# PurePos – an open source morphological disambiguator

György Orosz and Attila Novák

MTA-PPKE Language Technology Research Group –
Pázmány Péter Catholic University, Faculty of Information Technology
50/a Práter street, Budapest, Hungary
{oroszgy,novak.attila}@itk.ppke.hu

**Abstract.** This paper presents PurePos, a new open source Hidden Markov model based morphological tagger tool that has an interface to an integrated morphological analyzer and thus performs full disambiguated morphological analysis including lemmatization of words both known and unknown to the morphological analyzer. The tagger is implemented in Java and has a permissive LGPL license thus it is easy to integrate and modify. It is fast to train and use while having an accuracy on par with slow to train Maximum Entropy or Conditional Random Field based taggers. Full integration with morphology and an incremental training feature make it suited for integration in web based applications. We show that the integration with morphology boosts our tool's accuracy in every respect – especially in full morphological disambiguation – when used for morphologically complex agglutinating languages. We evaluate PurePos on Hungarian data demonstrating its state-of-the-art performance in terms of tagging precision and accuracy of full morphological analysis.

## 1 Introduction

Tools called 'Part-of-speech (PoS) taggers' are widely used in current language processing systems as a preprocessing tool. Calling them PoS taggers is a bit misleading, since a number if tagging algorithms are capable of effectively assigning tags to words from a much richer set of morphosyntactic tags than just the basic PoS categories. However, morphosyntactic tagging is still just a subtask of morphological disambiguation: in addition to its tag, the lemma of each word also needs to be identified. Most of currently available taggers only concentrate on determining the morphological tag but not the lemma, thus doing just half of the job. For morphologically not very rich languages like English, and in situations where there is ample training material, a cascade of a tagger and a lemmatizer yields acceptable results. For morphologically rich agglutinating languages – such as Hungarian, Finnish or Turkish – and especially in cases where only a limited amount of training material is available, our results show that a closer integration of the tagger and the morphological analyzer (MA) is necessary to achieve acceptable results.

Halácsy et al. previously demonstrated [8] that a morphological analyzer can improve the accuracy of tagging. Improvement is considerable when disambiguating texts written in agglutinating languages. In our paper, we first introduce the main fields of application where we believe our new open source tool could be used more efficiently than other tools available. We then describe the process of creating our tool. Our implementation is based on algorithms used in TnT [3] and HunPos [7] but the new tool is capable of morphosyntactic tagging and lemmatization at the same time thus yielding complete disambiguated morphological annotation[1]. In addition to the Hidden Markov model (HMM) used for the disambiguation of morphosyntactic tags, the tool includes an interface to an integrated morphological analyzer that not only greatly improves the precision of the tagging of words unseen in the training corpus but also provides lemmata. Finally we compare the performance of PurePos with other tagging tools.

## 2 Need of an integrated morphological analyzer

### 2.1 Agglutinating languages

If we compare agglutinating languages like Hungarian or Finnish with English in terms of the coverage of vocabulary by a corpus of a given size, we find that although there are a lot more different word forms in the corpus, these still cover a much smaller percentage of possible word forms of the lemmata in the corpus than in the case of English. On the one hand, a 10 million word English corpus has less than 100,000 different word forms, a corpus of the same size for Finnish or Hungarian contains well over 800,000 [14]. On the other hand, however, while an open class English word has about 4–6 different word forms, it has several hundred or thousand different productively suffixed forms in agglutinating languages. Moreover, there are much more different possible morphosyntactic tags in the case of these languages (corresponding to the different possible inflected forms) than in English (several thousand vs. a few dozen). Thus data sparseness is threefold: *i*) an overwhelming majority of possible word forms of lemmata occurring in the corpus is totally absent, *ii*) word forms that do occur in the corpus have much less occurrences, and *iii*) there are also much less examples of tag sequences, what is more, many tags may not occur in the corpus at all.

