential) information state W° to a discourse d and an information state W[i,t] (of an interpreter i): Dyn(d) : ⟨ℜ,W[i,t]⟩ → ⟨W°, e°, U°⟩, where U°, shown up in the output triple, is the COST of the given dynamic interpretation (coming from presuppositions legitimized by *accommodation* instead of *verification*), and e° is the eventuality that the output cursor points to (e°is to be regarded as representing the content of discourse d). Function Dyn(d) is *partial*: where there is no output value, the discourse is claimed to be ill-formed in the given context. Due to the application of cost, ill-formedness is practically a gradual category in ℜeALIS: a great cost of interpretation qualifies the discourse to be "almost unacceptable".

The STATIC INTERPRETATION of a discourse d is nothing else but the static interpretation of the eventuality referent representing it. Its recursive definition is finally based upon anchoring internal entities of interpreters to external entities in the external universe, and advances from smaller units of (the sentences of) the discourse towards more complex units. We do not intend to enter into details in this paper.

3 Example

The detailed analysis of a Hungarian sentence will serve as an illustration of the process of dynamic interpretation.

Hungarian is a "challenge" because of its very rich morphology and extremely free word order [18], which enables to express subtle differences in meaning [23]. We claim that the very abstract and morpheme-based monostratal ℜeALIS approach to grammar, relying on the four internal functions σ, α, λ and κ (discussed above) and a complex system of *ranked* lexical requirements (which is nearly an apparatus similar to those known from optimality theories [8]), *neutralizes* the difference between languages where the meaning of a sentence can primarily be calculated on the basis of words in a strict order and languages where what are relevant to this calculation are basically morphemes within words and affixes (e.g. case and agreement markers).

How can an interpreter anchor to each other the different types of referents occurring in copies of lexical items retrieved in the course of dynamic interpretation? Let us consider the Hungarian sentence below:

(1) *REQUIREMENTS* (↑) AND *OFFERS* (↓) IN LEXICAL ITEMS

Péter hasonlít ar-ra a magas német úszó-bajnok-ra.
Peter resemble that-onto the tall German swimming-champion-onto
'Peter resembles that tall German swimming champion.'

a. Péter: [e_P: $\Rightarrow_?$ e_P' e_P", e_P' : p_{Peter} r_P]
e_P ↑ λ:: ⟨...⟩
e_P" ↑ α:: ⟨...⟩
e_P' ↑ λ:: ⟨⬉.supp⟩↓ e_P
5th row e_P" ↑ λ:: ⟨⬉.cons⟩↓ e_P'
r_P ↑ α:: Pred: ⟨⟨Cat,+2,$X_\varnothing$ ⟩, ⟨Agr,+2,3Sg⟩⟩
↑ α:: Ant: ⟨...⟩
↓ α:: ∅: ⟨⟨Cat,0,PropN⟩, ⟨Case,0,∅⟩, ⟨Agr,0,3Sg⟩⟩

b. hasonlít: [e_{res} : $p_{resemble}$ r_{res}' r_{res}"]
e_{res} ↑ λ:: ⟨...⟩
r_{res}'↑ α:: ArgN: ⟨⟨Ord,-7,Nei⟩, ⟨Cat,+2,N⟩, ⟨Case,+2,∅⟩⟩
↑ α:: ArgD: ⟨⟨Cat,+2,GQD⟩⟩
5th row ↓ α:: ∅: ⟨⟨Cat,0,$V_{\varnothing+rA}$⟩, ⟨Agr,0,3Sg⟩⟩
r_{res}" ↑ α:: ArgN: ⟨⟨Ord,+7,Nei⟩, ⟨Cat,+2,N⟩, ⟨Case,+2,rA⟩⟩
↑ α:: ArgD: ⟨⟨Cat,+2,GQD⟩⟩
↓ α:: ∅: ⟨⟨Cat,0, $V_{\varnothing+rA}$⟩⟩

c. arra:
r_{that} ↑ α:: Adj: ⟨⟨Ord,+6,Nei⟩, ⟨Cat,+2,N⟩, ⟨Agr,+2,{3,Sg,rA}⟩⟩
↑ α:: Out: ⟨⟨Gest,+2,Glance⟩, ⟨Dist,+2,Long⟩⟩

d. a(z): [e_{the}: $\Rightarrow_?$ e_{the}' e_{the}", σ(⟨Arg, ∅⟩, e_{the}')= r_{the}]
e_{the} ↑ λ:: ⟨...⟩
e_{the}' ↑ α:: ⟨...⟩
e_{the}" ↑ α:: ⟨...⟩
5th row e_{the}' ↑ λ:: ⟨⬉.supp⟩↓ e_{the}
e_{the}" ↑ λ:: ⟨⬉.cons⟩↓ e_{the}'
r_{the} ↑ α:: Adj: ⟨⟨Ord,+5,Nei⟩, ⟨Cat,+2,N⟩⟩
↑ α:: Pred: ⟨⟨...⟩⟩
↑ α:: Ant: ⟨...⟩
10th row ↓ α:: Arg: ⟨⟨Cat,0,+Art⟩⟩

e. magas: [e_{tall}: ∧ e_{tall}' e_{tall}", e_{tall}': p_{tall} r_{tall}]
e_{tall}" ↑ α:: ⟨...⟩
e_{tall}'↑ λ:: ⟨⬀.conj⟩↓ e_{tall}"
e_{tall}↑ λ:: ⟨⬀.conj⟩↓ e_{tall}"
5th row r_{tall}↑ α:: Adj: ⟨⟨Ord,+2,Nei⟩,

⟨Cat,+2,N⟩⟩
↓ α:: ∅: ⟨⟨Cat,0,A_{color}⟩⟩

f. német: e_{germ}: ∧ e_{germ}’ e_{germ}”,
e_{germ}’: p_{German} r_{germ}]
e_{germ}” ↑ α:: ⟨...⟩
e_{germ}’↑ λ:: ⟨⬀.conj⟩↓ e_{germ}”
e_{germ}↑ λ:: ⟨⬀.conj⟩↓ e_{germ}”
5th row r_{germ}↑ α:: Adj: ⟨⟨Ord,+1,Nei⟩,
⟨Cat,+2,N⟩⟩
↓ α:: ∅: ⟨⟨Cat,0,A_{nation}⟩⟩

g. úszó-: [e_{sw}: ∧ e_{sw}’ e_{sw}”,
e_{sw}’: $p_{swimming}$ r_{sw}]
e_{sw}” ↑ α:: ⟨...⟩
e_{sw}’↑λ:: ⟨⬀.conj⟩↓ e_{sw}”
e_{sw} ↑λ:: ⟨⬀.conj⟩↓ e_{sw}”
5th row r_{sw}↑ α:: Adj: ⟨⟨Ord,+ ⅓,Nei⟩,
⟨Cat,0,N⟩⟩
↓ α:: ∅: ⟨...⟩

h. bajnok: [e_{ch}: $p_{champion}$ r_{ch}]
e_{ch} ↑ λ:: ⟨...⟩
r_{ch}↑ α:: Pred: ⟨⟨Cat,+5, $X_{\varnothing}$⟩⟩
↓ α:: ∅: ⟨⟨Cat,0,ComN⟩,
⟨Case,0,∅⟩, ⟨Agr,0,3Sg⟩⟩)

-ra:
r_{onto}’ ↑ α:: Stem: ⟨⟨Ord, – ⅛Nei⟩, ⟨Cat,+2, N⟩⟩
r_{onto}”↑ α:: Pred: ⟨⟨Cat,+2, X_{-rA}⟩⟩

In ℜeALIS the lexical representation belonging to a morpheme typically contains reference to a predicate (e.g. $p_{champion}$) furnished with argument referents (e.g. r_{ch} above in (1h)), a temporal referent and a referent referring to the fact that "the given predicate holds true" (the eventuality referent e_{ch} refers to the fact that somebody is a champion). In the analysis that this paper provides temporal referents are ignored for the sake of simplicity. As was mentioned earlier, this "eventuality construction" is registered by internal function σ.

The lexical representation belonging to a morpheme should predict about these referents how they will connect to referents coming from other lexical representations retrieved in the course of dynamic interpretation of a sentence when the given morpheme gets into the given sentence. We mean the extension of α, practically responsible for identification, and λ, responsible for scope hierarchy and/or rhetorical relations. Lexical items thus impose "requirements" on the potential intrasentential environments accommodating the given morphemes and provide "offers" for other morphemes' items to help them (these other morphemes) find them.

In what follows we provide comments on a few (but not all) lexical requirements and offers. Let the verb (*hasonlít* 'resemble') be the first (1b), with its 8 row long lexical description. The first row contains the "eventual" representation of the semantic contribution of this verb, which consists of an eventuality referent (referring to the fact that somebody resembles somebody), a predicate referent, a temporal referent (ignored), and two argument referents. What the second row says is that the eventuality referent, which practically represents the piece of information carried by this word-size morpheme) should be linked to some part of the interpreter's current information state (potentially including pieces of information coming from the sentence being interpreted) by means of an appropriate level label.

According to the third row, the referent belonging to the first argument of the verb (r_{res}’) should be anchored to the referent coming from the lexical description of a morpheme (word) "in the neighbourhood" ('Nei') as for word order ('Ord'), which belongs to the category of nouns ('⟨Cat,...,N⟩') and bears the Nominative case, which is unmarked in Hungarian ('∅'). What the fourth row adds to this complex requirement is that referent r_{res}’ should also be anchored to a referent coming from the lexical item of a morpheme (or word) which can play the role of a generalized-quantifier determiner 'GQD' [10]. Let us look at (2f-g) below: the proper name *Péter* is suitable for both roles as it is a noun in the nominative and implicitly includes an article ('*the* person called Peter'). The fifth row says that lexical item (1b) "offers" a verb with two specific arguments ('$V_{\varnothing+rA}$') in 3Sg. On the basis of this characterization referent r_P in lexical item (1a) may find r_{res}’; see (2a) below. The intuitive content of the lexical information conveyed in rows 3-5 is that *who resembles somebody is nothing else but Peter*[2]; the three equations below in (2a, f, g) declare this fact three times, on the basis of different grammatical evidence.

Rows 6-8 characterize the second argument of *hasonlít*. Its referent r_{res}” should be anchored to two different referents: one coming from the lexical description of a noun ('ArgN') (2h, x) and another one contained by the item of a determiner-like element ('ArgD') (2i, m). In this way the interpreter can identify the person that Peter resembles with one mentioned as "the champion".

