Improving Morphosyntactic Tagging of Slovene Language through Meta-tagging

Jan Rupnik, Miha Grčar and Tomaž Erjavec
Jožef Stefan Institute, Jamova cesta 39, Ljubljana
E-mail: {jan.rupnik, miha.grcar, tomaz.erjavec}@ijs.si, http://kt.ijs.si

Keywords: PoS tagging, meta-tagger, Slavic languages, FidaPLUS, JOS corpus, machine learning, Orange, decision trees, CN2 rules, Naive Bayes

Received: July 12, 2009

Part-of-speech (PoS) or, better, morphosyntactic tagging is the process of assigning morphosyntactic categories to words in a text, an important pre-processing step for most human language technology applications. PoS-tagging of Slovene texts is a challenging task since the size of the tagset is over one thousand tags (as opposed to English, where the size is typically around sixty) and the state-of-the-art tagging accuracy is still below levels desired. The paper describes an experiment aimed at improving tagging accuracy for Slovene, by combining the outputs of two taggers – a proprietary rule-based tagger developed by the Amebis HLT company, and TnT, a tri-gram HMM tagger, trained on a hand-annotated corpus of Slovene. The two taggers have comparable accuracy, but there are many cases where, if the predictions of the two taggers differ, one of the two does assign the correct tag. We investigate training a classifier on top of the outputs of both taggers that predicts which of the two taggers is correct. We experiment with selecting different classification algorithms and constructing different feature sets for training and show that some cases yield a meta-tagger with a significant increase in accuracy compared to that of either tagger in isolation.

Povzetek: V članku je opisano označevanja slovenskih besedil z združevanjem Amebisovega označevalnika in označevalnika TnT.

1 Introduction

Morphosyntactic tagging, also known as part-of-speech tagging or word-class syntactic tagging is a process in which each word appearing in a text is assigned an unambiguous tag, describing the morphosyntactic properties of the word token. Such tagging is the basic pre-processing step for a number of applications or more advanced analysis steps, such as syntactic parsing. Morphosyntactic tagging is, in general, composed of two parts: the program first assigns, on the basis of a morphological lexicon all the possible tags that a word form can be associated with (morphological look-up), and then chooses the most likely tag on the basis of the context in which the word form appears in the text (disambiguation). For words not appearing in the lexicon, various taggers either ignore them or employ heuristics to guess at their tag.

Unlike English, morphologically richer Slavic languages such as Czech (Hajič and Hladka, 1998) or Slovene typically distinguish more than a thousand morphosyntactic tags. In the multilingual MULTEXT-East specification (Erjavec, 2004) almost 2,000 tags (morphosyntactic descriptions, MSDs) are defined for Slovene. MSDs are represented as compact strings, with positionally coded attribute values, so they effectively serve as shorthand notations for feature-structures. For example, the MSD `Agufpa` expands to `Category = Adjective, Type = general, Degree = undefined, Gender = feminine, Number = plural, Case = accusative`.

Having such a large number of tags makes assigning the correct one to each word token a much more challenging task than it is e.g. for English. The problem for Slovene has been exacerbated by the lack of large and available validated tagged corpora, which could serve as training sets for statistical taggers.

Recently, new annotated language resources have become available for Slovene. FidaPLUS[1] (Arhar & Gorjanc, 2007) is a 600 million word monolingual reference corpus automatically annotated with MULTEXT-East MSDs by the Slovene HLT company Amebis[2]. But while FidaPLUS is freely available for research via a Web concordancer, it is not generally available as a dataset. In order to remedy the lack of publicly available annotated corpora for HLT research on Slovene, the JOS project (Erjavec and Krek, 2008) is making available two corpora under the Creative Commons license. Both contain texts sampled from FidaPLUS, with the smaller jos100k containing 100,000 words with fully validated morphosyntactic annotations, and the larger, jos1M having 1 million words, and partially hand validated annotations – project resources preclude fully validating the latter.

Previous experiments (Erjavec et al., 2000) showed that from various publicly accessible taggers the best

[1] http://www.fidaplus.net/
[2] http://www.amebis.si/

results were achieved by TnT (Brants, 2000). TnT is a Hidden Markov Model tri-gram tagger, which also implements an unknown-word guessing module. It is fast in training and tagging, and is able to accommodate the large tagset used by Slovene.

