Efficient Stochastic Part-of-Speech Tagging for Hungarian

Csaba Oravecz*, **Péter Dienes**†

*Research Institute for Linguistics
Hungarian Academy of Sciences, Budapest
oravecz@nytud.hu

†Saarland University
Saarbrücken
dienes@coli.uni-sb.de

Abstract

Many of the methods developed for Western European languages and used widespread to produce annotated language resources cannot readily be applied to Central and Eastern European languages, due to the large number of novel phenomena exhibited in the syntax and morphology of these languages, which these methods have to handle but have not been designed to cope with. The process of morphological tagging when applied to Hungarian data to produce corpora annotated at least at the morphosyntactic level is most indicative of this problem: several of the algorithms (either rule-based or statistical) that have been used very successfully in other domains cannot readily be applied to a language exhibiting such a varied morphology and huge number of wordforms as Hungarian. The paper will describe a robust tagging scenario for Hungarian using a relatively simple stochastic system augmented with external morphological processing, which can overcome the two most conspcicuous problems: the complexity of morphosyntactic descriptions and most importantly the huge number of possible wordforms.

1. Introduction

Part of speech tagging is considered a cheap and well-studied method to produce language resources with some low level linguistic annotation usable for various purposes such as serving as a preparatory phase for more complex language processing tasks. For Western European languages tagging has been extensively described and various methods have been developed (Merialdo, 1994; Brill, 1995; Ratnaparkhi, 1997; Daelemans et al., 1996) and even combined (van Halteren et al., 1998; Brill and Wu, 1998) producing results so satisfactory that might seem to suggest that research at this level of natural language processing is no longer that interesting or rewarding. However, when trying to deal with highly inflectional or agglutinative languages many of the otherwise successful methods face extra difficulties, the number of different possible morphosyntactic descriptions (MSD) and wordforms being the most apparent.

There has already been considerable research addressing these issues in highly inflectional languages (Hajič and Hladka, 1998; Erjavec et al., 1999; Tufiş, 1999), experimenting with some procedure to overcome the problem of the large number of morphosyntactic descriptions by generating or designing a reduced tagset and use it for the proper tagging of texts. However, this process is faced with a difficulty of identifying those features which actually carry information relevant for the disambiguation task. The other phenomenon (which is particularly characteristic of Hungarian), i.e. the large number of wordforms, also presents an unwelcome situation for most statistical models widely used in state-of-the-art taggers: lexical probabilities are calculated from a wordform lexicon generated during training, thus extremely large training corpus would be necessary to derive reliable statistics for the possible wordforms in unseen data and even if the training corpus is fairly large (several 100k words), this process will inevitably result in a large number of words not seen in the training. A simple solution is mentioned in passing in Tufiş et al. (2000), and also proposed by Hajič (2000), which basically amounts to making the list of each possible tags for any wordform available for the tagger. For Hungarian, this proves to be the only workable alternative and so will be adopted here, necessitating an external morphological analyzer as a core component of the tagging system.[1]

The paper will empirically investigate how the information supplied by the morphological analyzer can be utilized to improve tagging performance, our goal being here not so much to build a PoS tagger for Hungarian with high accuracy, but rather to define an architecture suitable for the disambiguation task which, based on a simple stochastic tool, can cope with the particular characteristics of highly inflectional languages. In section 2. we will discuss the main problem that originates from these characteristics as it is reflected in Hungarian data. Section 3. will give a brief description of the data and tools used in the experiments. In section 4. we will show how stochastic tagging can be aided by information from a morphological analyzer. Section 5. will give a short summary on the error analysis. Conclusions and suggestions for further work will follow in section 6.

2. Data sparseness in highly inflective languages

It is a computational linguistic common sense that (morphological) tagging of highly inflective and agglutinative languages is a more difficult task than, for example, PoS

[1] Hajič (2000) proposes the preparation of an independent morphological dictionary by some automatic method. For Hungarian, the only feasible solution is to use a morphological analyzer that provides the necessary information on-line.

tagging of morphologically less rich languages, such as English, French, German, etc. (cf. Hajič (2000)). The reason for this situation is generally claimed to be *data sparseness*. Data sparseness, however, manifests itself in several, clearly different ways when the task of morphological tagging is addressed.