The identification of the correct lemma is not trivial either, especially in the cases of guessed lemmata. But also for words that can be analyzed by the regular morphological analyzer, identifying the lemma can still be a problem. In Hungarian, for example, there is a class of verbs that end in *-ik* in their third person singular present tense indicative, which is the customary lexical form (i. e. the lemma) of verbs in Hungarian. Another class of verbs has no suffix in their lemma. The two paradigms differ only in the form of the lemma, so inflected forms can belong to the paradigm of either an *-ik* final or an non-*ik* final verb and many verbs (especially ones containing the productive derivational suffix *-z/-zik*) have an ambiguous lemma.

### 2.2 Resource poor languages

A great proportion of resource poor languages (that lack annotated corpora) is morphologically complex. For these languages, to create an annotated corpus, an iterative workflow is a feasible approach. First, a very small subset of the corpus is disambiguated manually and the tagger is trained on this subset. Then another subset of the corpus is

[1] i.e. each word is annotated with its lemma and its morphosyntactic tag

tagged automatically and corrected manually, yielding a new, bigger training corpus and this process is repeated. For this workflow to be in fact feasible, a fast turnaround time is needed for the retraining process. In terms of training time, Hidden Markov Model based taggers greatly outperform other tagger algorithms, like maximum entropy and conditional random fields, which take comparatively longer time to train. HMM thus fits better the iterative workflow sketched above. (To be specific, training HunPos, a Markov model based tagger on a 1 million word Hungarian tagged corpus takes less than a minute vs. several hours of training of a maximum entropy based OpenNLP [2] tagger model on the same hardware.) The higher accuracy the tagger used for disambiguation has in the small training corpus scenario, the less time and human resource is needed to develop the annotated corpus.

### 2.3 Web services

Nowadays, an increasing number of Natural Language Processing tools are becoming available as an online web service (Google Translate, OpenCalais, HealthMash and other semantic search engines). Since many of these systems heavily employ a tagger and / or a lemmatizer, a solution is required that yields the most accurate results in the web service setting. The web service scenario imposes an additional constraint on the morphological tagger in comparison with off-line tasks: the text to be processed is not known at the time of the initialization of the tool. This constraint makes the simple solution of loading a table of possible morphological analyses (MT) in the tagger – as it is done in HunPos [7] – a poor substitute for a real morphological analyzer, especially in the case of agglutinating languages, where a morphological table equivalent to the full MA would not be bounded in size due to the recursive nature of the morphology. Reinitializing the tool for each request loading a table optimized to the actual request leads to an unacceptable response time. On the other hand, using a constant table is costly in terms of process memory and inefficient in terms of coverage. When providing morphological annotation as a web service, another feature comes handy: incremental training of the tool with additional annotated text.

## 3 Implementation

### 3.1 Background

Our goal was to find or create a morphological disambiguation tool that *i*) contains an interface to integrate a morphological analyzer, *ii*) performs full morphological disambiguation including lemmatization, *iii*) is open source, *iv*) can handle Unicode input, *v*) can efficiently handle (including lemmatization of) both words unknown to the morphological analyzer and ones missing from the training corpus, *vi*) is fast to train and *vii*) is easy to train and use.

Although there are a few tools (see the end of this section) that meet part of the above requirements, none of them satisfies all, therefore, we decided to build a morphological tagger implemented in Java, a widely known and platform independent object oriented programming language. Java has an additional advantage of making an easier integration with popular and language independent natural language processing tools like UIMA [19] and GATE [5]. Choosing Java as the base of our implementation our tool has the advantage of universally representing characters of texts written in any language covered by Unicode. We modelled our implementation on HunPos, an open source HMM tagger written in OCaml. Enriching HunPos with the required functionality would probably have been the easiest way to attain our goal. However, the lesser known programming language in which it is implemented could be a hindrance to the portability, integration and further customization of the tool.

HunPos itself is an open source reimplementation and enhancement of Thorsten Brants' TnT. It is capable of *i*) using a generalized smoothed n-gram language model (while TnT has trigrams only), *ii*) context sensitive emission probabilities (another enhancement over TnT), *iii*) clever tricks like applying different suffix guesser models to capitalized and lower case words, *iv*) different emission models for ordinary words and special tokens that contain digits and other non-letter characters, and, *v*) in order to improve tagging speed, a beam search instead of an unconstrained Viterbi search. In addition, HunPos can load a morphological table at initialization time that can be used to emulate the operation of an integrated morphological analyzer if all the word forms in the text to be processed had been listed along with all their possible tags using the analyzer in an off-line manner before initializing the tagger. However, this kind of poor man's MA is not applicable in web service scenarios as noted above. A special enhanced version of TnT had a similar table loading mechanism that was used by Oravecz and Dienes [14]. However, as far as we know, that version was not made accessible to the public.