(2) THE CONNECTIONS COMPLETED AMONG "SIMPLE" REFERENTS
a. r_P ↑α:: Pred: ⟨Cat, Agr⟩↓ r_{res}’
b. r_P ↑α:: Ant: ⟨...⟩↓ $r_{Péter}$
c. e_P ↑λ:: ?↓ ?
d. e_P” ↑α:: ?↓ ?
e. e_{res}↑λ:: ⟨⬉.cons⟩↓ ? (assertion!)
f. r_{res}’ ↑α:: ArgN: ⟨Ord, Cat, Case⟩↓ r_P
g. r_{res}’ ↑α:: ArgD: ⟨Cat⟩↓ r_P
h. r_{res}” ↑α:: Arg: ⟨Ord, Cat, Case⟩↓ r_{ch}
i. r_{res}” ↑α:: Arg: ⟨Cat⟩↓ r_{the}
j. r_{that} ↑α:: Adj: ⟨Ord, Cat, Agr⟩↓ r_{ch}
k. r_{that} ↑α:: Out: ⟨Gest, Dist⟩↓ $u_{HansMüller}$
l. r_{the} ↑α:: Adj: ⟨Ord, Cat⟩↓ r_{ch}
m. r_{the} ↑α:: Pred: ⟨⟨...⟩↓ r_{res}”

[2] We call this relation established between two lexical items that show some grammatical sensitivity to each other (e.g. agreement, case marking, adjacency in word order) – *copredication*: they provide two predications about the *same* referent.

n. $r_{the}\uparrow\alpha$:: Ant: ⟨...⟩$\downarrow$ r_a
o. $e_{the}\uparrow\lambda$:: ⟨$\nwarrow$.cons⟩$\downarrow$? (argument position!)
p. e_{the}’ $\uparrow\alpha$:: ?$\downarrow$?
q. e_{the}” $\uparrow\alpha$:: ?$\downarrow$?
r. $r_{tall}\uparrow\alpha$:: Adj: ⟨Ord, Cat⟩$\downarrow$ r_{ch}
s. e_{tall}” $\uparrow\alpha$:: ?$\downarrow$?
t. $r_{germ}\uparrow\alpha$:: Adj: ⟨Ord, Cat⟩$\downarrow$ r_{ch}
u. e_{germ}” $\uparrow\alpha$:: ?$\downarrow$?
v. $r_{sw}\uparrow\alpha$:: Adj: ⟨Ord, Cat⟩$\downarrow$ r_{ch}
w. e_{sw}” $\uparrow\alpha$:: ?$\downarrow$?
x. $r_{ch}\uparrow\alpha$:: Pred: ⟨Cat⟩$\downarrow$ r_{res}”
y. r_{onto}’ $\uparrow\alpha$:: Stem: ⟨⟨Ord, Cat⟩$\downarrow$ r_{ch}
z. r_{onto}” $\uparrow\alpha$:: Pred: ⟨Cat⟩$\downarrow$ r_{res}”
aa. $e_{ch}\uparrow\lambda$:: ?$\downarrow$?

Now let us turn to the lexical item that belongs to the noun stem *bajnok* ‘champion’ (1h). According to row 3, a predicate with a nominative argument is to be found because, according to row 4, this stem is a common noun which “seems” to be in the nominative case. Number ‘5’ in row 3, however, “permits” that the requirement in question be satisfied *indirectly*. Numbers like this in the middle of the triples expressing requirements are *ranks*. If this rank is ‘1’, the given requirement must be satisfied in the way described. If the rank is weaker (ρ>1), there are also alternative ways of satisfaction at our disposal, typically with reference to higher ranked requirements (e.g. (1i.row3). Requirement (1h.row3), thus, can be satisfied (2x), but indirectly, due to (2z)).

It is also typical that requirements concerning word order can be satisfied *indirectly*. There are five lexical items in the example that contain requirements demanding that a certain word *immediately precede* the common noun ‘champion’ (see (1c-g)). The adjective expressing nationality is required to be the word adjacent to the noun to the highest degree: rank ‘+1’ expresses this fact in (1f.5). The fraction rank in (1g.5) implies an even stricter neighbourhood but this should be carried out within one word in Hungarian (*úszóbajnok* ‘swimming champion’). The other adjective referring to a personal characteristic, ‘tall’, should remain before ‘German’ because of its rank number ‘2’ in (1e.5). Then the weaker ranks ‘5’ in the lexical item of the definite article (1d.7) and ‘6’ in that of the demonstrative pronoun (1c.2) lead to the following grammatical word order in the prenominal zone in question: *arra a magas német úszóbajnokra* ‘that’ ‘the’ ‘tall’ ‘German’ ‘swimming’ ‘champion’. Alternative orders are ill-formed.

The explanation relies on the ranks discussed above: an adjacency requirement of rank k concerning words w’ and w” can be regarded as satisfied if w’ is adjacent to w”, indeed, or each word ξ between w’ and w” is such that the requirement demanding its being there is of a higher rank n (n<k, or n≤k) or ξ is a dependent of a word like this. We have a hypothesis concerning the nature of UG highly relevant to the efficiency of *implementation*: it is possible to work out a system of adjacency ranks in a way that enables us to check whether there is a *single* “legitimate” word between w’ and w” in the above discussed sense, instead of checking in the case of *all* words between w’ and w” whether they are legitimate “inhabitants” in that zone.

Our next comment concerns the semantic content of the definite article (1d.1), which is represented by eventuality referent e_{the}. Following [3], we assume that ‘the’ organizes the semantic content of a sentence in the form of an implication with the information coming from a certain noun and its “dependents” as its premise (e_{the}’) and the information typically coming from the verb as the conclusion (e_{the}”). What e_{the} expresses, thus, is something like this: “if somebody is a tall German swimming champion (3e, g, h, i), then somebody resembles him (3f)”.

Similarly, the lexical item belonging to the proper name (1a) also contains an implication (e_P) due to the implicit definite article hidden in (Hungarian) proper names (1a.1). Its approximate content will prove to be the following at the end of the successful dynamic interpretation: “if somebody is a person called Peter (1a.1), then (3b) he resembles a tall German swimming champion”.

(3) THE CONNECTIONS COMPLETED AMONG REFERENTS REFERRING TO EVENTUALITIES
a. $e_P\uparrow\lambda$:: ⟨$\nwarrow$.exp⟩$\downarrow$ κ^{prev}(Eve) (topic!; „What a surprise!”)
b. e_P” $\uparrow\alpha$:: ⟨...⟩$\downarrow$ e_{the}
c. $e_{res}\uparrow\lambda$:: ⟨$\nwarrow$.cons⟩$\downarrow$ e_{tall} (e_{res} will represent the *assertion* of the sentence)
d. $e_{the}\uparrow\lambda$:: ⟨$\nwarrow$.cons⟩$\downarrow$ e_P’ (argument position!)
e. e_{the}’ $\uparrow\alpha$:: Adj: ⟨Ord, Cat⟩$\downarrow$ e_{tall}
f. e_{the}” $\uparrow\alpha$:: Adj: ⟨Ord, Cat⟩$\downarrow$ e_{res}
g. e_{tall}” $\uparrow\alpha$:: Adj: ⟨Ord, Cat⟩$\downarrow$ e_{germ}
h. e_{germ}” $\uparrow\alpha$:: Adj: ⟨Ord, Cat⟩$\downarrow$ e_{sw}
i. e_{sw}” $\uparrow\alpha$:: Adj: ⟨Ord, Cat⟩$\downarrow$ e_{ch}
j. $e_{ch}\uparrow\lambda$:: ⟨$\nwarrow$.cons ⟩$\downarrow$ e_P’

A sentence like the one in (1) is to be embedded in the interpreter’s current information state, converting the double implicative structure into a (one-level) collection of *conjunctions* partly due to the successful anchoring of referents r_P (2b) and r_{ch} (2j-l) to two specific persons (say, Péter Puskás and Hans Müller) and the linking of the main eventuality of the sentence to some previous piece of information (‘κ^{prev}(Eve)’ in (3a) where κ is the cursor function). What we receive in this way can be as follows, for instance: “Péter resembles a tall German swimming champion, Hans Müller; and this fact serves as an Explanation (‘⟨$\nwarrow$.exp⟩’) [9] for the speaker’s surprise.

4 Implementation

The syntactic background of ReALIS has a “totally” lexicalist nature, which means that the grammar consists only of lexical items and their highly rich descriptions: properties and expectations (offers ($\downarrow$) and requirements ($\uparrow$) as used above). Phrase structure trees are not built, the only operation is unification. In this homogenous grammar word order is handled exactly like any other requirement (e.g. case or agreement). This is a more universal approach than applying phrase structure rules, since some languages hardly have any restrictions for word order (but have much more rules about agreement). We have been working on the implementation of this totally lexicalist grammar, which uses ReALIS for semantic analysis and representation.

In the past few years lexicalist parsers have become more and more successful and widely used. They can provide more detailed analysis than any other parser (some of them even have semantic component), and they can handle languages with rich morphology and free word order as well; furthermore, the outputs of these analyses can be parallel, thus machine translation can be achieved more easily. Coverage has been a secondary issue (many of these applications are still in experimental phase), but some of these parsers have actually reached the coverage of parsers using shallow techniques and statistical methods (e.g. the HPSG-based DELPH-IN, [12], or the LFG-based Parallel Grammar, [13]).

The success of lexicalist approaches (not only in theory but in the field of language technology as well) encourages us to keep working on the implementation, and see whether a totally lexicalist approach can be even more successful. A further argument for developing a parser based on ReALIS is the lack of programs which aim at providing detailed analysis and semantic representation, and can handle phenomena like rhetoric relations, discourse functions (topic, focus) and aspect. Finally, we believe that if the semantic representation is detailed enough (and ReALIS is more sophisticated than any earlier one) it can serve as some kind of interlingua, which could make it easier to achieve language-independent machine translation. (Lexicalist approaches like LFG and HPSG usually use transfer-based machine translation, which needs different transfer lexicons for every language pair.)

The first step of the implementation was to create a relational (SQL) database for the lexicon [24], which is universal enough to be able to store any lexical item of any language with all their properties (↓) and requirements (↑). This can be possible because properties are stored as tuples (rows) and not as attributes (columns), and the lexical items (which are also rows in the system) are connected to the relevant features by matching tables. This way we have gained a dynamically expandable system, since we do not have to define all possible properties of every language at the beginning, we can easily add new ones any time without changing the structure of the database.

The parsing begins with finding the main predicate (verb or nominal in Hungarian), then its requirements (↑) have to be satisfied by finding all the necessary elements with the proper features (↓), and then *their* requirements have to be satisfied, etc. The cursor controls the search, and makes sure that every need is fulfilled. Finally, the remaining morphemes have to be legitimized, such as adverbs or adjectives. An important operation is unification, which is responsible for the right matches.