One of the most frequently discussed issues that might lead to the sparsity of training data is the high cardinality of the tagset (Brants, 1997; Tufiş, 1999; Tufiş et al., 2000). While the number of tags for non-agglutinative languages is generally between 50 and 200, systems for agglutinative and highly inflective languages use tagsets of cardinality with a magnitude higher (cf. eg. (Harris et al., 2000)). This entails that – in case of an $n$gram statistical model – we have to estimate 10^n times more parameters, which would need bigger training corpora for these languages. Contrary to the needs, however, the amount of available annotated linguistic resources for these languages is much smaller than for well-researched languages, such as English. One way out of the problem caused by the huge tagsets is using an internal CTAG set of smaller cardinality for tagging purposes. Dienes and Oravecz (2000) propose an automatic tagset reduction algorithm with full recoverability (i.e. for each word in an ambiguity class the original tags are mapped to distinct CTAGS) and discuss the effect of the size of the internal tagset on morphological tagging of Hungarian. The results show that significant reduction can be achived without a decrease in the accuracy of the tagger.

Another often cited problem with highly inflective languages is the amount of ambiguity present in the morphological system, which relates to the tagset size as well. However, there is some uncertainty about which token should in fact be taken as ambiguous, and what kind of distinctions should be made in the morphological specification of a wordform. Often the decision that a specific token is assigned to more than one morphological class depends on several factors ranging from linguistic tradition to the projected use in further processing.

Two tentative guidelines seem to be considered in the domain of tagging when ascribing different analyses to the same wordform:

(i) Full neutralization is dispreferred. That is, a wordform should only have more than one analysis if there is an unambiguous wordform for each of the analyses that can be substituted for the original wordform in the specific environment.[2]

(ii) What distinctions are present in features is performance oriented. That is, one should underspecify those features that cannot be resolved reliably, and try to handle them by post-processing with varying methods; and should preserve those or even introduce new distinctions that can be well resolved by the current model. In any way, the primary objective is to try to reduce the error rate to that of the oracle available.

Unfortunately, no standard methodology exists that would provide an independent metric of the difficulty of the tagging task across different settings, so even the standardly reported measures of tagset size, ratio of ambiguous tokens and average ambiguity can be of little indication how hard the task for the classifier actually is, and there are so many parameters that can be set differently[3] that the usual error rate/performance measures might well become entirely uninformative.

Nonetheless, following standard practice two measures of ambiguity will be considered here: the percentage of ambiguous tokens and the average number of tags assigned to tokens. According to the first measure, English seems to be as difficult as any highly inflective language, since the amount of ambiguous tokens is about the same (English: 38.65%, Romanian: 40%, Czech: 45.97% (Hajič, 2000)). In this respect, Hungarian should be easier to tag, since only about 23% of the tokens are ambiguous, though this obviously depends on the definition of ambiguity and on how refined analysis we would like to have. Another measure of ambiguity is the average number of tags per tokens. According to this measure, the figures show more variation: Czech: 3.65 or 2.36 tags per token, depending on the tagset and corpus (Hajič, 2000), English: 2.34 or 1.72, Romanian: 1.59-67 (Tufiş, 2000), Hungarian: 1.33. To what extent these differences have an effect on tagging is far from clear and is difficult to test, since there are many other factors that prevent us from controlling this parameter only. According to these measures, however, all other things being equal, tagging Hungarian should be easier than tagging English. This is not the case, which means that there are other factors that come into play.