A drawback of HunPos is that it can only process 8 bit input, it handles case distinctions incorrectly if we attempt to use it on UTF-8 text. There is a degradation of performance even when trying to use it for UTF-8 Latin text with accented letters but using it for Cyrillic text, especially for a language with accented characters, would be quite a challenge. Moreover a model which is trained on 8-bit encoded text, cannot be correctly used for a text which has a different character encoding.

Candidates for a starting point for our development could also have been one of the following tools: TriTagger [12, 11], an open source trigram HMM tagger implemented in Java, has a language specific integrated morphological analyser (for Icelandic) – called IceMorphy [10] – but it does not perform lemmatization and lacks a few clever tricks implemented in HunPos, which seem to boost the performance of that tagger significantly. Another open source trigram HMM tagger [18, 17] implemented in C/C++ is included in the Apertium [1] rule based machine translation toolset. It performs full morphological disambiguation by taking morphologically analyzed input and disambiguating it. This tagger heavily relies on the concept of ambiguity classes, which implies that during training a morphological analyzer is necessary. In addition, words unknown to the morphological analyzer (that were left unannotated in the input) are handled in a uniform manner not taking the form of the word into consideration. Handling unknown words is thus left to the preprocessor/morphological analyzer.

## 3.2 Reimplementation

We implemented the following new features in PurePos in addition to the integration of a morphological analyzer interface: *i*) a lemma guesser for words unseen in the training corpus and unknown to the MA and *ii*) incremental training, i.e. additional training data can be added to the tagger model without a full recompilation of the model.

Incremental training is made possible by the fact that the model generated during training is only normalized right before tagging (i.e. we serialize the model without normalizing it). The calculation of model parameters and normalization takes very little time when loading the model.

When revising the original HunPos code, we also discovered an implementation error in it. The authors use the special tokens lexicon erroneously: the lexicon is built up using names of special token classes, but in the tagging process, the algorithm searches for special tokens themselves in the lexicon instead of the name of the class to which they belong. This erroneous behaviour was fixed in PurePos.

## 3.3 Using the integrated morphological analyzer

The integrated morphological analyzer has a twofold role in the tagger. On the one hand, it helps to determine the correct tag for words unseen in the training corpus by eliminating false tag candidates generated by the suffix guesser. As Halácsy et al. [7] showed and as is clearly demonstrated in the evaluation section, the knowledge of possible tags for a particular token yields better accuracy during tagging, since the suffix guesser may have many false guesses that can be filtered out by using a morphological table. Keeping this feature in our implementation we also introduced an interface for a morphological analyzer, which is used in the same way.

Since the identification of lemmata is part of our goal, we need to select the most probable lemma that corresponds to each selected tag and token. In the case of words missing from the lexicon of the integrated morphological analyzer, possible lemmata are generated by a morphological guesser. This guesser gets its input from the training data, and learns [lemma transformation, tag] pairs (TTP) for the given suffixes. A transformation is represented as the difference between the word and the lemma. A reverse trie of suffixes is built in which each node can have a weighted list containing the corresponding TPPs. Once the training is finished, the lemma guesser is used along with the MA to calculate the possible lemmata. For tokens recognized by the MA, the analyzer supplies lemmata, while in the case of out-of-vocabulary (OOV) words, the lemma guesser is used to generate possible lemmata. Each guess contains a morphosyntactic tag with which it is compatible. Having the guesses for each token and the best tag sequence for the sentence, for each token the most probable compatible lemma (having the same morphosyntactic tag) is selected, where probability is estimated as the relative frequency of the lemma (given its main PoS category) in the training set.

The tagger uses the same rich tagset that is used in the training corpus and the MA without any mapping. In the case of the Szeged corpus this amounts to 1030 different tags.