Having the validated jos100k at our disposal, we experimented with training TnT and seeing how its errors compare to the ones assigned by the Amebis tagger. It turned out that the two taggers are comparable in accuracy, but make different mistakes. This gave us a method of selecting the words that should be manually corrected in jos1M – only those tokens where the annotations between the taggers differ were selected for manual inspection. This approach concentrated on validating the words where state-of-the-art taggers are still able to make correct decisions, at the price of ignoring cases where both taggers predict the same but incorrect tag, i.e. the truly difficult cases.

Having several automatically tags for each word also offers the possibility of combining their outputs in order to increase accuracy, say, over the whole FidaPLUS corpus. Experiments in combining PoS taggers have been attempted before, using various learning strategies, and for various languages, e.g. voting, stacking, etc. for Swedish (Sjöbergh, 2003) or multi-agent systems for Arabic (Othmane Zribi et al., 2006). An experiment, more similar to ours, is reported in Spoustová et al. (2007) for Czech, also using a rich positional tagset, where several stochastic taggers are combined with a rule based one; the rule based tagger is used predominantly as a pre-disambiguation step, to filter out unacceptable tags from the ambiguity classes of the tokens.

This paper presents a similar experiment, which, however, uses only two independent taggers therefore precluding combination methods such as voting or pipelining. But as in the Czech case, we also need to deal with a very large and positionally encoded tagset.

The rest of this paper is structured as follows: Section 2 presents the dataset used in the experiments, Section 3 explains the methods used to combine the output of the taggers, Sections 4 and 5 give the results of experiments on the jos100k and jos1M corpora with different methods and features, and Section 6 gives the conclusions and directions for further work.

2 Dataset

The dataset used in the first set of experiments is based on the jos100k corpus; the corpus contains samples from almost 250 texts from FidaPLUS, cca. 1,600 paragraphs or 6,000 sentences. The corpus has just over 100,000 word tokens, and, including punctuation, 120,000 tokens. jos100k contains only manually validated MSDs, of which 1,064 different ones appear in the corpus.

For the dataset we added MSDs assigned by Amebis and TnT to the manually assigned ones. Two sentences from the dataset are given in Figure 1. Annotations marking texts and paragraphs have been discarded and end of sentence is marked by an empty line. Punctuation is tagged with itself.

Prišlo	Vmep-sn	Vmep-sn	Vmep-sn
je	Va-r3s-n	Va-r3s-n	Va-r3s-n
do	Sg	Sg	Sg
prerivanja	Ncnsg	Ncnsg	Ncnsg
in	Cc	Cc	Cc
umrla	Vmep-sf	Vmep-sf	Vmep-sf
je	Va-r3s-n	Va-r3s-n	Va-r3s-n
.	.	.	.
Tega	Pd-nsg	Pd-msa	Pd-msg
se	Px------c	Px------c	Px------c
sploh	Q	Q	Q
nisem	Va-r1s-y	Va-r1s-y	Va-r1s-y
zavedel	Vmep-sm	Vmep-sm	Vmep-sm
.	.	.	.

Figure 1: Example stretch of the corpus dataset ("*Prišlo je do prerivanja in umrla je. Tega se sploh nisem zavedel.*"). First column is the word-form, second the gold standard manually assigned tag, third the one assigned by TnT, and the fourth by Amebis. Note the first word of the second sentence, where both taggers make a mistake.

The source FidaPLUS corpus also contains, for each word token, all possible MSDs that could be assigned to it, i.e. its ambiguity class. Based on this information, we computed the average per-word MSD ambiguity which turns out to be 3.13 for the jos100k corpus. So, on the average, a tagger needs to choose the correct MSD tag between three possibilities. Note that disambiguation is only possible for known words.

2.1 Amebis MSDs

The Amebis MSDs were taken from the source FidaPLUS corpus; as mentioned, the Amebis tagger is largely a rule-based one, although with heuristics and quantitative biases. The tagger uses a large lexicon, leaving only 2% of the word tokens in jos100k unknown. Amebis doesn't tag these words, and they have all been given a distinguished PoS/MSD "unknown". Furthermore, FidaPLUS is annotated according to the MULTEXT-East specification, while the JOS corpus uses a modification, based on, but different from the MULTEXT-East/FidaPLUS one. Differences concern reordering of attribute positions, changes in allowed values, etc., as well as lexical assignment. For the most part an information-preserving conversion is possible, but for MSDs (attributes) of some lexical items only heuristics can be used for the conversion. Taking into account that all Amebis "unknowns" are by definition wrong, as all words are manually annotated with specific MSDs, and that a certain number of errors is introduced by the tagset mapping, Amebis obtains 87.9% accuracy on all tokens (incl. punctuation) in the dataset.