In fact, the most relevant manifestation of data sparseness is not the one we mentioned above. The principal reason for the rather poor performance of otherwise fairly accurate taggers (eg. for English) can be inferred from Table 1 (HU and EN stand for Hungarian and English, respectively; with AC we refer to the modified Hungarian corpus where every word is mapped to its ambiguity class – see Section 4.2.). Generally, in the case of Hungarian and English, there are twice as many different word types in a corpus of the same size (for a brief description of the data see Section 3.). This entails that, for lexicalized taggers, Hungarian is much more difficult, since the data is sparser to

[2]According to this criterion one would not assign for instance the two analyses *genitive* and *dative* to Hungarian nominal forms with *-nak/-nek* suffix in an otherwise genitive construction because there is no unambiguous form that could only be analyzed as genitive in that environment. On these grounds, one can even deny the presence of genitive case in Hungarian, which many in fact do.

[3]It is not unusual that in the tagging of highly inflective languages morphological syncretism is not reflected in underspecified values in cases when the resulting disambiguation task is not easier than resolving a distinction not inherently present in the morphology of some other language would be (e.g. different *-ing* forms of English main verbs are not normally distinguished although they have quite distinct distributions). While in certain languages some disambiguation problems are just naturally regarded as belonging to the domain of tagging, in other languages it is delegated to other levels of analysis, and this can be due only to the presence of morphological marking of some phenomena, not necessarily to the fact that the disambiguation task itself would be any harder in one language than in the other.

		HU 571 tags	EN 50 tags	AC 571 tags
Training corpus	tokens	243829	245714	243829
Training corpus	types	49649	19021	1914
Test corpus	tokens	27001	27319	27001
Test corpus	types	9994	5701	892
Types in test not seen while training		36.1%	17.5%	7.2%
Tokens in test not seen while training		17.3%	4.5%	0.29%
Tokens in test with unseen tags while training		17.9%	5.6%	0.5%
% of diff. tags in test not seen while training		8.0%	1.6%	8.0%

Table 1: Comparison of English and Hungarian corpora

make accurate estimation for the probability distribution of tags given a word.

Figure 1 (lines EN and HU) shows a different aspect of the same phenomenon: in Hungarian low frequency words (words with at most 5 occurrences) amount to 33% of the word tokens, whereas in English low frequency words constitute only 10% of the total corpus (without numbers). The

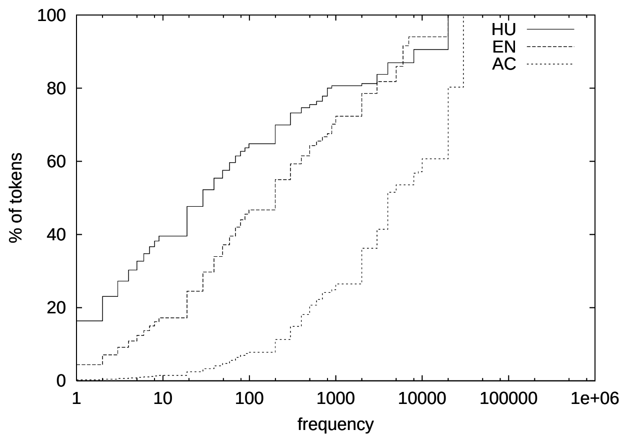

Figure 1: Cumulative percentage of tokens wrt. type frequency

difference is even more striking in the case of singletons: in Hungarian, they add up to 16% of the corpus, while in English they contribute only 4% of the words to the corpus. A straightforward consequence of the difference in the behavior of the two languages is the different amount of tokens of the test data previously not seen during the training phase. As Table 1 shows, for Hungarian, 17.3% of the word tokens in the training data is not seen while training, which is almost four times as much as the corresponding amount for English (4.5%). This means that, without a suitable guessing method, lexicalized taggers on Hungarian are bound to achieve much worse results than on English. Or, to put it the other way round, in order to be able to achieve acceptable tagging accuracy for Hungarian (and other agglutinative languages), one has to apply a suitable guesser to handle previously unseen words. In these experiments we used an external morphological analyzer, which is able to return the possible morphological analyses for a wordform.

