HunTag3: a general-purpose, modular sequential tagger – chunking phrases in English and maximal NPs and NER for Hungarian

István Endrédy, Balázs Indig

Pázmány Péter Catholic University, Faculty of Information Technology and Bionics
MTA-PPKE Hungarian Language Technology Research Group
50/a Práter Street, 1083 Budapest, Hungary
{endredy.istvan,indig.balazs}@itk.ppke.hu

Abstract

The available statistical tools for sequential tagging, especially for maximal NP (noun phrase) chunking, perform well, and the current popular methods usually involve CRFs. However, all the freely available tools consist of tightly coupled modules. Combining well-known methods (HMM, Maximum Entropy, first and second order transition models, beam search, CRFs, different smoothing techniques) in a general-purpose sequential tagger would help to find the best combination for each language and task. This paper introduces an updated, modular, universal sequential tagger which is evaluated using combinations of multiple methods for chunking phrases in English (arbitrary phrase identification) and maximal NPs and NER for Hungarian. Our simple trigram based MEMM solution amended with enhanced POS categories based on WordNet beat first order CRF for Hungarian NP chunking (best ever F-score 93.59%), but the same method was slightly outperformed for English.

1. Introduction

As the first step towards the syntactic parsing of sentences, especially if one only needs specific parts of a sentence (in our case Noun Phrases (NP)), it is trivial to divide the sentence into syntactically linked parts of subsequent tokens (chunks). In a later step, if needed, one may try to find the internal structure of these chunks and their relations. This task is therefore called shallow parsing, which can be interpreted as *sequential tagging*. (See Section 2. for detailed introduction to chunking.)

This kind of labeling is applied in many different tasks of NLP (e. g. Named Entity Recognition (NER), NP chunking, POS-tagging). Different methods have been applied to individual tagging tasks, however, they all share the same general idea (i.e. to issue one unambigous label per token). There even exist attempts that apply some well-known methods to tasks other than they were originally developed for, like a POS tagger for NP chunking (Brill, 1992), even for loosely related tasks like parsing with a sequential tagger (shallow parser) (Charniak, 2000).

The available tools, however, consist of a set of tightly coupled methods, which try to provide a product that 'just works' and might even use the same basic techniques for NER, POS-tagging and NP chunking (Baldridge, 2005). Even though a method can be used for many purposes, it might produce different results. Instead of working with tightly coupled modules of multiple tools, we introduce a single tool, *HunTag3*, that consists of loosely coupled modules, such as feature generation, *pluggable unigram model* (currently maximum entropy), and first- or second-order transition models using the Viterbi algorithm (for details see Section 3.), in order to be able to select the appropriate method for the given task and language. Apart from the features described in Section 4.2. our tool is language-independent and might be applied to other languages easily, but demonstrating its performance in other languages is out of the scope of this paper.

This paper provides the preliminary results (achieved with the aforementioned tool) of experiments on multiple chunking tasks in Hungarian and in English in terms of newly found features as well as the applied methods. We tested the English dataset also to show that the gain in F-score compared to our baseline affects the English results as well, and to compare our method with that of CRF.

We use a specialized variant of TnT (Brants, 2000) in conjunction with second-order (trigram) transition model for NP chunking and NER, which greatly outperforms the original version in NP chunking and its performance is comparable with the currently popular first-order (bigram) CRF-based tools (Okazaki, 2007).

2. Background

Chunking, in general, is a fairly standardized task since the *CoNLL-2000 shared task* (Tjong Kim Sang and Buchholz, 2000) which is the de-facto standard for measuring and comparing taggers in English. The problem is also known as *IOB tagging* of NPs as each chunk can be formulated as a sequence of the following tags: *B* indicates the beginning, *I* the inside and *E* the end of a (syntactically) correlated token sequence respectively. Additionally, one may distinguish the outside (*O*) of a sought sequence and sequences that span only one token (*S* or *1*) (for other representations see (Shen and Sarkar, 2005). In addition, each marked sequence has a type that corresponds to the name of the parsing unit that the syntactic parse would issue. In general, the task is to assign these labels to tokens correctly. This idea was successful and made it possible to use the same formalism to tag Named-Entities in *CoNLL-2003 shared task* (Tjong Kim Sang and De Meulder, 2003).