Since our aim is to provide a highly detailed semantic representation, the logical choice was to proceed from the semantics: even the "syntactic" search is directed by the semantic need to find the referents which play a role in the meaning of the sentence [6]. If all the referents which are present in a lexical item's requirements (↑) can be identified with other referents in other lexical items' properties (↓), then the sentence is grammatical, and the proto-DRSs and the identity relations are listed.

In our system lexical items are morphemes (stems and affixes) for two reasons. The practical reason is effectiveness: in the case of agglutinative languages (like Hungarian) the size of the lexicon would be enormous if every possible word form were added. The other – more important – reason is theoretical: the idea of "total" lexicalism is better served by this approach (TLM, Totally Lexicalist Morphology [5]), and higher degree of universality can be achieved. TLM does not follow the usual way by having a morphological component, which first creates the words, and then syntax and semantics can operate on them. In TLM every kind of morpheme can have their own requirements and semantic content (but not all of them actually have). This way a main difference between Hungarian and English can disappear [7], namely that in Hungarian suffixes express e.g. causativity or modality, while in English separate words are responsible for the same roles (there is a similar approach for Japanese [16]).

(4)	*Énekel*	-	*tet*	-	*het*	-	*l*	-	*ek.*
	sing	-	*cause*	-	*may*	-	*2sg.obj*	-	*1sg.subj*

'*I may make you sing.*'

The "cost" of TLM is that the "usual" information is not cumulated in a word (e.g. the case of a noun), but it can be solved by rank parameters.

Using rank parameters is a crucial point of the theory, and so the implementation. Every expectation can be overridden by a stronger requirement (like in optimality theory); in other words, every requirement can be satisfied directly or indirectly (by fulfilling a stronger need). This way several phenomena can be handled easily, such as word order (see above), or case and agreement (without gathering the information of all the morphemes of a word). Consider the above mentioned construction *hasonlít a bajnok-ra* (resembles the champion-onto), for instance: one of the requirements of the verb *hasonlít* ('resemble') is to find a noun which is in sublative case (*-ra/-re*). It could be satisfied directly if a pronoun was found (e.g. *rám*, sublative form of me, onto-me). In this case direct satisfaction is not possible, since the noun *bajnok* (champion) is in nominative case; but it can be satisfied indirectly, because there is an element, the suffix *-ra* ('onto'), which is identified with the *bajnok* ('champion') by a referent, and this morpheme can meet the verb's expectation.

The implementational significance of the function σ (footnote 1) is that robustness can be achieved by using it. This function makes it possible to separate the referents of an eventuality, such as predicate, time, first argument, second argument, etc. This way we can assign semantic representations to ungrammatical or incomplete sentences as well (when producing complete condition rows fails).

The size of the database is rather small at the moment, but there are applications which do not need a large corpus. The first one we will develop is a system which can find the focus (or foci) of a sentence (the phenomena is very interesting in Hungarian, since the element is marked by a change in word order as well, not only emphasis). Our approach is different from the usual phrase-structure approach, where only a whole phrase can be the focused element. The totally lexicalist method is more appropriate since any word can be the focus (*két*, *okos*, *fiúval*):

(5) *Péter két okos fiú - val találkoz-ott.*
Peter two clever boy-with meet - past
'Peter met two clever boys.'

5 Conclusion

We have demonstrated a new "post-Montagovian" theory of interpretation, called ReALIS. After arguing for its decisive theoretical innovation (the reconciliation of compositionality and some kind of "discourse representationalism") and sketching its definition, we have shown the process of dynamic interpretation of a Hungarian sentence in this framework: how an interpreter can anchor to each other the different types of referents occurring in copies of lexical items retrieved by the interpreter on the basis of the morphemes, word and morpheme order, case and agreement markers of the sentence performed by the speaker. The last section has been devoted to the demonstration of the computational implementation of ReALIS. Due to its "totally" lexicalist (morpho-)syntactic background and ranked lexical requirements, we avoid building phrase structure trees in the course of producing semantic representations. The only operation is unification of lexical constructions. In this "abstract" approach the radical differences between languages like English and Hungarian will practically disappear.

Acknowledgement

Special thanks are due to the National Scientific Fund of Hungary (OTKA K60595) for their financial support.

References

[1] Alberti, G. (2000): Lifelong Discourse Representation Structures, *Gothenburg Papers in Computational Linguistics* 00–5, 13–20.

[2] Alberti, G. (2004): ReAL Interpretation System, in Hunyadi, L., Rákosi, Gy., Tóth, E. (eds.) *Preliminary Papers of the Eighth Symposium on Logic and Language,* Univ. of Debrecen, 1–12.

[3] Alberti, G. (2005): Accessible Referents in "Opaque" Belief Contexts, in Herzig, A., van Ditmarsch, H. (eds.) *Belief Revision and Dynamic Logic*, *ESSLLI 2005*, Heriot – Watt Univ., Edinburgh, 1–8.

[4] Alberti, G. (2008): *ℜeALIS: Interpreters in Worlds, Worlds in Interpreters*, ms. (in Hungarian), János Bolyai Research Scholarship, Hungarian Academy of Sciences.

[5] Alberti, G., Balogh, K., Kleiber, J., Viszket, A. (2002): Towards a totally lexicalist morphology, in Siptár, P., Piñón, Ch. (eds.) *Approaches to Hungarian* 9, 9–33.

[6] Alberti, G., Balogh, K., Kleiber, J., Viszket, A. (2003): Total Lexicalism and GASGrammars: A Direct Way to Semantics, in Gelbukh, A. (ed.) *Proceedings of CICLing2003*, NLCS 2588, Springer-Verlag, 37–48.

[7] Alberti, G., Kleiber, J. (2004): The *GeLexi* MT Project, in Hutchins, J. (ed.): *Proceedings of EAMT 2004 Workshop* (Malta), Foundation for Int. Studies, Univ. of Malta, Valletta, 1–10.

[8] Archangeli, D., Langendoen, T. D. (eds.) (1997): *Optimality Theory: an overview*, Blackwell, Oxford.

[9] Asher, N., Lascarides, A. (2003): *Logics of Conversation*, Cambridge Univ. Press.

[10] Barwise, J., Cooper, R. (1983): Generalized quantifiers and natural language, *Linguistics and Philosophy* 4, 159–219.

[11] Benz, A. (2000): Chains and the Common Ground, in Poesio, M., Traum, D. (eds.) *GötaLog 2000 — Gothenburg Papers in Computational Linguistics* 00–5, 181–184.

[12] Bond, F., Oepen, S., Siegel, M., Copestake, A., Flickinger, D. (2005): Open source machine translation with DELPH-IN. In: *Proceedings of the Open-Source Machine Translation Workshop at the 10th Machine Translation Summit*, Phuket, Thailand, 15–22.

[13] Butt, M., Dyvik, H., King, T. H., Masuichi, H., Rohrer, C. (2002): The Parallel Grammar project. In: *Proc. of COLING 2002 Workshop on Grammar Engineering and Eval.*, Taipei.

[14] Dekker, P. (2000): Coreference and Representationalism, in von Heusinger, K., Egli, U. (eds.) *Reference and Anaphoric Relations*, Kluwer, Dordrecht.

[15] Dowty, D. R., Wall, R. E., Peters, S. (1981): *Introduction to Montague Semantics,* D. Reidel Publishing Company, Dordrecht.

[16] Gambäck, B. (2005): Semantic Morphology. In: *Inquiries into Words, Constraints and Contexts,* CSLI Publications, Stanford, California, 204–213.

[17] Kamp, H., van Genabith, J., Reyle, U. (2004): Discourse Representation Theory, ms. to appear in *Handbook of Philosophical Logic*, source: http://www.ims.uni-stuttgart.de/~hans.

[18] Kiss, K. É., van Riemsdijk, H. (eds.) (2004): *Verb Clusters; A study of Hungarian, German and Dutch*, Linguistics Today 69, Amsterdam / Philadelphia, John Benjamins

[19] Landman, F. (1986): *Towards a Theory of Information,* Foris, Dordrecht.

[20] Partee, B. H. (1996): The Development of Formal Semantics in Linguistic Theory, in Lappin, S. (ed.) *The Handbook of Contemporary Semantic Theory*, Oxford: Blackwell, 11–38.

[21] Reyle, U. (1993): Dealing with Ambiguities by Underspecification: Construction, Representation and Deduction, *Journal of Semantics* 10, 123–179.

[22] Seligman, J., Moss, L. S. (1997): Situation Theory, in van Benthem, J., ter Meulen, A. (eds.) *Handbook of Logic and Language,* Elsevier / MIT Press, Cambridge, Mass., 239–309.

[23] Szabolcsi, A. (ed.) (1997): *Ways of Scope Taking*, SLAP 65, Kluwer, Dordrecht.

[24] Szilágyi, É. (2008): The Rank(s) of a Totally Lexicalist Syntax, in Balogh, K. (ed.) *Proceedings of the 13th ESSLLI Student Session,* 175–184.

Using Bagging and Boosting Techniques for Improving Coreference Resolution

Smita Vemulapalli
Center for Signal and Image Processing (CSIP),
School of Electrical and Computer Engineering, Georgia Institute of Technology,
Atlanta, GA 30332, USA
E-mail: smita@ece.gatech.edu

Xiaoqiang Luo, John F. Pitrelli and Imed Zitouni
IBM T.J. Watson Research Center,
Yorktown Heights, NY 10598, USA
E-mail: {xiaoluo,pitrelli,izitouni}@us.ibm.com

Keywords: coreference resolution, information extraction, classifier combination, bagging, boosting, entity detection and tracking, majority voting

Received: February 5, 2009

Classifier combination techniques have been applied to a number of natural language processing problems. This paper explores the use of bagging and boosting as combination approaches for coreference resolution. To the best of our knowledge, this is the first effort that examines and evaluates the applicability of such techniques to coreference resolution. In particular, we (1) outline a scheme for adapting traditional bagging and boosting techniques to address issues, like entity alignment, that are specific to coreference resolution, (2) provide experimental evidence which indicates that the accuracy of the coreference engine can potentially be increased by use of multiple classifiers, without any additional features or training data, and (3) implement and evaluate combination techniques at the mention, entity and document level.

Povzetek: Kombiniranje učnih algoritmov je uporabljeno za iskanje koreferenc.