2.2 TnT MSDs

The TnT tagger was trained on the dataset itself, using 10-fold cross-tagging. The dataset was split into 10 parts, with 9 folds used for training, and the remaining fold tagged with the resulting model, and this process repeated for all 10 folds. As the lexical stock of jos100k is small, the tagging model used a backup lexicon which was extracted from the FidaPLUS corpus and its annotations. In other words, tri-gram statistics and lexicon containing uni-gram statistics of word-forms (their ambiguity classes) of frequent words were learned from jos100k, while less frequent words obtained their ambiguity classes from MSDs assigned by the Amebis tagger. Given such a tagging set-up, the obtained accuracy over the all dataset tokens (incl. punctuation) for TnT is 88.7%, slightly better than Amebis; but TnT has the advantage of learning how to correctly tag at least some unknown words (such as those marked as "foreign", i.e. tokens in spans of non-Slovene text), as well as having less problems with tagset conversion. Nevertheless, on the dataset it performs better than Amebis, so the TnT accuracy can be taken to constitute the baseline for the experiment.

2.3 Error comparison

Table 1 compares the errors made by the taggers against the gold standard. The first line gives the complete size of the corpus in words. The second gives the number of correct MSD assignment to word tokens for TnT (86.6% per-word accuracy), and the third for Amebis (85.7%). The fourth line covers cases where both taggers predict the correct MSD, for 78% of the words.

Lines 5 and 6 cover cases where one tagger correctly predicts the tag, while the other makes a mistake. These two lines cover a significant portion (2/3) of all the errors, so if such mistakes can be eliminated by deciding which tagger made the correct choice, the gains in accuracy are considerable.

The last two lines indicate upper bounds on the gains achieved by concentrating on choosing the correct tag. Line 7 gives cases where both taggers agree, but on an incorrect tag (3.2%), and line 8 the number of cases where both are wrong, but in different ways (2.4%); the upper bound on combination accuracy is thus 94.3%.

Let us look at two typical examples of cases 7 and 8. An example of both taggers being wrong, but agreeing on the assigned tag is exemplified in the fragment *"ni mogoče povedati" (it is not possible to tell)* where *"mogoče"* should be an adverb but both taggers assign it an adjectival tag. An example of both taggers being wrong in different ways is the fragment *"ni priporočene/Adj zgornje/Adj mejne/Adj vrednosti/Adj" (there is no recommended upper bound value)*. The correct tag for the noun is *Ncfsg*, i.e. feminine singular genitive, the genitive being determined by the (long distance) dependency on *"ni"*. The Amebis tagger correctly predicts this tag, while TnT makes a mistake, and assigns to the noun the plural accusative. As adjectives must agree with the noun in gender, number and case, the three adjectives preceding the noun must also be tagged as feminine singular genitive. Here both taggers are wrong: while TnT correctly posits the agreement between the noun and adjectives, all the adjective tags are wrong, due to the noun being incorrectly tagged. Amebis, on the other hand, does not pick up the agreement, and tags all three adjectives as masculine ones.

	Words	Gold	Amebis	TnT	Gloss
1	100,003	MSD1			Words in dataset
2	86,623	MSD1		MSD1	TnT tagger correct
3	85,718	MSD1	MSD1		Amebis tagger correct
4	78,018	MSD1	MSD1	MSD1	Both taggers correct
5	7,700	MSD1	MSD1	MSD2	Amebis correct, TnT error
6	8,605	MSD1	MSD2	MSD1	Amebis error, TnT correct
7	3,232	MSD1	MSD2	MSD2	Both wrong, and identical
8	2,448	MSD1	MSD2	MSD3	Both wrong, and different

Table 1: Comparison of tagging accuracy of Amebis and TnT over the 100k dataset.

3 Combining the taggers

As mentioned, our meta-tagger is built on top of two taggers, the Amebis rule-based tagger and TnT. The sole task of the meta-tagger is to decide which tag to consider correct. The meta-tagger is implemented as a classifier which, if the two underlying taggers disagree, classifies the case into one of the two classes indicating which of the two taggers is more likely to be correct. To train the classifier, we needed two things: a way to describe a case with a set of features, and a classification algorithm. The following section describes the feature construction process and the subsequent section the classification algorithms we tried out for this task.

3.1 Feature construction

To be able to train the classifier we needed to describe each case with a set of features. We decided to keep our meta-tagger relatively simple and to construct features solely out of tags predicted by the underlying taggers. Alternatively, we could compute content features as well (such as *n*-grams, prefixes, and suffixes) as it is the case with the SVM-based taggers such as SVMTool (Giménez & Márquez, 2004).

