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.

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

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

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:

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

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.