1 Introduction

Classifier combination techniques have been applied to many problems in natural language processing (NLP). Popular examples include the ROVER system [Fiscus1997] for speech recognition, the Multi-Engine Machine Translation (MEMT) system [Jayaraman and Lavie2005], and also part-of-speech tagging [Brill and Wu1998, Halteren et al.2001]. Even outside the domain of NLP, there have been numerous interesting applications for classifier combination techniques in the areas of biometrics [Tulyakov and Govindaraju2006], handwriting recognition [Xu et al.1992] and data mining [Aslandogan and Mahajani2004] to name a few. Most of these techniques have shown a considerable improvement over the performance of single-classifier baseline systems and, therefore, lead us to consider implementing such a multiple classifier system for coreference resolution as well. To the best of our knowledge, this is the first effort that utilizes classifier combination techniques for improving coreference resolution.

This study shows the potential for increasing the accuracy of the coreference resolution engine by combining multiple classifier outputs and describes the combination techniques that we have implemented to establish and tap into this potential. Unlike other domains where classifier combination has been implemented, the coreference resolution application presents a unique set of challenges that prevent us from directly using traditional combination schemes [Tulyakov et al.2008]. We, therefore, adapt some of these popular yet simple techniques to suit our application, and study the results of the implementation.

The main advantage of using combination techniques is that in cases where we have multiple classification engines, we do not merely use the classifier with highest accuracy, but instead, we combine all of the available classification engines attempting to achieve results superior to the single best engine. This is based on the assumption that the errors made by each of the classifiers are not identical and therefore if we intelligently combine multiple classifier outputs, we may be able to correct some of these errors.

The main contributions of this paper are:

- *demonstrating the potential for improvement in the baseline* – By implementing a system that behaves like an oracle, where we combine the outputs of several coreference resolution classifiers with knowledge of the truth *i.e.* the correct output generated by a human, we have shown that the output of the combination of multiple classifiers has the potential to be significantly higher in accuracy than any of the individual classifiers. This has been proven in certain other areas of NLP; here, we have experimentally demonstrated the potential for this to be true in the area of coreference resolution.

- *adapting traditional bagging techniques for coreference resolution* – Multiple classifiers were generated from the same classification engine by subsampling the training-data set and the feature set. These classifiers were combined using entity-level sum rule and mention-level majority voting, after overcoming the problem of entity alignment between the classifier outputs.
- *implementing a document-level boosting algorithm for coreference resolution* – A document-level boosting algorithm was implemented in which a coreference resolution classifier was iteratively trained using a reweighted training set. Here, the training set is a set of documents, and since coreference resolution is performed for the entire document, the reweighting is done at the document level. This reweighting of the training set took into account the distribution of documents from different genres such as broadcast news, web logs and newswire articles.
- *addressing the problem of entity alignment* – To implement any combination technique for coreference resolution, we need to compensate for the fact that the number of entities and the number of mentions in each of the entities are different in the outputs of the coreference resolution classifiers to be combined. Therefore, there is the big challenge of aligning the entities before any of the traditional combination techniques may be implemented.

1.1 Organization of the paper

The remainder of this paper is organized as follows. In Section 2, we briefly describe the existing coreference resolution system and the data set used. Sections 3 and 4 present our adaptation of traditional bagging and boosting techniques. Section 5 contains an experimental evaluation of the proposed combination techniques. Section 6 discusses the related work. Finally, we conclude in Section 7 with suggestions for future work.

2 Coreference system and data

The terminologies used in the Automatic Content Extraction (ACE) task [NIST] are adopted in this paper: a *mention* is an instance of reference to an object, and the collection of mentions referring to the same object in a document form an *entity*. Coreference resolution is nothing but partitioning mentions into entities. For example, in the following sentence:

`John said Mary was his sister.`

there are four mentions: `John`, `Mary`, `his`, and `sister`. `John` and `his` belong to the same entity since they refer to the same person; so do `Mary` and `sister`. Furthermore, `John` and `Mary` are *named* mentions, `sister` is a *nominal* mention and `his` is a *pronominal* mention.

The basic coreference system is similar to the one described by Luo *et al.* [Luo et al.2004]. In such a system, the mentions in a document are processed sequentially, and at each step, a mention is either linked to one of existing entities, or used to create a new entity. At the end of this process, each possible partition of the mentions corresponds to a unique sequence of link or creation actions, each of which is scored by a statistical model. The one with the highest score is output as the final coreference result.

Experiments reported in the paper are done on the ACE 2005 data [NIST2005], which is available through the Linguistic Data Consortium (LDC). The dataset consists of 599 documents from rich and diversified sources (called *genres* in this paper), which include newswire articles, web logs, and Usenet posts, transcription of broadcast news, broadcast conversations and telephone conversations. We reserve the last 16% documents of each source as the test set, and use the rest of the documents as the training set. The number of documents, words, mentions and entities of this data split are tabulated in Table 1.

3 Bagging

One way to obtain multiple classifiers is via bagging or bootstrap aggregating, proposed by Breiman [Breiman1996] to improve the classification by combining outputs of classifiers that are trained using randomly-generated training sets. We have implemented bagging by using semi-randomly generated subsets of the entire training set and also subsets of the feature set.

3.1 Generation of multiple classifiers

In bagging, multiple classifiers are obtained by randomly generating subsets of the training set. Here, the training set refers to the set of documents that we use to train the system. When we subsample the training set, we do it at the document level.

We generated several classifiers by two techniques: the first is by semi-randomly sampling the training set and the second is by sampling the feature set. In the first technique, we try to sample the training set in a random fashion and generate a few classifiers by maintaining the initial distribution of the documents of different genres and a few others by not maintaining this distribution. In the second technique, we need to reduce the feature set and this is not done in a random fashion. Instead, we use our understanding of the individual features and also their relation to other features to decide which features may be dropped. In most of our experiments, we used classifiers in which either the training set or the feature set was subsampled, but not both.

Table 1: Statistics of ACE 2005 data: number of documents, words, mentions and entities in the training and test set.

DataSet	#Docs	#Words	#Mentions	#Entities
Training	499	253771	46646	16102
Test	100	45659	8178	2709
Total	599	299430	54824	18811

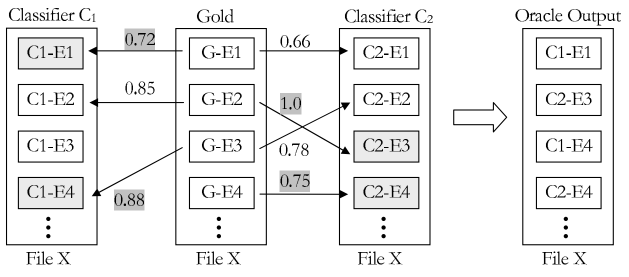


Figure 1: Working of the oracle

3.2 Oracle

In this paper, we refer to an *oracle* system which uses knowledge of the truth. In this case, truth, called *gold standard* henceforth, refers to mention detection and coreference resolution done by a human for each document. It is possible that this gold standard may have errors and is not perfect truth, but, as in most NLP systems, the human-annotated gold standard is considered the reference for purposes of evaluating computer-based coreference resolution.

To understand the oracle itself, consider an example in which we have two classifiers, and their outputs for the same input document are C_1 and C_2, as illustrated in Figure 1. The number of entities in C_1 and C_2 may not be the same and even in cases where they are, the number of mentions in corresponding entities may not be the same. In fact, even finding the corresponding entity in the other classifier or in the gold standard output G is not a trivial problem and requires us to be able to align any two classifier outputs.

The alignment between any two coreference labelings, say C_1 and G, for a document is done by finding the best one-to-one map between the entities of C_1 and the entities of G, using the algorithm explained by Luo [Luo2005]. To align the entities of C_1 with those of G, under the assumption that an entity in C_1 may be aligned with at most only one entity in G and vice versa, we need to generate a bipartite graph between the entities of C_1 and G. Now the alignment task is a maximum bipartite matching problem. This is solved by using the Kuhn-Munkres algorithm [Kuhn1955, Munkres1957]. The weights of the edges of the graph, in this case, are entity-level alignment measures. The metric we use is a relative measure of the similarity between the two entities. To compute the similarity metric $\phi\,(R,S)$ for the entity pair (R,S), we use the formula shown in Equation 1, where the intersection $(\cap)$ is the commonality with attribute-weighted partial scores. Attributes are things such as (ACE) entity type, subtype, entity class, etc.

$$\phi(R,S) = \frac{2\,|R \cap S|}{|R| + |S|} \tag{1}$$

The oracle output is a combination of the entities in C_1 and C_2 with the highest entity-pair alignment measures with the entities in the gold standard G. We can see in Figure 1 that the entity G-E1 is aligned with entities C1-E1 and C2-E1. We pick the entity with the highest entity-pair alignment measure (highlighted in gray in Figure 1) with the corresponding gold standard entity which, in this case, is C1-E1. This is repeated for every entity in G. The oracle output can be seen in the right-hand side of Figure 1. This technique can be scaled up to work for any number of classifiers.

3.3 Preliminary classifier combination approaches

Imitating the oracle. Making use of the existing framework of the oracle, we implement a combination technique that imitates the oracle except that in this case, we do not have the gold standard. If we have N classifiers C_i, $i = 1$ to N that we plan to combine, then we replace the gold standard by each of the N classifiers in succession, to get N outputs $Comb_i$, $i = 1$ to N.

The task of generating multiple classifier combination outputs that are of a significantly higher accuracy than the original classifiers is often considered to be easier than the task of finding out which of the output classifiers is highest-accuracy to pick as the final output. We used the formulas in Equations 2, 3 and 4 to assign a score S_i to each of the N combination outputs $Comb_i$ obtained, and then we pick the one with the highest score. The function Sc gives the similarity between the entities in the pair (R,S). Here, we have used the function ϕ in Equation 1 to compute the similarity between the entity-pair that forms the argument of the function Sc.

$$S_i = \frac{1}{N-1}\sum_{j \neq i} Sc(Comb_i, C_j) \tag{2}$$

$$S_i = Sc(Comb_i, C_j) \tag{3}$$

$$S_i = \frac{1}{N-1}\sum_{j \neq i} Sc(Comb_i, Comb_j) \tag{4}$$

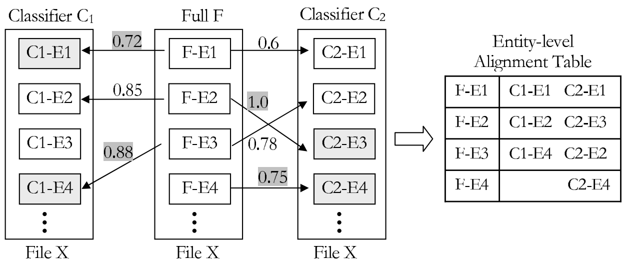


Figure 2: Entity alignment between classifier outputs

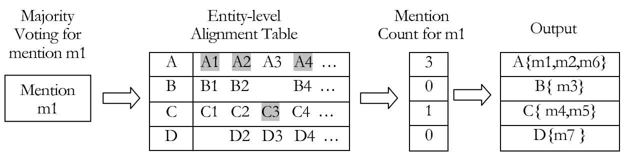


Figure 3: Mention-level majority voting

Entity-level sum-rule. We implemented a basic sum-rule at the entity level, where we generate only one combination classifier output by aligning the entities in the N classifiers and picking only one entity at each level of alignment. In the oracle, the reference for entity-alignment was the gold standard and here, since the gold standard is not available, we make use of the full system to do this. The full system is the baseline system where the entire training set and feature set have been used. The entity-level alignment has been represented as a table in Figure 2.