For training and testing we used the dataset discussed in Section 2, with each word assigned three tags: the correct tag (assigned manually), the tag assigned by TnT, and the tag assigned by the Amebis tagger. Each of these three tags can be decomposed into 15 attributes such as the part-of-speech category, type, gender, number, and so on. For a given tag, not all attribute

values are set, therefore the data is sparse in this sense (e.g. the value of gender and number for prepositions is "undefined").

The attributes of the tags assigned by the two taggers (but not those of the manually assigned tags) were directly used as features for training. In addition, we constructed features that indicate whether the two taggers agree on a particular attribute value or not (the so called agreement features). The example was labeled according to the tagger which correctly tagged the word (the label was thus either TnT or Amebis). Note that we built a training feature vector only when the two taggers disagreed and one of them was correct (if none of the taggers was correct, we were unable to label the feature vector). The entire feature construction process is illustrated in Figure 2.

For the first set of experiments we used the tag attributes and agreement features of the current word to construct a feature vector (termed non-contextualized features in Figure 2). In the second set of experiments, on the other hand, we also added tag features (from both, TnT and Amebis) from the previous and the next word (termed contextualized features in Figure 2). It is also important to mention that we ran a set of experiments where we excluded punctuation from the text and a set of experiments where each different type of punctuation was treated as a separate part-of-speech category (e.g. POS_T=,) with all the other attributes set to "not applicable". Each of these settings gave slightly different results. The results are discussed in Section 4 in more detail.

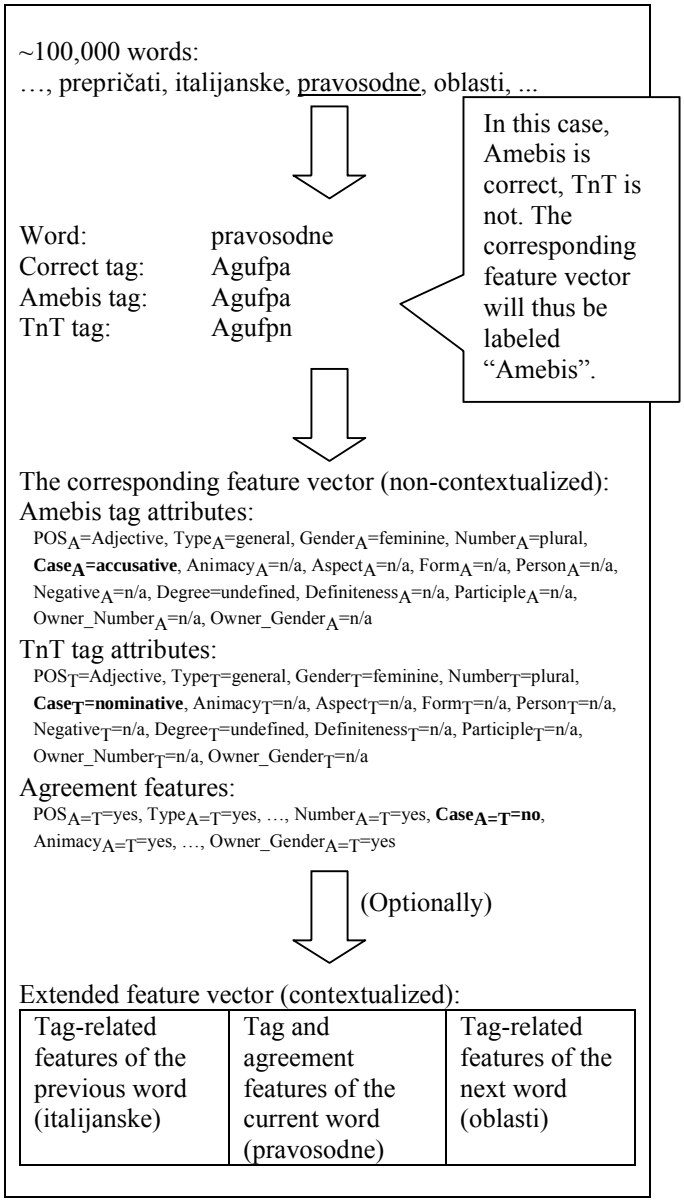


Figure 2: The feature construction process.

3.2 Learning algorithms

We experimented with three different classification algorithms: the Naive Bayes classifier, CN2 rule-induction algorithm, and C4.5 decision tree building algorithm. In this section, we briefly describe each of them.

The **Naive Bayes (NB)** classifier is a probabilistic classifier based on Bayes' theorem.[3] It naively assumes a strong independence of features. Furthermore, it is a black box classifier in the sense that its decisions are not easily explainable.