2.1. From HunTag to HunTag3: General architecture

Our starting point was an existing tool: *HunTag* (Recski, 2014; Recski and Varga, 2009), a general-purpose sequential tagger for NLP written in Python 2. This tool was used for NER and NP chunking as well and is the current state of the art for Hungarian (Simon, 2013). Its architecture enables us to swap modules deep inside the engine to experiment with other well-known Machine Learning methods and practices. HunTag combines the *linear SVM classifier (Maximum Entropy)* of *Liblinear* (Fan et al., 2008) and *first-order Hidden Markov Models* (bigrams), which are also called *Maximum Entropy Markov Models* (MEMM) in the literature (McCallum et al., 2000).

In response to the increasing shift towards Python 3 as the default Python version, we have rewritten HunTag in Python 3 and it has been renamed to *HunTag3* [1]. The Liblinear library was switched to *Scikit-learn* (*LinearRegressionClassifier*), a Machine Learning library in Python (Pedregosa et al., 2011) to enable the possibility to use and compare other classifiers and to further ease later development. The non-standard sparse matrix representation of Liblinear was switched to sparse matrices of *SciPy* (Jones et al., 2001–) in conjunction with arrays of *NumPy* (Van Der Walt et al., 2011), two more convenient implementations of matrices in Python. Pluggable classifiers are only required to support the sparse matrix type of SciPy as input and to provide probability distribution of labels as output conforming the *Scikit-learn API*. Differences between Huntag and HunTag3 are shown in Table 1.

As Scikit-learn currently does not support *Conditional Random Fields (CRF)*, which overcomes the weaknesses of MEMMs (Lafferty et al., 2001), we created an interface in HunTag3 to generate featurized input suitable for independent tools like *CRFsuite* (a first-order CRF learner tool (Okazaki, 2007)) to be able to compare our results.

We used CRFsuite because it is a fast, reliable and compact CRF implementation. Since it cannot be integrated by design, the tool required two featurized input files: one for training and one for testing. Our results concerning CRFs, which yielded some improvement, are from this program.

For the feature names we followed the data format of CRFsuite, because it met our requirements, is more accepted and is more human readable than the original homebrewed feature name convention of HunTag.

Due to the switching to Scikit-learn library, the required amount of memory for the training is lowered, but the training time is increased. Technical changes in HunTag3 include UTF-8 support and Python 3 support.

3. Trigram on IOB labels

In general, the processing is divided into the following steps: the input is first featurized and then it can be processed either with external tools – where the order of the transition model is up to the external tool – (see *--toCRFsuite* option in the documentation) or internally with the available methods. When using the internal processing, the remaining task is divided into two steps: the unigram and the transition models. In the unigram model, we favored existing tools and we mainly followed the original idea of HunTag, as it is discussed in the previous section.

In the transition model for choosing the most probable IOB sequence in HunTag, the first-order (bigram) Viterbi algorithm is used as it is done in other available tools (e. g. (Finkel et al., 2005) and (Baldridge, 2005)).

We modified it to a pluggable algorithm to ease later replacement. This made it possible to switch to and fro between the bigram and the trigram versions, the latter is using trigrams as formalised in TnT for POS tagging (Brants, 2000) with additional check for boundary symbols at the end of sentences to prevent 'loose end'. Our approach differs from TnT in the lexical probability distribution (emission) which comes from our pluggable unigram model.

In theory, plain IOB labels do not have many possible valid transitions to indicate larger window size. But, in practice, typed and lexicalized IOBES (or SBIEO) labels do (see (Molina and Pla, 2002)), especially those which mark adjacent ones, where the boundary is mistaken.

The following two examples demonstrate this problem for Hungarian. NPs in Table 2 and in Table 3 differ only in the case markings of their nouns, but the resulted NPs are a single bigger one in the former case (Table 2), and two smaller ones in the latter (Table 3) case. These two cases are commonly mixed up.

NP				
DET	NOUN.DAT	DET	**NOUN.POSS**	ADJ
A The	gyereknek kid	a the	tolla his pen	piros. is red.

Table 2: Hungarian example for one NP which contains two nouns. (*The pen of the kid is red.*) This sentence contains one NP with two subsequent nouns.