Let A_i, $i = 1$ to M be the aligned entities in one row of the table in Figure 2. Here, $M \leq N$ if we exclude the baseline from the combination and $M \leq N + 1$ if we include it. To pick one entity out of these M entities, we use traditional sum rule [Tulyakov et al.2008], shown in Equation 5, to compute the $S(A_i)$ for each A_i and pick the entity with the highest $S(A_i)$ value. Again, we use the function ϕ in Equation 1 to compute $Sc(A_i, A_j)$.

$$S(A_i) = \sum_{j \neq i} Sc(A_i, A_j) \tag{5}$$

3.4 Mention-level majority voting

In the previous techniques, entities are either picked or rejected as a whole but never broken down further. In the mention-level majority voting technique, we work at the mention level, so the entities created after combination may be different from the entities of all the classifiers that are being combined.

As shown in Figure 3, we have made use of the entity-level alignment table. This table is generated by aligning the entities output by the classifiers with the baseline system, as explained in the Section 3.3. In the entity-level alignment table, A, B, C and D refer to the entities in the baseline system and A1, A2, ..., D4 represent the entities of the input classifiers that are aligned with each of the baseline classifier entities. Majority voting is done by counting the number of times a mention is found in a set of aligned entities. So for every row in the table, we have a mention count. The row with the highest mention count is assigned the mention in the output. This is repeated for each mention in the document. In Figure 3, we are voting for the mention m1, which is found to have a voting count of 3 at the entity level A and a count of 1 at the entity-level of C, so the mention is assigned to the entity A as it has the majority vote. It is important to note that some entities of the classifiers may not align with any of the baseline classifier's entities as we allow only a one-to-one mapping during alignment. This leads to some entities not being a part of the alignment table. If this number is large, it may have a considerable effect on the combination.

4 Boosting

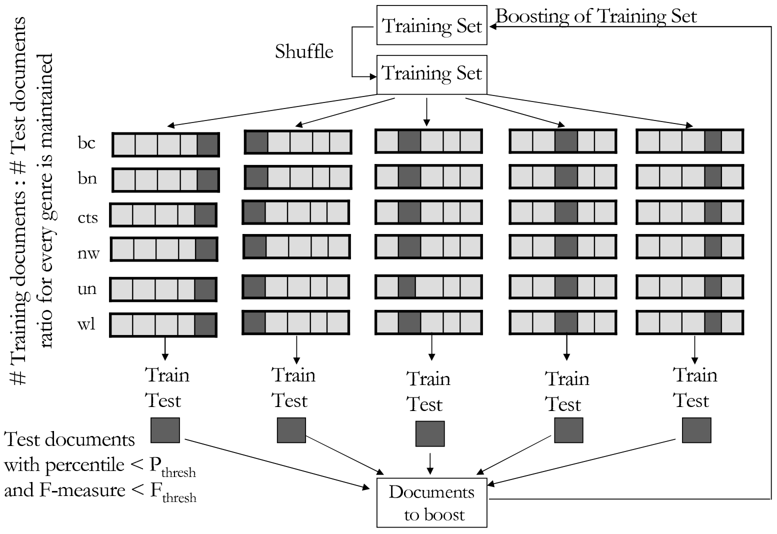
Unlike bagging techniques, the document-level boosting algorithm that we have implemented is adaptive in nature. The training set of the classifier is adaptively reweighted based on the performance of the previous classifiers. Since coreference resolution is done for a whole document, we can not split a document further. So when we reweight the training set, we are actually reweighting the documents. Figure 4 shows the overview of the document-level boosting algorithm.

The decision of which documents to boost is made using two thresholds: percentile threshold P_{thresh} and the F-measure threshold F_{thresh}. Documents in the test set that are in the lowest P_{thresh} percentile and that have a document F-measure less than F_{thresh} will be boosted in the training set for the next iteration. We shuffle the training set to create some randomness and then divide it into groups of training and test sets in a round-robin fashion such that a predetermined ratio of the number of training documents to the number of test documents is maintained. In Figure 4, the light gray regions refer to the training documents and the dark gray regions refer to the test documents. Another important consideration is that it is difficult to achieve good coreference resolution performance on documents of some genres compared to others, even if they are boosted significantly. In an iterative process, it is likely that documents of such genres will get repeatedly boosted. Also our training set has more documents of some genres and fewer of others. So we try to maintain, to some extent, the ratio of documents from different genres in the training set while splitting this training set further into groups of training and test sets.

5 Evaluation

This section describes the general setup used to conduct the experiments and presents an evaluation of the combination techniques that were implemented.

Experimental setup. The coreference resolution system used in our experiments makes use of a Maximum En-



Training Set
Boosting of Training Set
Shuffle
Training Set
# Training documents : # Test documents ratio for every genre is maintained
bc
bn
cts
nw
un
wl
Train
Test
Test documents with percentile < P_{thresh} and F-measure < F_{thresh}
Documents to boost

Figure 4: Document-level boosting

Table 2: Accuracy of the generated and baseline classifiers

Classifier	Accuracy (%)
$C_1 - C_{15}$ Average	77.52
Highest	79.16
Lowest	75.81
C_0 Baseline	78.53

tropy model which has lexical, syntactical, semantic and discourse features [Luo et al.2004]. The data set used here, which is split into the training and test sets, is part of the ACE 2005 data [NIST2005]. A short description of this data set may be found in Section 2 of this paper. For the purpose of evaluation, we make use of the human-annotated gold standard, described in Section 3.2, as the reference.

5.1 Bagging

The classifiers for the following experiments were generated using bagging techniques described in Section 3.1. A total of 15 classifiers (C_1 to C_{15}) were generated, 12 of which were obtained by semi-random sampling of the training set and the remaining 3 by sampling of the feature set. We also make use of the baseline classifier C_0, which was trained using the full training and feature sets. The accuracy of classifiers C_0 to C_{15} has been summarized in Table 2. The agreement between the generated classifiers' output was found to be in the range of 93% to 95%. In this paper, the metric used to compute the accuracy of the coreference resolution is the Constrained Entity-Alignment F-Measure (CEAF) [Luo2005] with the entity-pair similarity measure in Equation 1.

Oracle. To conduct the oracle experiment described in Section 3.2, we train 1 to 15 classifiers, whose output are aligned to the gold standard. For all system-generated entities aligned with a gold entity, we pick the one with the highest score as the output. We measure the performance for varying number of classifiers, and the result is plotted in Figure 5.

First, we observe a steady and significant increase in CEAF for every additional classifier. This is not surprising since an additional classifier can only improve the alignment score. Second, it is interesting to note that the oracle performance is 87.58% for a single input classifier C_1, *i.e.* an absolute gain of 9% compared to the baseline. This is because the availability of gold entities makes it possible to remove many false-alarm entities. Finally, the performance of the oracle output when all 15 classifiers (C_1 to C_{15}) are used as input is 94.59%, a 16.06% absolute improvement.

The oracle experiment is a "cheating" one since the gold standard is used. Nevertheless, it helps us understand the performance bound of combining multiple classifiers and the quantitative contribution of every additional classifier.

Preliminary classifier combination approaches. While the oracle result is encouraging, a natural question is. how much performance gain can be attained if the gold standard is not available. To answer this question, we replace the gold standard with one of the 15 classifiers C_1 to C_{15}, and align the rest classifiers. This is done in a round robin fashion as described in Section 3.3. The best performance of this procedure is 77.93%. The sum-rule combination output had an accuracy of 78.65% with a slightly different baseline of 78.81%. In other words, these techniques do not yield a statistically significant increase in CEAF relative to the baseline. This is not entirely surprising as the the 15 classifiers C_1 to C_{15} are highly correlated.

Mention-level majority voting. This experiment is conducted to evaluate and understand the mention-level majority voting technique for coreference resolution. Compared with the baseline, the results of this experiment are not statistically better, but they give us valuable insight into the working of the combination technique. The example in Figure 6 shows the contents of a single entity-alignment level for the full system C_0 and 3 classifier outputs C_1, C_2, and C_3 and the combination output by mention-level majority voting. The mentions are denoted by the notation *'EntityID - MentionID'*, for example *7-10* is the mention with *EntityID=7* and *MentionID=10*. Here, we use the *EntityID* in the gold file. The mentions with *EntityID=7* are "correct" *i.e.* they belong in this entity, and the others are "wrong" *i.e.* they do not belong in this entity.

The aligned system mentions are of the following four types:

- *Type I mentions* – These mentions have a highest voting count of 2 or more at the same entity alignment level and therefore appear in the output.
- *Type II mentions* – These mentions have a highest voting count of 1 and are also present in more than one input classifier. So, there is a tie between the mention counts for a single mention at different entity alignments. The rule to break the tie is that mentions are included if they are also seen in the full system C_0. As can been seen, this rule brings in correct mentions such as *7-61, 7-63, 7-64*, but it also admits *20-33,20-39* and *20-62*. This is a fundamental difference between the oracle and real experiment: in the oracle, the gold standard helps to remove entities with false-alarm mentions, while the full system output itself is noisy and it is not strong enough to reliably remove undesired mentions.
- *Type III mentions* – There is only one mention *20-66* which is of this type. It is selected in the combination output since it is present in C_2 and the baseline C_0, although it has been rejected as a false-alarm in C_1 and C_3.
- *Type IV mentions* – These mentions are false-alarm mentions (relative to C_0) and are rejected in the output.

  As can be seen, this correctly rejects mentions such as *15-22* and *20-68*, but it also rejects correct mentions *7-18, 7-19* and *7-30*.

In summary, the current implementation of the mention-level majority voting technique has a limited ability to distinguish correct mentions from wrong ones due to the noisy nature of the full system C_0 which is used for alignment. We also observe that mentions spread across different alignments often have low-count and they are often tied in count. Therefore, it is important to set a minimum threshold for accepting these low-count majority votes and also investigate better tie-breaking techniques.