CN2 is an if-then rule-induction algorithm (Clark & Niblett, 1989). It is a covering algorithm meaning that each new rule covers a set of examples which are thus removed from the dataset. Unlike the Naive Bayes classifier, the trained model (i.e. a set of induced rules) provides an explanation for a decision (i.e. an if-then rule that was taken into account when classifying the example). Looking at the induced rules, it is also possible to read, understand, and also verify the knowledge that was discovered in the training set.

C4.5 is an algorithm for building decision trees; it is based on information entropy[4] (Quinlan, 1993). C4.5 uses the fact that each attribute of the data can be used to make a decision that splits the data into smaller subsets. It examines the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is the one used to make the decision. This process is repeated several times on smaller and smaller subsets of data. Similarly to CN2 (the rule-induction algorithm), C4.5 builds glass box models. Unlike its predecessor, the ID3 algorithm, C4.5 knows how to handle data with missing values (i.e. sparse data) and prunes the tree by cutting off branches that do not contribute to the classification accuracy.

[3] c.f. http://en.wikipedia.org/wiki/Naive_Bayes_classifier

[4] c.f. http://en.wikipedia.org/wiki/C4.5_algorithm

4 Experiments

In this section, we present tagging accuracies of the meta-tagger for different combinations of feature sets and underlying classification models. The size of the set of examples for training and testing is 16,305 and consists of 8,605 cases where TnT tagger predicted the correct tag and Amebis tagger did not and 7,700 cases where Amebis was correct and TnT was not. All experiments were conducted with the Orange data mining tool (Demšar et al., 2004). 5-fold cross validation method was used to evaluate the tagging accuracy of the meta-tagger in all experimental scenarios. We first discuss two baseline models for the meta-tagger, after that we define several different feature sets, then continue with the description of non-contextualized models and end the section with models that incorporate context features.

4.1 Baselines

The first baseline is the majority classifier which always predicts that TnT tagger is correct. This classifier achieves the accuracy of 52.8%.

The second baseline model is a Naive Bayes model trained on only one feature: Amebis MSD. This is a very simple model, since to classify a new example (with only one feature f, that is the Amebis MSD), all one needs to do is count the number of cases with MSD equal to f where Amebis was correct and the number of cases with MSD equal to f where Amebis was incorrect ($P(x = f, y = amebis\text{-}correct)$ and $P(x = f, y = amebis\text{-}incorrect)$) and predict the class (amebis-correct or amebis-incorrect) with the higher count. This model achieves the accuracy of 70.95% (approx. 18% higher than the first baseline).

Let us consider two examples. Assume that there were 200 cases where Amebis predicted the tag Pd-nsg, and it was correct in 150 of these cases (this means the TnT was correct in the remaining 50 cases). This means that P(Amebis-predicts: Pd-nsg, Amebis-correct) = 0.75. In this case the meta-tagger would always predict the tag Pd-nsg if Amebis predicted it as well.

Now, if we assumed that Amebis was correct in 80 of 200 cases, P(Amebis-predicts: Pd-nsg, Amebis-correct) = 0.4), then the meta-tagger would always predict the tag predicted by TnT, given that Amebis predicted Pd-nsg (the evidence in the training data tells us not to trust the Amebis tagger, since the probability of it being correct is less than 0.5).

4.2 Feature sets

We will now describe the features for the non-contextualized models. The first set of features for the non-contextualized models are the so called FULL features; they only include full Amebis MSD and full TnT MSD (two features). The second set of features called DEC is a decomposition of the FULL features as described in Section 3.1 (45 features: 15 Amebis features, 15 TnT features, 15 Agreement features). The third set of features, BASIC, is a subset of DEC features, where we only take the features corresponding to Category, Type, Gender, Number and Case into account (10 features: 5 for Amebis and 5 for TnT). The final set of features, ALL, is a union of FULL and DEC (47 features).

Feature sets for contextualized models (with and without punctuation) are extensions of non-contextualized feature sets, where the features of examples surrounding our training example are added (see Section 3.1). The context features (i.e. the features of the previous and next word) are the same ones as that of the current word except for the Agreement features which are only computed for the current word (in the DEC feature set we thus keep only 15 Agreement features: the ones of the current word).

Features ALL, when contextualized, include six features for MDS tags (Amebis-Prev, Amebis, Amebis-Next, TnT-Prev, TnT, TnT-Next), 45 for Amebis tag features (3 × 15 features), 45 for TnT tag features and 15 Agreement features, which sums up to 111 features.