3. Data and tools

The data for calculating the results in Figure 1 and Table 1 and for the tagging experiments discussed in Section 4. consisted of the Hungarian training corpus which contained 270,830 tokens. We used a similar sized part of the first section of the Wall Street Journal with the Penn Treebank PoS tagset as an English reference corpus. The results were obtained by standard 10-fold cross validation. In the tagging experiments the default tagset used preserved basically all morphological information with the cardinality of more than 2000, of which only 571 actually occurred in the training corpus.

The external morphological analyzer is a wide coverage,[4] rule based morphological analyzer developed originally for Hungarian (Prószéky and Tihanyi, 1996), which returns a list of possible morphological analyses for the given word.

The tagging experiments needed a slightly modified[5] version of Thorsten Brants' TnT tagger (Brants, 2000), the underlying architecture of which is a trigram Hidden Markov Model. The present model, however, did not learn lexical probabilities from the training corpus, but could be fed with *inverted lexical probability* distribution, that is, in the input each token was associated with a list of possible tags augmented with the conditional probabilities of tags given the token ($P(t_i|w)$). The tagger, then, calculated the lexical probabilities $P(w_j|t_i)$ required by the Markov Model via Bayesian inversion.

4. Class-based guessing

4.1. Using a morphological analyzer

As mentioned above, in the case of languages with rich inflection, an accurate tagger must have a proper guesser which is able to provide all possible tags for words unseen in the training corpus. In these experiments we opt for a symbolic guesser, that is, for a morphological analyzer. When using a symbolic module in a stochastic NLP architecture, one has to answer three important questions:

[4]Considering that our aim in the tagging experiments was to test the tagging architecture instead of the guesser, there were no unknown words for the morphological analyzer in the test and training corpora.

[5]We would like to thank Thorsten Brants for making these modifications for our purposes.

(i) why using a symbolic module instead of an alternative stochastic one

(ii) how to incorporate the symbolic module into a stochastic framework (eg. how to induce probabilities from the output of the symbolic module)

(iii) what is the domain of application of the symbolic module

Our choice of using an external morphological analyzer is due to the fact that the suffix guessing algorithm of Samuelsson (1996), which is built into TnT, and which seems to be suitable for English and German (Brants, 2000), does not work well for Hungarian (cf. Table 2, models 2 and 3). In our opinion, there are two main reasons for the unsuitability of the suffix guessing algorithm for agglutinative languages: (i) data sparseness and (ii) high level of suffix ambiguity. On the one hand, agglutinative languages carry a considerable amount of morphological information on the suffixes, and the suffix sequence can be usually fairly long (up to 8 characters in our training corpus). This means that, in the worst case, there could be 33^8 morphologically distinct endings. This is, of course, an overestimation, but clearly shows that long suffixes might carry important morphological information, but they might not occur in the training corpus in sufficient number. On the other hand, the most frequent suffix forms in Hungarian are homographs, ie. they tend to participate in several inflectional paradigms (eg. the ending *-ek* can be attached to a verb or a noun). Interestingly, words with these endings do not tend to be ambiguous, since the stem generally takes either reading – a suffix guessing algorithm cannot disambiguate these cases whereas a morphological analyzer can. The problem is further complicated by the fact that the zero morph is the allomorph of several frequent morphemes, though, as before, the stem generally disambiguates between the morphemes. Finally, morphemes in Hungarian usually have several surface forms, which also contribute to the sparse data problem. These findings motivated us to use a morphological analyzer instead of relying on the suffix guessing algorithm.

Table 2 summarizes the results of experiments with a few basic models. In the baseline model, tokens seen in the training phase were assigned tags on the basis of unigram frequencies calculated from the training corpus while previously unseen tokens with a very rudimentary strategy received the nominal tag that was found most frequent in training.[6] In the second model the baseline strategy for seen words was augmented with the suffix guessing algorithm which gave some rise in overall accuracy, whereas the third model was based on (the trigram) TnT with the same suffix guessing algorithm again. Given that it was now a trigram model it is predictable that accuracy on previously seen tokens should be higher. However, performance on unseen tokens did not increase to a level comparable to other languages.[7]

When the symbolic morphological analyzer is plugged into the system, a significant increase in the unseen (and consequently in the overall) accuracy is clearly noticeable (Model 4 in Table 2). This is hardly surprising provided that a fair number of tokens are already assigned only one analysis by the analyzer. More striking, however, is the very high accuracy on previously seen tokens obtained by the primitive unigram models in Table 2, which suggests that in Hungarian the frequent ambiguous items tend to occur in running text with their most frequent analysis.