## 4 Evaluation

We compared our implementation with other state-of-the-art PoS taggers in terms of tagging accuracy, such as HunPos and the maximum entropy based OpenNLP tagger. The evaluation was done on a modified version of the Hungarian Szeged Corpus [4] in which the morphosyntactic annotation was converted into a form that is compatible with that of the HUMor [15] morphological analyzer that we interfaced with our tagger. Conversion was complicated by the fact that many closed class grammatical words like conjunctions, pronouns, postpositions and sentence adverbials are differently categorized in the two formalisms. At the same time, lemmata also had to be converted to be compatible with the used analyzer's output. HUMor can yield a richer analysis including a number of productive derivational affixes (e.g. participles, gerunds etc.) which results in a different lemma than the one in the original Szeged Corpus. Since this conversion was done automatically – that might contain errors in the case of lemmata – we had to create a handcrafted test set, on which the lemmatization tests were run. In addition to reporting accuracy of morphosyntactic tagging, we also present the results of lemmatization accuracy and a combined accuracy of morphological annotation of the tool (i.e. both the lemma and the tag must be identical to that in the gold standard for the annotation of the token to be accepted as correct).

Comparison of PurePos with HunPos and OpenNLP is only possible in terms of tagging accuracy as the latter perform no lemmatization. We know of only one freely available tool for Hungarian: magyarlanc [22] that performs full morphological annotation using a similar approach. Direct comparison of its performance with PurePos is not possible due to the difference in the set of tags and lemmata produced by the two systems, so, unfortunately, we currently cannot present a comparison of the accuracy of the two systems. Nevertheless, PurePos has a few clear advantages: due to its dependence on the Stanford tagger, magyarlanc has a more restrictive license than PurePos, it includes a lot of machinery irrelevant to the tagging task and cannot be readily used for other languages or annotation schemes than the one built into it.

Since our algorithm is entirely modelled on that of HunPos, it doesn't really make sense to compare them in the regular way: using a morphological table of the text to be processed and using a built in MA to analyze the same words yield the same result. If we do this we get almost exactly the same results. The differences in the output of HunPos and PurePos mainly originate in having fixed a few relatively unimportant implementation errors in HunPos and another part of them are cases, where there is an indeterminacy in the exact set of paths to follow during beam search. While small bug fixes result in about less than 0.01% improvement (and only in those cases where the training set is small enough) the beam search indeterminacy accounts for even less difference.

Table 1 and 2 present tagging, lemmatization and combined (full morphological disambiguation) accuracy of PurePos. The version termed "Guesser" in the tables uses only information learned from the corpus both for tagging unseen words and lemmatization of all tokens. The version termed "Gue.+MT" uses an off-line generated morphological table that lists all possible tags of all tokens in the test set that was generated using the HUMor morphological analyzer. However, this version still uses only the lemma guesser for lemmatization. "Gue.+MA" uses the integrated MA both for tagging

|  | Guesser | Gue.+MT | Gue.+MA |
|---|---|---|---|
| Tagging acc. | 98.14% | 98.99% | 98.99% |
| Lemmatization acc. | 90.58% | 91.02% | 99.08% |
| Combined acc. | 89.79% | 90.35% | 98.35% |

**Table 1.** Morphological disambiguation accuracy per token

|  | Guesser | Gue.+MT | Gue.+MA |
|---|---|---|---|
| Tagging acc. | 75.08% | 85.21% | 85.21% |
| Lemmatization acc. | 29.17% | 30.74% | 87.13% |
| Combined acc. | 26.17% | 28.05% | 78.11% |

**Table 2.** Morphological disambiguation accuracy per sentence

|  | Accuracy |
|---|---|
| PurePos (with MA) | 98.99% |
| PurePos (without MA) | 98.14% |
| OpenNLP perceptron | 97.16% |
| OpenNLP maxent | 96.45% |

**Table 3.** Comparison of part-of-speech tagging accuracy

and lemmatization. In this setup, the guesser is only used for OOV words. Table 1 shows token accuracy and table 2 shows sentence accuracy using 90% as training set and 10% as test set of the modified Szeged Corpus. The results clearly show that using only the lemma guesser yields mediocre results, but if the MA is applied for lemmatization for words in its vocabulary, accuracy significantly increases. Comparing PurePos (with and without MA) with OpenNLP (in table 3) on the same training and test set (both with the Maximum Entropy and Perceptron learning package) shows that our system is competitive with nowadays popular systems.