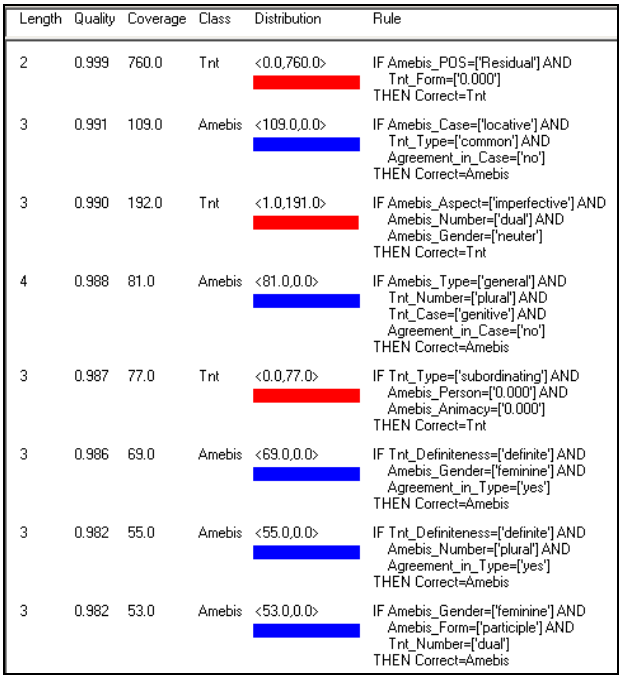
4.3 Non-contextualized models

Experiments with features that do not take context into account (Table 2) show that C4.5 is the most robust classifier with respect to different feature sets and that it can achieve the highest accuracy. We can also observe that tag features are not very suitable for the Naive Bayes classifier because the conditional independence assumptions are too strongly violated.

Feature set / Classifier	FULL	DEC	BASIC	ALL
NB	73.90	67.55	67.50	69.65
C4.5	73.51	**74.70**	74.23	73.59
CN2	60.61	72.57	71.68	70.90

Table 2: Non-contextualized models (accuracy in %). Feature sets FULL, DEC, BASIC and ALL are explained in Section 4.2.

Even though the CN2 algorithm results in slightly lower accuracy it can prove useful since the rules that it produces are easy to interpret and thus discover the strengths and weaknesses of the TnT and Amebis classifiers (see Figure 3).



Length	Quality	Coverage	Class	Distribution	Rule
2	0.999	760.0	Tnt	<0.0,760.0>	IF Amebis_POS=['Residual'] AND Tnt_Form=['0.000'] THEN Correct=Tnt
3	0.991	109.0	Amebis	<109.0,0.0>	IF Amebis_Case=['locative'] AND Tnt_Type=['common'] AND Agreement_in_Case=['no'] THEN Correct=Amebis
3	0.990	192.0	Tnt	<1.0,191.0>	IF Amebis_Aspect=['imperfective'] AND Amebis_Number=['dual'] AND Amebis_Gender=['neuter'] THEN Correct=Tnt
4	0.988	81.0	Amebis	<81.0,0.0>	IF Amebis_Type=['general'] AND Tnt_Number=['plural'] AND Tnt_Case=['genitive'] AND Agreement_in_Case=['no'] THEN Correct=Amebis
3	0.987	77.0	Tnt	<0.0,77.0>	IF Tnt_Type=['subordinating'] AND Amebis_Person=['0.000'] AND Amebis_Animacy=['0.000'] THEN Correct=Tnt
3	0.986	69.0	Amebis	<69.0,0.0>	IF Tnt_Definiteness=['definite'] AND Amebis_Gender=['feminine'] AND Agreement_in_Type=['yes'] THEN Correct=Amebis
3	0.982	55.0	Amebis	<55.0,0.0>	IF Tnt_Definiteness=['definite'] AND Amebis_Number=['plural'] AND Agreement_in_Type=['yes'] THEN Correct=Amebis
3	0.982	53.0	Amebis	<53.0,0.0>	IF Amebis_Gender=['feminine'] AND Amebis_Form=['participle'] AND Tnt_Number=['dual'] THEN Correct=Amebis

Figure 3: List of rules discovered by CN2 in Orange. Rules are ordered by their quality which is a function of rule coverage and rule accuracy. The second rule, for example, tells us that if Amebis predicted locative case and TnT predicted some other case and TnT predicted common type, then the meta-tagger should predict the same tag as Amebis. The first rule, IF Amebis_POS=[‘Residual’] AND TnT_Form=[‘0.000’] THEN Correct = TnT, covers the examples mentioned in Section 2.1, where Amebis predicts POS tag “unknown” (by definition incorrect). The rule says that in such case, TnT is always correct, which is what is expected.