4.2. Weighting the output of the analyzer

Since the morphological analyzer does not assign probabilities to the tags, we have to address the question how to incorporate such knowledge into our tagging system. As described in Section 3., the underlying model of the architecture is a Hidden Markov Model, so we have to augment the information about the tokens with inverted lexical probabilities. The most naive way to get around this problem is simply assuming uniform distribution of the tags proposed by the guesser given the word. This approach already gives a considerable rise in accuracy compared to the suffix guessing method discussed above (cf. Model 4 in Table 2 and Model 7 in Table 3). In this paper, however, we propose another approach which yield higher accuracy with the present architecture.

How could we infer the inverted lexical probabilities of tags given a previously unseen word? Even if we have not seen the word in the training corpus, we might have seen other words that behave similarly in the sense that they occur in similar environments, where they take the same tag. Words, for example, that are in the same *ambiguity class*, that is, words that show the same ambiguity behavior (eg. *books* and *loves*, both of which are ambiguous between NNS and VBZ) might be very good candidates for inferring information about the word in question. Indeed, (infrequent) words with the same morphology tend to have the same preferences. Hence, to determine the distribution of the tags for a previously unseen word belonging to a given ambiguity class, it is a motivated step to use the (inverted) lexical probabilities for the words in the same ambiguity class. A computational motivation for taking ambiguity classes into account is that such a step highly reduces the sparse data problem. As Figure 1 and Table 1 show, with the introduction of ambiguity classes (AC) instead of lexical words, Hungarian becomes a very well-behaved language: low-frequency tokens amount only to the less than 1% of the training corpus and only 0.29% of the tokens in the test are unseen while training.

To our knowledge the idea of using ambiguity classes is first proposed by Kupiec (1989) and later is pursued by, for example, Tzoukermann and Radev (1996). Their methodology is simple and elegant: in both the training and test corpora, words are substituted with their ambiguity class (thus the training and test corpora contained tokens like NNS_VBZ instead of *books* or *loves*), and train and test their models with the modified corpora. The problem with such a

[6] We did not exclude specifically punctuation from evaluation since once a morphological analyzer is utilized, they become just the same unambiguous tokens as any other unambiguous token.

[7] Brants (2000) reports 85.5% for English and 89% for German with TnT.

Model		Performance			
		% of unseen	Seen acc.	Unseen acc.	Overall acc.
1.	Baseline	17.31%	97.97%	17.54%	84.06%
2.	Baseline + Suffix Guess	17.31%	98.00%	64.51%	92.18%
3.	TnT	17.31%	98.32%	67.07%	92.88%
4.	Baseline + Morph. Analyzer	17.31%	97.97%	87.67%	96.19%

Table 2: The performance of basic models

Model		Performance			
		% of unseen	Seen acc.	Unseen acc.	Overall acc.
5.	TnT + AC	2.16%	98.27%	61.89%	97.48%
6.	TnT + Full AC	0.29%	97.83%	6.57%	97.57%
7.	TnT + MA Uniform	17.31%	98.25%	95.50%	97.77%
8.	TnT + MA Classbased	17.31%	98.25%	97.32%	98.08%
9.	TnT + MA Smoothed	17.31%	98.28%	97.32%	98.11%