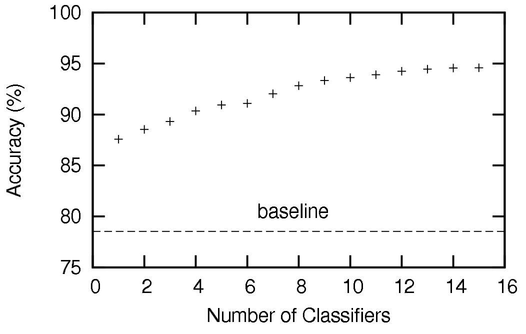


Figure 5: Oracle performance vs. number of classifiers

Table 3: Results of document-level boosting

Iteration	Accuracy (%)
1	78.53
2	78.82
3	79.08
4	78.37

5.2 Boosting.

This experiment is conducted to evaluate the document-level boosting technique for coreference resolution. Table 3 shows the results of this experiment with the ratio of the number of training documents to the number of test documents equal to 80:20, F-measure threshold F_{thresh} = 74% and percentile threshold P_{thresh} = 25%. The accuracy increases by 0.7%, relative to the baseline. Due to computational complexity considerations, we used fixed values for the parameters. Therefore, these values may be suboptimal and may not correspond to the best possible increase in accuracy.

6 Related work

A large body of literature related to statistical methods for coreference resolution is available [Ng and Cardie2003, Yang et al.2003, Ng2008, Poon and Domingos2008, McCallum and Wellner2003]. Poon and Domingos [Poon and Domingos2008] use an unsupervised technique based on joint inference across mentions and Markov logic as a representation language for their system on both MUC and ACE data. Ng [Ng2008] proposed a generative model for unsupervised coreference resolution that views coreference as an EM clustering process. In this paper, we make use of a coreference engine similar to the one described by Luo *et al.* [Luo et al.2004], where a Bell tree representation and a Maximum entropy framework are used to provide a naturally incremental framework for coreference resolution. To the best of our knowledge, this is the first effort that utilizes classifier combination techniques to improve coreference resolution. Combination techniques have earlier been applied to various applications including machine translation [Jayaraman and Lavie2005] and part-of-speech tagging [Brill and Wu1998]. However, the use of these techniques for coreference resolution presents a unique set of challenges, such as the issue of entity alignment

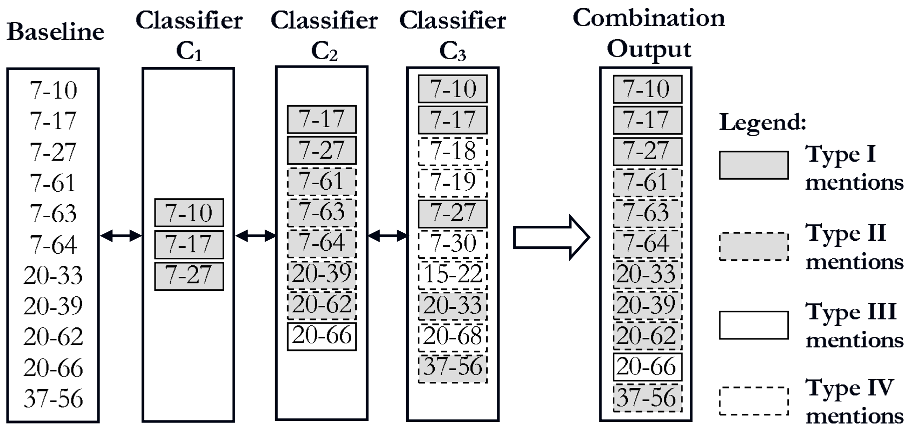


Figure 6: A real example showing the working of mention-level majority voting

between the multiple classifier outputs.

7 Conclusions and future work

This paper examined and evaluated the applicability of various bagging and boosting techniques to coreference resolution. In this paper, we also provided empirical evidence that coreference resolution accuracy can potentially be improved by making use of multiple classifiers. We proposed and evaluated new approaches to well-known classifier combination techniques that work at the mention, entity and document level. In future, we plan to work on a better alignment strategy and also explore various possibilities for improving mention-level majority voting such as setting a minimum threshold for the majority-vote and better tie-breaking. We would also like to work on further development of the document-level boosting algorithm to automatically find optimal values for the parameters that have been manually set in this paper. Another possible avenue for future work would be to test these combination techniques with other coreference resolution systems.

Acknowledgement

The authors would like to acknowledge Ganesh N. Ramaswamy for his guidance and support in conducting the research presented in this paper.

References

[Breiman1996] L. Breiman. 1996. Bagging predictors. In *Machine Learning*.

[Brill and Wu1998] E. Brill and J. Wu. 1998. Classifier combination for improved lexical disambiguation. In *Proc. of COLING*.

[Fiscus1997] J. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recogniser output voting error reduction (rover). In *Proc. of ASRU*.

[Halteren et al.2001] H. Van Halteren, J. Zavrel, and W. Daelemans. 2001. Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics*, 27.

[Jayaraman and Lavie2005] S. Jayaraman and A. Lavie. 2005. Multi-engine machine translation guided by explicit word matching. In *Proc. of ACL*.

[Kuhn1955] H. W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2.

[Luo et al.2004] X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proc. of ACL*.

[Luo2005] X. Luo. 2005. On coreference resolution performance metrics. In *Proc. of EMNLP*.

[McCallum and Wellner2003] A. McCallum and B. Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proc. of IJCAI/IIWeb*.

[Munkres1957] J. Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1).

[Ng and Cardie2003] V. Ng and C. Cardie. 2003. Bootstrapping coreference classifiers with multiple machine learning algorithms. In *Proc. of EMNLP*.

[Ng2008] V. Ng. 2008. Unsupervised models for coreference resolution. In *Proc. of EMNLP*.

[NIST2005] NIST. 2005. ACE'05 evaluation. `www.nist.gov/speech/tests/ace/ace05/index.html`.

[Poon and Domingos2008] H. Poon and P. Domingos. 2008. Joint unsupervised coreference resolution with Markov Logic. In *Proc. of EMNLP*.

[Schapire1999] R.E. Schapire. 1999. A brief introduction to boosting. In *Proc. of IJCAI*.

[Tulyakov et al.2008] S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann. 2008. Review of classifier combination methods. In *Machine Learning in Document Analysis and Recognition*.

[Yang et al.2003] X. Yang, G. Zhou, J. Su, and C. L. Tan. 2003. Coreference resolution using competition learning approach. In *Proc. of ACL*.

[Aslandogan and Mahajani2004] Y.A. Aslandogan and G.A. Mahajani. 2004. Evidence combination in medical data mining. In *Proc. of ITCC*, volume 2.

[Tulyakov and Govindaraju2006] Sergey Tulyakov and Venu Govindaraju. 2006. Classifier combination types for biometric applications. In *Proc. of CVPR Workshop*.

[Xu et al.1992] L. Xu, A. Krzyzak, and C.Y. Suen. 1992. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3), May/Jun.

[NIST] NIST. The ACE evaluation plan. `www.nist.gov/speech/tests/ace/index.html`.

Cascaded Regression Analysis Based Temporal Multi-document Summarization

Ruifang He, Bing Qin, Ting Liu, Sheng Li
Information Retrieval Lab, Harbin Institute of Technology P.O.Box 321, HIT,P.R.China 150001
E-mail: rfhe@ir.hit.edu.cn, http://ir.hit.edu.cn/

Keywords: temporal multi-document summarization, temporal semantic labeling, macro importance discriminative model, micro importance discriminative model

Received: February 5, 2009

Temporal multi-document summarization (TMDS) aims to capture evolving information of a single topic over time and produce a summary delivering the main information content. This paper presents a cascaded regression analysis based macro-micro importance discriminative model for the content selection of TMDS, which mines the temporal characteristics at different levels of topical detail in order to provide the cue for extracting the important content. Temporally evolving data can be treated as dynamic objects that have changing content over time. Firstly, we extract important time points with macro importance discriminative model, then extract important sentences in these time points with micro importance discriminative model. Macro and micro importance discriminative models are combined to form a cascaded regression analysis approach. The summary is made up of the important sentences evolving over time. Experiments on five Chinese datasets demonstrate the encouraging performance of the proposed approach, but the problem is far from solved.

Povzetek: Metoda kaskadne regresije je uporabljena za izdelavo zbirnega besedila.

1 Introduction

Multi-document summarization is a technology of information compression, which is largely an outgrowth of the late twentieth-century ability to gather large collections of unstructured information on-line. The explosion of the World Wide Web has brought a vast amount of information, and thus created a demand for new ways of managing changing information. Multi-document summarization is the process of automatically producing a summary delivering the main information content from a set of documents about an explicit or implicit topic, which helps to acquire information efficiently. It has drawn much attention in recent years and is valuable in many applications, such as intelligence gathering, hand-held devices and aids for the handicapped.

Temporal multi-document summarization (TMDS) is the natural extension of multi-document summarization, which captures evolving information of a single topic over time. The greatest difference from traditional multi-document summarization is that it deals with the dynamic collection about a topic changing over time. It is assumed that a user has access to a stream of news stories that are on the same topic, but that the stream flows rapidly enough that no one has the time to look at every story. In this situation, a person would prefer to dive into the details that include the most important, evolving concepts within the topic and have a trend analysis.

The key problem of summarization is how to identify important content and remove redundant content. The common problem for summarization is that the information in different documents inevitably overlaps with each other, and therefore effective summarization methods are needed to contrast their similarities and differences. However, the above application scenarios, where the objects to be summarized face to some special topics and evolve with time, raise new challenges to traditional summarization algorithms. One challenge for TMDS is that the information in summary must contain the evolving content. So we need to effectively take into account this temporally evolving characteristics during the summarization process. Thus a good TMDS must include information as much as possible, keeping information as novel as possible. In this paper, we focus on how to summarize the series news reports by the generic and extractive way.

Considering the temporal characteristic of the series news reports at different levels of topic detail, redundancy is a good feature. We adopted cascaded regression analysis to model the temporal redundancy from the macro and micro view. We hierarchically extract important information with the macro and micro importance discriminative models. We detected the important time points based on macro importance discriminative model, and extracted the important sentences based on micro importance discriminative model. Macro and micro importance discriminative models are combined to form a cascaded regression analysis model. This method not only reduces the complexity of the problem, but also fully mines the temporal characteristics of evolving data over time. The summary is made up of the important sentences evolving over time. Experiments on five Chinese datasets demonstrate the encouraging performance of the proposed approach, but the problem is far from solved.

The rest of this paper is organized as follows: Section 2 introduces related work. The details of the proposed approach are described in Section 3. Section 4 presents and discusses the evaluation results. We conclude this paper and discuss future work in Section 5.