NP		**NP**		
DET	NOUN.NOM	DET	**NOUN.ACC**	VERB
A The	gyerek kid	a the	tollat his pen	fogja. grabs.

Table 3: Hungarian example for subsequent NPs which differ only in the case (*The kid grabs the pen.*)

We used deleted interpolation for smoothing as Brants did, but our program is open to evaluate other smoothing methods. However, smoothing is likely to favor invalid IOB sequences using plain IOB tags without lexicalizing. Even if the size of the training set is small, there is still a high chance to rank up invalid IOB sequences. We report 0 invalid transitions with HunTag3 (see Section 5.2.).

4. Engineering features

The format of the input sentences follows the typical formalism: each word is in a new line, with tab separated values (word, stem, pos, any other features), and every sentence ends with an empty line. There can be hundreds of features per token (even without their combinations). As a result, the search space is huge. Therefore, manual feature selection and the fine-tuning of the chunking process becomes difficult.

[1] HunTag3's source code and documentation are available at: `https://github.com/ppke-nlpg/HunTag3`

	HunTag	**HunTag3**
encoding	Latin-2	UTF-8
language	Python2	Python3
feature matrices	cType	NumPy/SciPy arrays
unigram model	Liblinear	Scikit-learn, LinearRegressionClassifier
feature naming convention (featurized input data for external programs)	homebrewed	CRFsuite format
file format of the configuration	homebrewed	YAML
memory usage on Mihaltz's test/traing splitting (Miháltz, 2011)	4.2GB	4GB (5% lower)
training time on Mihaltz's test/traing splitting (Miháltz, 2011)	127min	154min (18% higher, due to Scikit-learn)

Table 1: Improvements of HunTag3 compared to HunTag

Our method generated many features into the input, and used HunTag3 to aid the selection process. We used manual feature selection during feature development, because it was important to control and understand the inner working of assignments between features and labels.

To further aid this process, we developed a new mode in HunTag3, called *most informative features*, identical to the one found in NLTK (Bird et al., 2009). It generates a feature rank by computing $P(feat = val|label)$ probability for each <feature, value (with the possibility to consider negative correlation), label> triplet. Then, the maximum of this probability will be divided by the minimum for each feature and finally sorted to get the best candidates.

This output is useful for inspecting feature quality. For example, this function can show whether value VERB of the feature POS correlates with a given output label (in CoNLL-2000 with B-VP) or not. Early elimination of the useless features (i. e. before training) reduces training space, which makes the process faster and more lightweight (even suitable for older machines).

4.1. Word-level features

Word-level features can be very useful for certain word classes which consist of a single token (e. g. punctuation marks, verbs, OOVs). Although the frequencies of these classes are low, but still, these word-level features can be explored and used. Nevertheless, a word (that may belong to a chunk class that spans multiple tokens) alone can hardly provide satisfactory information for classification, even if it has many features. NP chunking needs the context, just like humans: nobody can tell whether a word belongs to an NP tag or not: it depends on the context.

For example, the word *dog* can be either at the beginning (***dog** with happy face*) or inside (*my old **dog's** house*) or at the end (*my **dog***) of an NP as well. Additionally, in English, most nouns have a verb homonym as well, which makes the task even more complicated (if we do not rely on POS tagging): *"I will **dog** him for the rest of his life."*

As a result, in most cases, a word alone cannot determine its own IOB label by using its own features only. It needs feature sequences of previous and following tokens. HunTag has an option to create a custom Python function which builds these feature sequences with arbitrary size and content. This makes it possible to easily create features for an arbitrary language or formalism. The same method can be applied not only in MEMMs, but in CRFs: it is called graphical features, and it is used for example for German NP chunking (Roth and Clematide, 2014).

Naturally, as the size of the context window increases, the search space and running time of the training process do as well. It is not necessary to emit every n-gram of features, but specific features should be defined for special cases. We handle more than 2 million features in our program for Hungarian chunking. However, feature selection may cause overfitting, therefore a 10-fold cross validation was done to verify the significance of the results.