In addition, we examined the learning curve of the system modelling the incremental creation of an annotated corpus from scratch. For this we followed the iterative workflow described above. We used the following systems: *i*) the reimplemented HunPos algorithm with no analyzer integrated (as a baseline) using the trained lemma guesser to perform lemmatization, *ii*) the same with a constant morphological table containing the 100000 most frequent words from the Hungarian Webcorpus [9, 6] (an unannotated corpus independent of Szeged Corpus) and *iii*) full PurePos with the integrated analyzer that performs lemmatization as well.

The integrated basic lemma learning algorithm – in the case of a training set of an adequate size – has a lemmatization accuracy of about 80–90% that can be used as a baseline. Its relatively low performance is due to overregularization of frequent irregular words. Figure 1 and 2 show that there is a clear advantage of having an integrated morphological analyzer to handle word forms missing from the training corpus and to do lemmatization. The advantage is striking at the initial phase of the corpus creation process. Although it becomes less pronounced as more and more training corpus is available, even with a 1 million word training corpus the tagger combined with the MA produces only about half as many tagging errors as the version containing no analyzer.

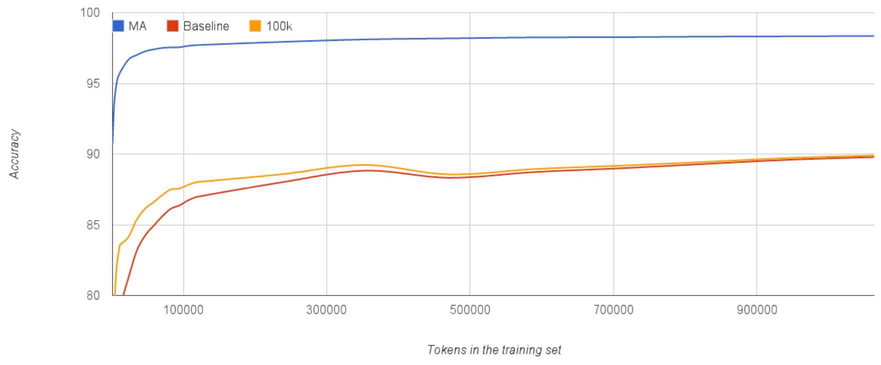

**Fig. 1.** Learning curve of full disambiguation accuracy

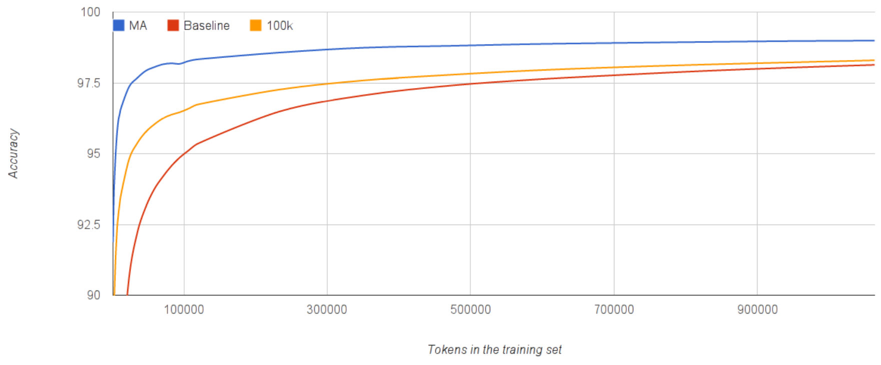

**Fig. 2.** Learning curve of PoS tagging accuracy

Tables 4 and 5 show the performance of the tool in a hypothetical web service scenario. We kept the – nine to one split – training and testing set, and compared the version with an integrated MA with variants which either do not employ any morphological information (as a baseline) or use a fixed morphological table (MT, generated from an independent corpus). In the latter case the morphological table is generated from the Hungarian Web corpus. For this we took the first 10000 ("MT-10k"), 30000 ("MT-30k"), 100000 ("MT-100k") most frequent words from the Web corpus. "MT-100k*" contains the most frequent 100000 words of the Web corpus that do not occur in the training set.