2 Related work

Temporal summary is a relatively new research direction, which originates from text summarization and topic detection and tracking (TDT). It is also related to time line construction techniques. Alan et al.[1] firstly put forward the concept of temporal summary inspired by TDT in SIGIR2001. Given a sequence of news reports on certain topic, they extract useful and novel sentences to monitor the changes over time. Usefulness is captured by considering whether a sentence can be generated by a language model created from the sentences seen to date. Novelty is captured by comparing a sentence with prior sentences. They report that it is difficult to combine the two factors successfully. Other researchers exploit distribution of events and extract the hot topics on time line by statistical measures. Swan and Allan[8] employ χ^2 statistics to measure the strength that a term is associated with a specified date, and then extract and group important terms to generate “topics" defined by TDT. In [3], Chen et al. import the aging theory to measure the “hotness" of a topic by analyzing the temporal characteristic of news report. The aging theory implies that a news event can be considered as a life form that goes through a life cycle of birth, growth, decay, and death, reflecting its popularity over time. Then hot topics are selected according to energy function defined by aging theory. Lim et al.[5] anchor documents on time line by the publication dates, and then extract sentences from each document based on surface features. Sentence weight is adjusted by local high frequency words in each time slot and global high frequency words from all topic sentences. They evaluate the system on Korean documents and report that time can help to raise the percentage of model sentences contained in machine generated summaries. Jatowt and Ishizuka[2] investigate the approaches to monitor the trends of dynamic web documents. They employ a simple regression analysis on word frequency and time to identify whether terms are popular and active. The importance of a term is measured by its slope, intercept and variance. The weight of a sentence is measured by the sum of the weights of the terms inside the sentence. The sentences with highest scores are extracted into a summary. However, they do not report any quantitative evaluation results. In [7], Mani is devoted to temporal information extraction, knowledge representation and reasoning, and try to apply them to multi-document summarization. In [4], Li et al. explore whether the temporal distribution information helps to enhance event-based summarization based on corpus of DUC2001.

Due to different tasks, the above researches do not uniformly incorporate the temporal characteristics. While macro and micro importance discriminative models based on cascaded regression analysis approach can mine the temporal characteristics at different levels of topic detail and produce summary.

3 Cascaded regression analysis approach

We know news has strong temporal characteristic. If there is a novel happening, many websites will concern it, and naturally produce vast relevant news reports, this time point would be very important at the moment. If happening is gradually disappearing, the relevant new reports will also decrease accordingly, whose importance would reduce. From macro view, redundancy implies the whole trends of the happening, which also hints the more finer progress from micro view. We model the temporal redundancy from macro and micro view, and integrate macro and micro importance discriminative models into a cascaded regression analysis framework for the content selection for TMDS. Algorithm 1 shows the basic steps.

Algorithm 1 Framework for Temporal Multi-document Summarization based on Cascaded Regression Analysis

```
Input: Stream of Chinese series news stories within the same topic; Output: Sentences containing important events evolving over time;
1: Parse the documents into the set of the sentences, and recognize and resolve the time expressions contained in sentences;
2: Construct the article/sentence count-time distribution curve;
3: Convert the curve given by step 2 to be the relative importance curve of the time points;
4: Extract the important time points with macro importance discriminative model;
5: Extract important events with micro importance discriminative model in each important time point;
6: Rank sentences according to the publication time and the real time;
```

3.1 Selection of content unit

Since series news reports are made up of time points, each time point consists of articles/sentences. Here, the ith time point t_i, the jth sentence s_j is formalized as the following, respectively: $t_i = \{s_j\}$,
$s_j = \{T_p, T_r, Trigger, Scope\}, i, j = 1...n$, Tp is publication time, Tr is real time through time resolution,

Trigger is the set of trigger words, and *Scope* is the sentence description containing events. Reference to the definition of event in ACE evaluation[1], a trigger word indicates the existence of an event. However, Chinese event extraction technology is not mature, we ignore the relevant attributes of event, including type, subtype, modality, polarity, genericity and tense. Generally, trigger word is verb or action noun. We just consider the situation of verb so as to simplify the question. Thus, the j th sentence can also be simply formalized as follows: $s_j = \{v_k\}, j, k = 1...n$.

The importance of a sentence depends on the importance of the verbs contained in a sentence. Based on the above analysis, we choose the time point and verb as the content unit of importance discrimination from macro and micro view, respectively.

3.2 Macro importance discriminative model

In the new World Wide Web environment, the number of news articles is increasing dramatically, and we can conveniently and instantaneously get the rich data. Usually, the reports about the same story from the different websites are mostly similar, especially the start and end time points and the important time points. With the evolution of an abrupt news story, the number of the news articles or events will form a distribution curve along the time axis. From the intuitive observation of macro view, this temporal characteristic of news articles gives us a good illumination on TMDS.

For example, figure 1 shows the temporal trends about the number of the news reports on the topic *Solomon turbulence* from the Sina. The horizontal axis is time, and the vertical axis is the number of articles or sentences. For the benefit of clearly observing the temporal characteristics of the curve and comparing the difference in extracting the important time points between articles and sentence count-time distribution curve, we convert it into the relative importance curve of time point and enlarge 100 times. This transformation can further help us modify the choice of the important time points. The concrete method is as follows: the relative importance value of time point is computed by the ratio of the article/sentence count in each time point to the article/sentence count in the highest peak. Figure 2 shows the curve through transformation.

According to this kind of distribution curve and our intuition, we give the macro importance discriminative model, including one assumption and one definition.

Assumption 1: The start and end time points, and the time points having more documents contain important information with a high probability. Valleys and slowly changing time points contain unimportant information with a high probability.

Definition of Slowly changing time point: If the left slope and right slope of the current time point are both lower than (we empirically assign 2 to λ), we say this time point changes slowly. They are defined as follows:

[1]ACE2007 evaluation plan:
http://projects.ldc.upenn.edu/ace/intro.html

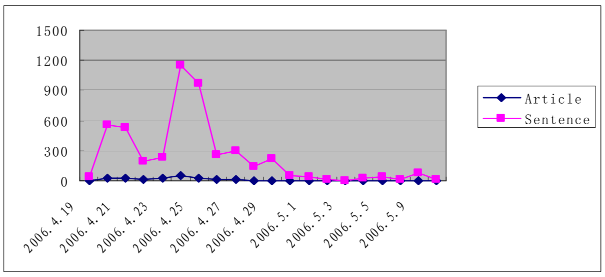


Figure 1: Article/Sentence count-time distribution about the Solomon turbulence news report.

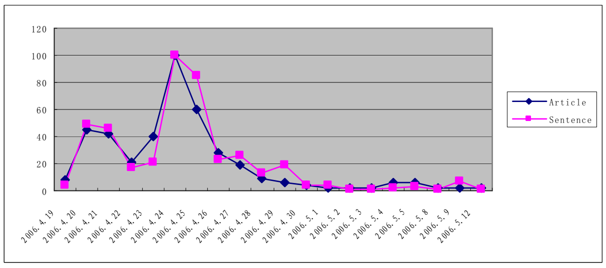


Figure 2: Relative importance distribution curve of time points.

$$\mathrm{left\_Slope}(t_i) = \frac{\mathrm{RI(left)}}{\mathrm{Td(left)}} \tag{1}$$

$$\mathrm{right\_Slope}(t_i) = \frac{\mathrm{RI(right)}}{\mathrm{Td(right)}} \tag{2}$$

$\mathrm{RIleft} = \mathrm{I(t_i)} - \mathrm{I(t_{i-1})}$, $\mathrm{RI(right)} = \mathrm{I(t_{i+1})} - \mathrm{I(t_i)}$,$\mathrm{Td(left)} = \mathrm{date(t_i)} - \mathrm{date(t_{i-1})}$, $\mathrm{Td(right)} = \mathrm{date(t_{i+1})} - \mathrm{date(t_i)}$, $\mathrm{i} = 2...\mathrm{n}$, $\mathrm{right\_Slope}(t_i)$, $\mathrm{right\_Slope}(t_i)$ are formed by the current time point, the left adjacent one, and the right adjacent one. $\mathrm{RI(left)}$, $\mathrm{RI(right)}$ is the relative importance difference value computed by the current time point, the left adjacent one, and right adjacent one, respectively. $\mathrm{I(t_i)}$ is the relative importance value of the ith time point. $\mathrm{Td(left)}$, $\mathrm{Td(right)}$ is the duration that the left one, the right one deviates from the current one, respectively. $\mathrm{date(t_i)}$ is the date of the ith time point.

Based on the above description, we give the algorithm of detecting the important time points with importance discriminative model.

3.3 Micro importance discriminative model

In order to extract the important sentences, we need to define the importance scoring scheme. Trigger words are core representatives of event, whose importance can reflect the importance of sentence. Therefore, we statistically analyze the importance of trigger words and define three kinds of scoring schemes as follows:

Algorithm 2 Detecting the important time points

```
Input: all time points; Output: the importance points;
1: Use climbing algorithm to find all the peaks and valleys, and keep the start and end time points;
2: Remove the valleys and the slowly changing time points, and get four time point sets figured out by the two kinds of relative importance distribution curve of time points;
3: Compute the intersection of four time point sets, then get the important time point set;
```

TFIOF based scoring scheme Depending on the basic idea of the feature weight about TFIDF, TFIOF is proposed to compute the importance of a trigger. $\forall t_i, i, k = 1...n$,

$$tfiof(v_k) = tf_i(v_k) \times log\frac{n}{of(v_k)} \tag{3}$$

$tf_i(v_k)$ is the occurrence number of v_k in the current time point. n is the number of all time points, $of(v_k)$ is the number of time points containing v_k.

Slope based scoring scheme $\forall t_i$, t_{i-1}, $i = 2...n$, $k = 1...n$,

$$left\_Slope(v_k) = \frac{tf_i(v_k) - tf_{i-1}(v_k)}{Td(left)} \tag{4}$$

$tf_{i-1}(v_k)$ denotes the occurrence number of v_k in the left adjacent time point. $left\_Slope(v_k)$ is the instantaneous left slope of the event on behalf of v_k, which adopts the linear regression method to express the consequent relation between time and event variables. If it is a positive value, it means that the event triggered by v_k is emerging, or the past event triggered by v_k is disappearing. It also shows whether this trigger word is active in the local scope.

Variance analysis based scoring scheme The model for the arrival of trigger words can be considered as a random process, and the arrival of every trigger word is a random variable. The variance value of the random variable X on behalf of v_k is represented as $Variance(X) = E\{(X - E(X))^2\}$ It represents the average magnitude of X in term of importance for a period of time, which helps us to detect the important trigger words in the global scope. The larger the $Variance(X)$ is, the more precious the trigger word. Every trigger word has a variance value. In order to compare the importance of trigger words in each time point, we normalize every random variable as X^*:

$$X^* = \frac{X - E(X)}{\sqrt{Variance(X)}} \tag{5}$$

Based on the importance scoring schemes, we give the algorithm of the important sentences extraction and ranking, see algorithm 3.