Table 3: The performance of TnT aided by the morphological analyzer (MA)

move, however, is that we lose preferences of the frequent tokens. For example, even though *books* and *looks* share the same ambiguity class, in the WSJ the former tends to prefer the nominal tag (there is no verbal occurrence at all) while the latter takes the verbal tag in most of the cases. (The results without using any lexical information are described in Table 3, model 6.) To remedy the situation, Kupiec (1992) maps only infrequent words to their ambiguity classes, and retain frequent tokens as they are. In our experiment, we tried a similar scenario: to determine which token is mapped to its ambiguity class, first, an independent token frequency list was prepared from an 80 million word corpus, which was analyzed by a morphological analyzer, allowing for the construction of another frequency list, that of the ambiguity classes. Now, in the conversion from tokens to equivalence classes, a token was represented by its actual wordform if the token occurred at least 500 times in the 80 million corpus, otherwise it was substituted with its ambiguity class. Clearly, this approach leaves much room for fine tuning, but we do not expect the different parameter settings to influence performance significantly. We trained and tested the tagger on the new corpora, and the result is described in Table 3 model 5. It is interesting to see that the slightly lexicalized model is (insignificantly) worse than the non-lexicalized one.

Although the equivalence class approach improves the tagging accuracy considerably, it has some important shortcomings. The most important one is the fact that we lose information which is otherwise present in the training corpus. Indeed, if a word is not mapped to its ambiguity class, it does not contribute to the tagger's knowledge about the tag distribution of its ambiguity class. On the other hand, if the token is not frequent enough and is mapped to its ambiguity class, we lose its lexical properties. In the light of the high baseline accuracy, which suggests that words generally occur with their most frequent tags, this approach is slightly penalized. Naturally, we would not like to lose any information when considering ambiguity classes, thus we propose a method that makes use of the distribution of the tags given both the lexical token and its ambiguity class. Moreover, the approach of using ambiguity classes needs further modification in order to be able to handle previously unseen ambiguity classes (in the previous experiments, we used the built-in suffix guesser).

In order to avoid loss of information, we train our model with both lexical items and ambiguity classes. In the present architecture, this means the generation of two lexicons: one of them containing frequency counts of tags for each word, from which we calculate the inverted lexical probabilities $\hat{P}(t_j|w_i)$. The other lexicon is quite similar, containing the frequency counts of tags for each ambiguity class. The input to the tagger is words with their ambiguity class and optionally with a list of possible tags. When the tagger encounters a previously unseen word, it uses the tag distribution of its ambiguity class. On the other hand, if the word is seen in the training corpus, we take the distribution of the tags for the lexical token. However, it is possible that a word did not occur in the training corpus with an otherwise possible tag predicted by the ambiguity class. In these cases we give some probability mass to the previously unseen tag. To be more exact, we add some (fixed) frequency counts to the tag, thus a previously unseen tag with low frequency tokens will get a higher conditional probability than with high frequency tokens. The algorithm is more formally described in Figure 2. The results of the experiment using this kind of weighting method, described in

Table 3 (model 8), show a considerable increase over the approaches using equivalence classes or assuming uniform distribution of tags.

```
if unseen(word)
  return normalize(dist(class))
else
  tags=union of tags of dist(word) and dist(class)
  num=number of tags in tags
  foreach tag in tags
     if tag is in distrib(word)
        add tag with freq from dist(word) to newdist
     else
        add tag with freq=1/num to newdist
  return normalize(newdist)
```

Figure 2: The calculation of the tag distributions given the word and its ambiguity class

Note that there is an important and welcome side-effect if a tagging architecture is of the type described above.[8] Apart from its ambiguity class, in the input file, each word can also be associated with a list of possible tags – the system then restricts the calculation of the inverted lexical distribution to the tags given. This enables us to use external (possibly symbolic) modules in order to filter out some readings based on wider contextual, semantic, etc. information, thus making the tagger even more suitable for fast and accurate shallow text processing.