Our results show that in a web service setting using an integrated MA increases performance significantly. The increase in efficiency is also noticeable if only the accuracy of PoS tagging is considered. Having a fixed morphological table provides little improvement in tagging performance. This is not very surprising in the MT-10k, MT-30k and MT-100k cases, as most common words in the table are already in the training

| | Lemmatization | Tagging | Combined |
|---|---|---|---|
| Baseline | 90.58% | 98.14% | 89.79% |
| MT-10k | 90.58% | 98.14% | 89.79% |
| MT-30k | 90.58% | 98.17% | 89.81% |
| MT-100k | 90.64% | 98.30% | 89.90% |
| MT-100k* | 90.72% | 98.39% | 89.97% |
| MA | 99.07% | 98.99% | 98.35% |

**Table 4.** Morphological disambiguation accuracy (per token) in a web service setting

| | Lemmatization | Tagging | Combined |
|---|---|---|---|
| Baseline | 29.17% | 75.08% | 26.27% |
| MT-10k | 29.17% | 75.15% | 26.27% |
| MT-30k | 29.23% | 75.45% | 26.33% |
| MT-100k | 29.47% | 76.85% | 26.63% |
| MT-100k* | 29.64% | 78.17% | 26.87% |
| MA | 87.13% | 85.21% | 78.11% |

**Table 5.** Morphological disambiguation accuracy (per sentence) in a web service setting

corpus. However, the integrated MA yields a considerable improvement even compared to the MT-100k* system.

## 5 Conclusion

In our paper, we presented PurePos a new stochastic morphological tagger that is freely available, is open source with a permissive LGPL license and has an interface for integrating a morphological analyzer. The tagging performance of the tool is higher than that of OpenNLP and as high as that of HunPos, another open source tagger, however, it is easier to integrate and modify due to being implemented in Java. It can handle Unicode input and it performs full disambiguated morphological analysis, not just morphosyntactic tagging. It is fast to train and use thus we hope that it will be a tool of choice for corpus annotation projects for less resourced languages. The integrated MA interface makes it suitable for providing high performance morphological annotation as a web service. The tool is available at http://nlpg.itk.ppke.hu/software/purepos.

## Acknowledgements

We would like to thank Balázs Indig for his continuous help in understanding the HunPos source code, Péter Halácsy for having implemented HunPos and Nóra Wenszky for her comments on preliminary versions of this paper. This work was partially supported by the grants TÁMOP-4.2.1.B-11/2/KMR-2011-0002 and TÁMOP-4.2.2/B-10/1-2010-0014.

# Bibliography

[1] Carme Armentano-Oller, Rafael C. Carrasco, Antonio M. Corbí-Bellot, Mikel L. Forcada, Mireia Ginestí-Rosell, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Gema Ramírez-Sánchez, Felipe Sánchez-Martínez, and Miriam A. Scalco. Open-source Portuguese-Spanish machine translation. In R. Vieira, P. Quaresma, M.d.G.V. Nunes, N.J. Mamede, C. Oliveira, and M.C. Dias, editors, *Computational Processing of the Portuguese Language*, pages 50–59. Springer-Verlag, Itatiaia, Brazil, May 2006.

[2] Jason Baldridge, Thomas Morton, and Gann Bierner. The OpenNLP maximum entropy package. Technical report, 2002.

[3] Thorsten Brants. TnT - A Statistical Part-of-Speech Tagger. In *Proceedings of the sixth conference on Applied Natural Language Processing*, pages 224–231. Universität des Saarlandes, Computational Linguistics, Association for Computational Linguistics, 2000.

[4] Dóra Csendes, János Csirik, and Tibor Gyimóthy. The Szeged Corpus: A POS tagged and syntactically annotated Hungarian natural language corpus. In *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora LINC 2004 at The 20th International Conference on Computational Linguistics COLING 2004*, pages 19–23, 2004.