4.4 Context and punctuation

When comparing the results of experiments with context, we notice that taking punctuation into account (see Section 3.1) is beneficial in almost all cases (see Tables 3 and 4). This can be explained by the fact that ignoring punctuation can yield unintuitive context tags, for instance the sequence of tags T1, T2, T3, where T1 is the last word of a sentence, T2 the first word and T3 the second word of the next sentence.

We notice that C4.5 can best benefit from extra contextual features, whereas the performance of the other algorithms does not change notably.

Feature set / Classifier	FULL	DEC	BASIC	ALL
NB	73.10	68.29	67.96	70.55
C4.5	73.10	78.51	**79.23**	76.72
CN2	62.16	73.26	72.75	72.29

Table 3: Context without punctuation (accuracy in %).

Feature set / Classifier	FULL	DEC	BASIC	ALL
NB	73.44	68.32	68.14	70.53
C4.5	74.18	78.91	**79.73**	77.68
CN2	62.23	74.27	72.82	73.01

Table 4: Context with punctuation (accuracy in %).

5 Large-scale experiment

In addition to the experiments on the jos100k corpus, we also performed a large-scale experiment on a larger subset of FidaPLUS, the jos1M corpus, consisting of 1,000,017 word tokens (without punctuation). The corpus was first tagged by both taggers (i.e. Amebis and TnT). Amebis is a rule-based tagger and does not require training, TnT, on the other hand, was trained on the complete jos100k corpus. Then, if (and only if) the two taggers disagreed on a particular word token, the token was manually validated. Consequently, we are unable to determine cases when both taggers are correct or agree on an incorrect tag. Dataset statistics (analogous to the ones in Table 1) are given in Table 5.

	Words	Gold	Amebis	TnT	Gloss
1	1,000,017				Words in dataset
2	809,897		MSD1	MSD1	Both taggers agree
3	75,378	MSD1	MSD1	MSD2	Amebis correct, TnT error
4	88,657	MSD1	MSD2	MSD1	Amebis error, TnT correct
5	26,085	MSD1	MSD2	MSD3	Both wrong, and different

Table 5: The jos1M corpus statistics.

5.1 Experimental setting

We confronted Naive Bayes with C4.5 (building CN2 rules was computationally too expensive). We experimented with all defined feature sets: FULL, DEC, BASIC, and ALL, with and without context. Punctuation was included in the contextualized cases. For some reason, the C4.5 algorithm was unable to handle feature sets FULL and ALL when contextualized. We speculate that the implementation in Orange does not manage memory efficiently when it comes to attributes with 1000+ different values. The results of the experiments are presented in the following section.

5.2 Results

In this section, we present tables analogous to the ones in Section 4. We show how the algorithms perform under different feature sets. As already said, we do not show results for the CN2 algorithm and for C4.5 under certain conditions (denoted with “N/A”). The results fully support our observations on the smaller jos100k corpus and are presented in Tables 6 and 7. Note also that the

second baseline yields 72.39% accuracy on the jos1M corpus.

Feature set / Classifier	FULL	DEC	BASIC	ALL
NB	73.93	66.85	66.67	69.81
C4.5	76.45	**76.56**	76.29	76.49

Table 6: The jos1M corpus – non-contextualized models (accuracy in %).

Feature set / Classifier	FULL	DEC	BASIC	ALL
NB	73.74	67.59	67.86	70.28
C4.5	N/A	**84.18**	84.01	N/A

Table 7: The jos1M corpus – context and punctuation (accuracy in %).

6 Conclusions

The paper presents a meta-tagger built on top of two taggers, namely the TnT HMM-based tagger and the Amebis rule-based tagger. The purpose of the meta-tagger is to decide which tag to take into account if the two taggers disagree in a particular case.

The experimental results show that the two taggers are quite orthogonal since very little information is needed to get a significant increase in performance from the first baseline.

Furthermore, using context can improve the performance of some models and taking punctuation into account when constructing context features is better than ignoring it. C4.5 with context and punctuation features achieves the highest accuracy, 79.73% on jos100k and 84.18% on jos1M, which results in a meta-tagger with significantly higher accuracy than Amebis tagger or TnT tagger. The overall accuracies are given in Figure 4. Note that the first baseline is equal to the TnT overall accuracy.

There are roughly 5% cases in which both taggers assign an incorrect tag. By using the technique discussed in this paper (i.e. rule inference), it would be possible to learn under which conditions the two taggers are both mistaken and thus alert the user about such tags.