Algorithm 3 Selection and ranking of sentences

```
Input: Stream of series news reports through preprocessing; Output: Assume $s_i$ to be the set of final summary in $t_i$, initially $s_i = \phi$;
1: Extract the trigger words;
2: Compute the importance weight of each trigger word according to three scoring schemes;
3: Rank the trigger words according to the weight from step 2, respectively;
4: repeat
5:   For each time point, according to each ranked $v_k$, select the most important, but not redundant sentence including $v_k$ with highest weight sum to add into $s_i$
6:   if $v_k \subset s_i$ then
7:     Compute the value of the goal function: $f(v_k) = \frac{|s_i| \cup |s_j|}{N} - \frac{|s_i|}{N}$
8:   end if
9:   if $f(v_k) \leq \lambda(\lambda = 2/N)$ then
10:    $s_j$ is the redundant sentence, where N is the number of trigger words in the $i$th time point
11:  end if
12: until Summary length is satisfied
13: Rank the sentences within different time points by their publication date, and rank the sentences within the time points by their real date; If the two sentences have the same real date, we donot care their relative rank;
```

4 Experiments

4.1 Corpus and evaluation metrics

TMDS is a new research, and there is no public corpus and evaluation metric. Therefore we have to build the corpus and the evaluation metric.

Corpus Our Chinese corpus construction includes two parts, one is the construction of raw corpus, another is the construction of reference summary. Five groups of Chinese data set are chosen from Sina's[2]international news topics between 2005 and 2006. Table1 illustrates the settings of the corpus, where there are five topics, 78 time points, 734 articles, 13486 sentences. Simultaneously, for each date set, we let experts annotate three groups of reference summary in term of the compression rate 10% and 20%.

ID	#time points	#articles	#sentences
1	20	214	4310
2	25	250	5253
3	3	17	101
4	20	158	2278
5	10	95	1544

Table 1: Corpus settings

Evaluation Metrics ROUGE[6] is used as the evalua-

[2]http://news.sina.com.cn

tion metric, which has been widely adopted by DUC for automatic summarization evaluation. It measured summary quality by counting overlapping units such as the n-gram, word sequences and word pairs between the candidate summary and the reference summary. ROUGE toolkit reported separate scores for 1, 2, 3 and 4-gram, and also for longest common subsequence co-occurrences and so on. However, this evaluation metrics faces to English. Based on it, we develop a Chinese-style ROUGE-N evaluation tool 'CROUGE-N'. The evaluation measure is F score:

$$F = \frac{\sum_{i=1}^{i=n} F_i}{n}, F_i = \frac{2P_iR_i}{P_i + R_i} \tag{6}$$

$i = 1...n$, i denotes the number of reference summary. P_i, R_i, F_i is Precision, Recall and F score, respectively.

4.2 Experiment results and analysis

Two groups of experiments are designed to validate the performance of hierarchical regression analysis approach for TMDS.

Experiment1: Micro importance discriminative model

In the first experiment, we use the micro importance discriminative model to produce the summaries under compression rate (CRate) 10% and 20%.

CRate	TFIOF	Slope	Variance
10%	17.29%	19.11%	14.81%
20%	27.18%	27.56%	27.02%

Table 2: Performance of content selection from micro view with CROUGE-4

The goal of Experiment 1 is to evaluate the performance of different micro importance discriminative modelsčňwhich extracts the important content from the fine particle scale. Table 2 shows that slope based micro model has the best performance whatever compression rate is 10% or 20%. It is because that this model can better represent the instantaneously temporal characteristics of series news reports. The performance of variance based micro model is the lowest, however, it is still meaningful. Variance of random variable indicates the average changing magnitude of its importance. It can capture the importance information from the global scope. While TFIOF based micro model is easy to be implemented and can extract important information from local and global scope. Three models observe micro information from different views, respectively. We can adopt the different micro models according to different practical applications. The more effective model incorporating their merits will be explored in the future. When the compression rate is 20%, the performance difference between three models is little. It shows that more ordinary sentences are added into the summary, while our models is apt to capture the particular sentences.

Experiment2: Macro and micro importance discriminative model

Based on the best micro importance discriminative model adopting slope scoring scheme, we further validated the performance of hierarchically extracting important information with macro and micro importance discriminative model.

The experiment results from Table 3 validated that macro and micro importance discriminative model displays the better system performance than the single micro model. The linear regression based macro and micro importance discriminative model that we adopted receives the best performance. Macro model is used to extract the important time points, which helps to have a coarse content selection. In the whole process, we try to mine the temporal characteristic of the articles, events and terms from the macro and micro view, and use the regression analysis to summarize the relationship between the time and the frequency of articles, sentences and terms. Macro importance discriminative model and micro importance discriminative model have the recursive properties to some extent. No matter what the slope used to select important time points or the slope used to extract important trigger words, their slope value from the regression different from zero represents the evolving trends of the series news. If it has a positive value, it means that the event abruptly happens, or the event is disappearing.

Though our system performance cannot directly be compared with that of Document Understanding Conference (DUC), it still has the similar performance trend. Our CROUGE-2 score is higher than CROUGE-4 score, and it is reasonable. Because of the limitation of space, our approach 's CROUGE-2 score wasn't listed here.

CRate	Micro	Macro+Micro
10%	19.11%	20.46%
20%	27.56%	29.13%

Table 3: Performance of content selection from macro and micro view with CROUGE-4

5 Conclusion and future work

This paper tries to explore the optimization content selection model for temporal multi-document summarization from different levels of topic detail. We mine the temporal characteristics of articles count, sentences count and events count with a topic changing over time, and proposed a cascaded regression analysis based macro and micro importance discriminative model to guide the content selection. This model not only reduces the complexity of the problem, but also could fully use the temporal characteristics from different levels of topic detail. However, since there are no public evaluation corpus and metrics for temporal multi-document summarization, our approach cannot compare with others.

In the future, we will explore the more effective model from information fusion view. Considering series news report has the strong temporal characteristic, we will further use the techniques of temporal text mining and temporal information extraction to improve the system performance. We also hope to do some contributions for Chinese temporal multi-document summarization evaluation.

References

[1] J. Allan, R. Gupta, and V. Khandelwal. Temporal summaries of new topics. Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, pages 10-18, 2001.

[2] A. Jatowt and M. Ishizuka. Temporal Web Page Summarization. 5th International Conference On Web Information Systems Engineering, Brisbane, Australia, November 22-24, 2004.

[3] C. Kuan-Yu, L. LUESUKPRASERT, and T. Seng-cho. Hot Topic Extraction Based on Timeline Analysis and Multidimensional Sentence Modeling. IEEE Transactions on Knowledge and Data Engineering, pages 1016-1025, 2007.

[4] M. L. Q. W. K. Li, W.J.and Wu. Integrating temporal distribution information into event-based summarization. International Journal of Computer Processing of Oriental Languages, 19:201-222, 2006.

[5] J. Lim, I. Kang, J. J.Bae, and J. Lee. Sentence extraction using time features in multi-document summarization. In Proceedings of the Asia Information Retrieval Symposium, pages 82-932, 2004.

[6] C. Lin. ROU2GE: A Package for Automatic Evaluation of Summaries. Proceedings of the Workshop on Text Summarization Branches Out, pages 25-26, 2004.

[7] I. Mani. Recent Developments in Temporal Information Extraction. Nicolov, N., and Mitkov, R. Proceedings of RANLP, 3, 2004.

[8] R. Swan and D. Jensen. Constructing Topic-Specific Timelines with Statistical Models of Word Usage. Proceedings of the 6th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD), pages 73-80, 2000.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 800 staff, has 600 researchers, about 250 of whom are postgraduates, nearly 400 of whom have doctorates (Ph.D.), and around 200 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of **Slove**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

From the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

Part of the Institute was reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project was developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park is a shareholding company hosting an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Higher Education, Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of the Economy, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 251 93 85
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Public relations: Polona Strnad

INFORMATICA
AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS
INVITATION, COOPERATION

Submissions and Refereeing

Please submit an email with the manuscript to one of the editors from the Editorial Board or to the Managing Editor. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible from typing errors to global philosophical disagreements. The chosen editor will send the author the obtained reviews. If the paper is accepted, the editor will also send an email to the managing editor. The executive board will inform the author that the paper has been accepted, and the author will send the paper to the managing editor. The paper will be published within one year of receipt of email with the text in Informatica MS Word format or Informatica LaTeX format and figures in .eps format. Style and examples of papers can be obtained from http://www.informatica.si. Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the managing editor.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1000 Ljubljana, Slovenia. E-mail: drago.torkar@ijs.si

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than sixteen years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: ....

Title and Profession (optional): ....

....

Home Address and Telephone (optional): ....

....

Office Address and Telephone (optional): ....

....

E-mail Address (optional): ....

Signature and Date: ....

Informatica WWW:

http://www.informatica.si/

Referees:

Informatica

An International Journal of Computing and Informatics

Web edition of Informatica may be accessed at: http://www.informatica.si.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2010 (Volume 34) is
– 60 EUR for institutions,
– 30 EUR for individuals, and
– 15 EUR for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

Typesetting: Borut Žnidar.
Printing: Dikplast Kregar Ivan s.p., Kotna ulica 5, 3000 Celje.

Orders may be placed by email (drago.torkar@ijs.si), telephone (+386 1 477 3900) or fax (+386 1 251 93 85). The payment should be made to our bank account no.: 02083-0013014662 at NLB d.d., 1520 Ljubljana, Trg republike 2, Slovenija, IBAN no.: SI56020830013014662, SWIFT Code: LJBASI2X.

Informatica is published by Slovene Society Informatika (president Niko Schlamberger) in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)
ACM Slovenia (Dunja Mladenič)

Informatica is surveyed by: ACM Digital Library, Citeseer, COBISS, Compendex, Computer & Information Systems Abstracts, Computer Database, Computer Science Index, Current Mathematical Publications, DBLP Computer Science Bibliography, Directory of Open Access Journals, InfoTrac OneFile, Inspec, Linguistic and Language Behaviour Abstracts, Mathematical Reviews, MatSciNet, MatSci on SilverPlatter, Scopus, Zentralblatt Math

The issuing of the Informatica journal is financially supported by the Ministry of Higher Education, Science and Technology, Trg OF 13, 1000 Ljubljana, Slovenia.

Volume 34 Number 1 March 2010 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Informatica **34** (2010) Number 1, pp. 1–125