4.3. Smoothing

The third question concerning the incorporation of the morphological analyzer into our stochastic NLP system addresses the domain of application of the symbolic module. At first sight, we only make use of the morphological analyzer when encountering a previously unseen word. This is partly true, however, since the algorithm described in Figure 2 does some simple smoothing for words that occur in the training corpus as well: possible but unseen tags get some probability mass depending on the number of occurrences of the word in the training. This method, however, does not really combine the tag distributions given the word and the ambiguity class – the most important factor is the word if it is seen. A more sophisticated approach would try to combine the two distributions via, for example, frequency dependent linear combination, i.e. for each tag, the inverted lexical probability given the word and ambiguity class would be

$$\hat{P}(t|w,c) = \lambda_{w,c}\hat{P}(t|w) + (1-\lambda_{w,c})\hat{P}(t|c)$$

at least for infrequent tokens. If we adopt this model, we have to find out how to determine $\lambda_{w,c}$. If the word is not very rare (in our experiments, words occurring more than 5 times in the training corpus), we used no smoothing (or more explicitly, we used the smoothing technique of the algorithm of Figure 2). But what to do with infrequent words? There are several options, the most obvious is to fix $\lambda_{w,c}$ independently of the word and the ambiguity class. Alternatively, one could set the parameter according to the relative frequency of the word and its ambiguity class. The problem with this approach is that the ambiguity class generally occurs much more often than the lexical item, hence the ambiguity class would get too heavy a weight. Thus, we have chosen $\lambda_{w,c}$ to be dependent on the number of word types belonging to the ambiguity class and the frequencies of the word and the ambiguity class. This kind of smoothing gives an insignificant rise in the accuracy (cf. Table 3 model 9).

5. Error analysis

The following is a short overview of the main findings from the analysis of tagging errors; a detailed account would be beyond the scope of the present paper. Most of the frequent errors fall into either of two basic categories, both consisting of items where the information needed for correct disambiguation cannot be captured by present model:

(i) disambiguation information is available in local context but model cannot utilize it. In particular, a typical type of behavior of the system is to overweight the unigram frequency scores to the detriment of bigram or trigram values even in places where local context definitely favors an alternative against the unigram frequencies.

(ii) disambiguation information would only be available from higher levels of linguistic analysis. These items are practically irresolvable by the model and constitute good candidates for feature merger in a (performance oriented) tagset.

For (i), one might have recourse to context dependent linear interpolation where, in standard practice, parameters are calculated for various frequency groupings. However, very often all possible alternative analyses of a token are related to low frequency bigrams falling to the same grouping so no distinct parameters are calculated, and the errors persist. We do not pursue the issue further here whether more elaborate partitioning of parameter settings would help solve the problem; instead we opt for a small symbolic preprocessor on the data, which operates with environmentally triggered deterministic rules. This has been built into the system filtering on the input to the stochastic classifier, and is able to achieve about 10% reduction in the average number of errors (0.2% rise in the overall accuracy in the present settings).

6. Conclusions

We have discussed a straightforward method that we hope have demonstrated that a simple tagging scenario based on a second order HMM can effectively cope with morphological disambiguation in an agglutinative language traditionally regarded as unsuitable for such a trivial model. We have shown that appropriate use of the information provided by an external morphological analyzer can solve the ubiquitous data sparseness problem. The advantages of the

[8] Beside being able to robustly handle unseen data.

system are that there is no need for huge dictionary resources (which are in fact practically impossible to prepare, an independent morphological lexicon would need to contain billions of entries), the architecture is able to tag running text robustly on-line, using the language model built from some training corpus, which does not need to be extremely large allowing for low cost manual preparation.

The choice of the tagging tool was motivated by the surprisingly high baseline performance with the brute force unigram relative frequency counts, which predicted that a relatively simple model would be able to give sufficient performance for real life applications, which was exactly to apply a statistical learner trained on labeled examples using Maximum Likelihood Estimates. This is why an HMM and TnT in particular was a comfortable choice for the task, which is rather contrary to intuition which would suggest that a trigram-based Hidden Markov Model is not a suitable method since Hungarian is a relatively free word order language where the constituents are ordered according to the information structure instead of the argument structure of the predicates. However, free word order does not lead to the crash of the HMM approach: once the problems stemming from the highly inflectional nature of Hungarian are overcome, the HMM tagger can give good results.

Some further work is needed to develop an external guesser which will preprocess words genuinely unknown even to the morphological analyzer and provide all possible analyses that Hungarian morphology would allow for a form, instead of the suffix approximating technique built into the tagger, which does not perform well for Hungarian.