Furthermore, it would be possible to apply our technique on a per-attribute basis. We would be able to predict incomplete tags, i.e. tags with some attributes missing, where the missing attributes would be those most likely predicted falsely by both taggers. This would be very useful as guidance for human taggers preparing the JOS corpus. The missing attributes would have to be entered manually; the rest would only need to be validated.

Tagging on a per-attribute basis and looking at cases in which both taggers predict an incorrect tag will be the focus of our future research. In addition, we will consider including more taggers into the system. The main idea is to develop taggers, specialized to handle cases in which the two currently used taggers are not successful.

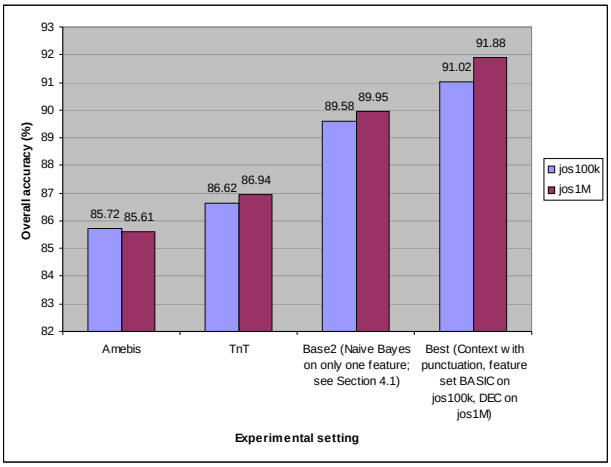


Figure 4: The overall accuracies (%). We can see that our meta-tagger exhibits around 4%–5% overall improvement over the two underlying taggers (i.e. TnT and Amebis). For computing the accuracies on the jos1M corpus, we needed to estimate the number of cases where the two taggers agreed on a correct tag. Looking at the statistics of the jos100k corpus (Table 1), we can see that the taggers are correct in 96.4% of the cases where they agree on the tag. Therefore, we computed the required number as 96.4% of 809,897 which is 780,740.71.

Acknowledgements

The work described in this paper was supported in part by grant ARRS J2-9180 "Jezikoslovno označevanje slovenskega jezika: metode in viri" and EU 6FP-033917 SMART "Statistical Multilingual Analysis for Retrieval and Translation".

References

[1] Arhar, Š. and Gorjanc, V. (2007). Korpus FidaPLUS: nova generacija slovenskega referenčnega korpusa. Jezik in slovstvo, 52(2): 95–110.

[2] Brants T. (2000). TnT – A Statistical Part-of-Speech Tagger. In Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000, 224–231.

[3] Clark, P. and Niblett, T. (1989). The CN2 Induction Algorithm. Machine Learning, 3(4): 261–283.

[4] Demšar J., Zupan B. and Leban G. (2004). Orange: From Experimental Machine Learning to Interactive Data Mining. White Paper (www.ailab.si/orange), Faculty of Computer and Information Science, University of Ljubljana.

[5] Erjavec, T., Džeroski, S. and Zavrel, J. (2000). Morphosyntactic Tagging of Slovene: Evaluating PoS Taggers and Tagsets. In Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'2000). ELRA, Paris.

[6] Erjavec, T. (2004). MULTEXT-East Version 3: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. In Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, 1535–1538.

[7] Erjavec, T. and Krek, S. (2008). The JOS morphosyntactically tagged corpus of Slovene. In Proceedings of the Sixth International Conference on Language Resources and Evaluation, LREC 2008.

[8] Giménez, J. and Márquez, L. (2004). SVMTool: A General POS Tagger Generator Based on Support Vector Machines. In Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04).

[9] Hajič, J. and Hladka, B. (1998). Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset. COLING-ACL'98. ACL.

[10] Quinlan, J.R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, Inc.

[11] Sjöbergh, J. (2003). Combining POS-taggers for improved accuracy on Swedish text. In NoDaLiDa 2003, 14th Nordic Conference on Computational Linguistics. Reykjavik.

[12] Spoustová, D., Hajič, J., Votrubec, J., Krbec, P. and Květoň, P. (2007). The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for Czech. Proceedings of the Workshop on Balto-Slavonic Natural Language Processing. June 2007. Prague, Czech Republic. Association for Computational Linguistics.

[13] Zribi, C.B.O., Torjmen, A. and Ahmed, M.B. (2006). An Efficient Multi-agent System Combining POS-Taggers for Arabic Texts. In Computational Linguistics and Intelligent Text Processing. LNCS Volume 3878/2006, Springer.