[5] Hamish Cunningham, Robert J Gaizauskas, and Yorick Wilks. A General Architecture for Language Engineering (GATE) - a new approach to Language Engineering R&D. *International Conference On Computational Linguistics*, page 52, 1996.

[6] Péter Halácsy, András Kornai, László Németh, András Rung, István Szakadát, and Viktor Trón. Creating open language resources for Hungarian. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, 2004.

[7] Péter Halácsy, András Kornai, and Csaba Oravecz. HunPos: an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 209–212, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[8] Péter Halácsy, András. Kornai, Csaba Oravecz, Viktor Trón, and Dániel Varga. Using a morphological analyzer in high precision POS tagging of Hungarian. In *5th edition of the International Conference on Language Resources and Evaulation*, pages 2245–2248, 2006.

[9] András Kornai, Péter Halácsy, Viktor Nagy, Csaba Oravecz, Viktor Trón, and Dániel Varga. Web-based frequency dictionaries for medium density languages. In Adam Kilgarriff and Marco Baroni, editors, *Proceedings of the 2nd International Workshop on Web as Corpus*, 2006.

[10] Hrafn Loftsson. Tagging icelandic text using a linguistic and a statistical tagger. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, NAACL-Short '07, pages 105–108, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[11] Hrafn Loftsson, Sigrún Helgadóttir, and Eiríkur Rögnvaldsson. Using a morphological database to increase the accuracy in pos tagging. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 49–55, Hissar, Bulgaria, 2011. RANLP 2011 Organising Committee.

[12] Hrafn Loftsson and Eiríkur Rögnvaldsson. Icenlp: a natural language processing toolkit for icelandic. In *Proceedings of InterSpeech 2007, Special session: "Speech and language technology for less-resourced languages*, pages 1533–1536, Antwerp, Belgium, 2007. ISCA.

[13] Attila Novák, György Orosz, and Indig Balázs. Javában taggelünk. In Attila Tanács and Veronika Vincze, editors, *VIII. Magyar Számítógépes Nyelvészeti Konferencia*, page 336, Szeged, 2011.

[14] Csaba Oravecz and Péter Dienes. Efficient Stochastic Part-of-Speech Tagging for Hungarian. In *Third International Conference on Language Resources and Evaluation*, pages 710–717, 2002.

[15] Gábor Prószéky and Attila Novák. Computational Morphologies for Small Uralic Languages. In *Inquiries into Words, Constraints and Contexts.*, pages 150–157, Stanford, California, 2005.

[16] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*, volume 1, pages 133–142, 1996.

[17] Felipe Sánchez-Martínez, Juan Antonio Pérez-Ortiz, and Mikel L. Forcada. Using target-language information to train part-of-speech taggers for machine translation. *Machine Translation*, 22:29–66, 2008.

[18] Zaid Md Abdul Wahab Sheikh and Felipe Sánchez-Martínez. A trigram part-of-speech tagger for the Apertium free/open-source machine translation platform. In Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, and Francis M. Tyers, editors, *Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation*, pages 67–74, Alicante, Spain, 2009. Universidad de Alicante. Departamento de Lenguajes y Sistemas Informáticos.

[19] Götz Thilo and Oliver Suhre. Design and implementation of the UIMA common analysis system. *IBM Systems Journal*, 43:476–489, 2004.

[20] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In Marti Hearst and Mari Ostendorf, editors, *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180, Edmonton, Canada, 2003. Association for Computational Linguistics.

[21] Veronika Vincze, D. Szauter, A. Almási, G. Móra, Z. Alexin, and J. Csirik. Hungarian Dependency Treebank. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*, pages 1–5, 2010.

[22] János Zsibrita, István Nagy, and Richárd Farkas. Magyar nyelvi elemző modulok az UIMA keretrendszerhez. In Attila Tanács, Dóra Szauter, and Veronika Vincze, editors, *VI. Magyar Számítógépes Nyelvészeti Konferencia*, pages 394–395, Szeged, 2009. Szegedi Tudományegyetem.