It is also worth exploring next how much the good results are language specific, applying the method to other languages of similar characteristics.

7. Acknowledgments

The authors are especially grateful to Thorsten Brants for modifying TnT, which enabled us to use his tagger with the architecture presented. The authors were supported by grant number T026091 of the Országos Tudományos Kutatási Alap.

8. References

Thorsten Brants. 1997. Internal and external tagsets in part-of-speech tagging. In *Proceedings of Eurospeech*, Rhodes, Greece.

Thorsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, Seattle, WA.

Eric Brill and Jun Wu. 1998. Classifier combination for improved lexical disambiguation. In *Proceedings of COLING-ACL'98*, pages 191–195, Montreal, Canada.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. *Computational Linguistics*, 21(4):544–565.

Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven E. Gillis. 1996. MBT: a memory-based part of speech tagger-generator. In *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 14–27, Copenhagen, Denmark.

Péter Dienes and Csaba Oravecz. 2000. Bottom–up tagset design from maximally reduced tagset. In Anne Abeille, Thorsten Brants, and Hans Uszkoreit, editors, *Proceedings of the Workshop on Linguistically Interpreted Corpora*, COLING 2000, pages 42–47.

Tomaž Erjavec, Sašo Džeroski, and Jakub Zavrel. 1999. Morphosyntactic tagging of Slovene: Evaluating pos taggers and tagsets. Technical Report 8018, Dept. of Intelligent Systems, Jožef Stefan Institute, Ljubljana.

Jan Hajič and Barbora Hladka. 1998. Tagging inflective languages: prediction of morphological categories for a rich structured tagset. In *Proceedings of the 36th annual meeting of the ACL – COLING*, Montreal, Canada.

Jan Hajič. 2000. Morphological tagging: Data vs. Dictionaries. In *Proceedings of ANLP-NAACL Conference*, pages 94–101, Seattle, Washington, USA.

Papageorgiou Harris, Prokopidis Prokopis, Giouli Voula, and Piperidis Stelios. 2000. A unified pos tagging architecture and its application to Greek. In *Proceedings of Second International Conference on Language Resources and Evaluation*, Athens.

Julian Kupiec. 1989. Probabilistic models of short and long distance word dependencies in running texts. In *Proceedings of the 1989 DARPA Speech and Natural Language Workshop*, pages 290–295, Philadelphia. Morgan Kaufman.

Julian Kupiec. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6:225–242.

Bernard Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Gábor Prószéky and László Tihanyi. 1996. Humor – a Morphological System for Corpus Analysis. In *Proceedings of the first TELRI Seminar in Tihany*, pages 149–158, Budapest.

Adwait Ratnaparkhi. 1997. A maximum entropy part-of-speech tagger. In *Conference on Empirical Methods in Natural Language Processing*, University of Pennsylvania.

Christer Samuelsson. 1996. Handling sparse data by successive abstraction. In *Proceedings of COLING-96*, Kopenhagen, Denmark.

Dan Tufiş, Péter Dienes, Csaba Oravecz, and Tamás Váradi. 2000. Principled hidden tagset design for tiered tagging of Hungarian. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, Athens.

Dan Tufiş. 1999. Tiered tagging and combined language models classifiers. In F. Jelinek and E. Nöth, editors, *Text, Speech and Dialogue*, Lecture Notes in Artificial Intelligence 1692, pages 28–33. Springer.

Dan Tufiş. 2000. Using a large set of EAGLES-compliant morpho-syntactic descriptors as a tagset for probabilistic tagging. In *Proceedings of Second International Conference on Language Resources and Evaluation*, Athens.

Evelyne Tzoukermann and Dragomir R. Radev. 1996. Using word class for part-of-speech disambiguation. In *Fourth Workshop on Very Large Corpora (WVLC-4)*, pages 1–13, Copenhagen, Denmark. International Conference on Computational Linguistics.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 1998. Improving data driven wordclass tagging by system combination. In *Proceedings of COLING-ACL'98*, pages 491–497, Montreal, Canada.