4.2. Useful features for Hungarian

In Hungarian, NPs can stand at any position of the sentence, even next to each other. Our error analysis (based on the classification of 150 randomly sampled errors) has shown, that a typical type of error was to join two different NP chunks at two neighbouring nouns, despite their different case markings. We found that the relations of neighbouring chunks are at least as important in terms of IOB tags as the relations of POS-tags and NPs.

Therefore, an explicit feature was assigned when a noun has different case marking from its adjacent noun. Second, when investigating false negative errors of HunTag (chunks not detected correctly), we found that possessors and their possessees are often missed. Accordingly, a feature was introduced which creates a part-of-speech sequence that connects possible possessors and possessees when a word has the possessor tag in its part-of-speech category. Similar ideas were evaluated recently without success in German (Roth and Clematide, 2014).

We also defined other language-specific features based on Hungarian grammatical behaviour of NPs. First of all, *the last determinant* feature: each word gets a feature with the sequence of the part-of-speech tags from the last determinant (if there was any). Second, in the MSD formalism the POS tag for participles did not differ from that of adjectives. A different tag was assigned for each occurrence manually, because these words behave differently.

Some of these ideas were transferred to English as well.

4.3. WordNet helps to define new features

Several types of features were used to augment the original set of part-of-speech tags. Features were extracted from the MetaMorpho rule-based translation system (Novák et al., 2008), such as *countable/uncountable*, *animate* or not, *abstract* or not, etc.

The basic idea was to define good features which emit mostly the same IOB labels. *WordNet* (Miller, 1995; Miháltz et al., 2008) was used to define such new features automatically. First, *hyponyms* of each word were generated as features. Second, the *most informative features* function was used to list features that would be useful for IOB purposes. Positive correlations with labels for English on the CoNLL-2000 dataset can be seen in Table 4.

WN synset	freq.	connected iob labels	rates
mister.n.01	785	B:767 / I:18	B:98% / I:2%
nation.n.03	63	I:63	I:100%
number.n.11	90	B:1 / I:89	B:1% / I:98%
country.n.04	67	B:1 / I:66	B:1% / I:99%
period.n.05	84	B:2 / I:82	B:2% / I:98%
day.n.10	186	B:7 / I:179	B:4% / I:96%
month.n.02	273	B:9 / I:264	B:3% / I:97%

Table 4: Examples for feature suggestions generated from WordNet (English on the CoNLL-2000 dataset)

If a WordNet synset correlates with an IOB label, then it can be used as a new feature or as a refined 'POS category' for IOB labeling. For instance, the category of nouns can be divided into finer categories (noun_nation, noun_country, etc.) on demand, where it is desirable.

These *WordNet suggestions* were generated for Hungarian too, and were used as new features. But these suggestions can also be reviewed by linguists, motivating them to refine POS categories to be more IOB friendly.

5. Results

5.1. English chunking

The current English state-of-the-art tagger is *SS05* (Shen and Sarkar, 2005), which achieves an F-score of 94.01% on the CoNLL-2000 dataset (Tjong Kim Sang and Buchholz, 2000). This dataset was used to evaluate the modified tagger engine on *arbitrary phrase identification*. Overall, HunTag3 performed better than HunTag, but both of them were slightly outperformed by CRFsuite. During the cross validation, all three methods turned out to be significantly different according to the *t-test*. Still, CRFsuite could not beat the state of the art (SS05) (see Table 5).

The ranking of the methods involving HunTag3 in this test differs from those described in Sections 5.2. and 5.3. In this test, applied to English chunking, CRF performed best, but it is achieved by a certain constellation of the modules in HunTag3. However, introducing modularity has the beneficial effect that each module can be switched easily depending on the actual task and language. Thus, further results are produced by different, task- and language-specific, use of the modules in our program.

method	F-score (%)
HunTag	91.38
HunTag3, CRFsuite	**93.42**
HunTag3, bigram	92.79
HunTag3, trigram	93.41
SS05 (state of the art)	94.01

Table 5: Results of HunTag3 in English with CoNLL-2000 compared to HunTag. (The state of the art is also listed.)

5.2. Hungarian chunking

In Hungarian, the current state-of-the-art chunker is HunTag (Recski, 2014; Recski and Varga, 2012), which uses the KR tagset (Kornai et al., 2004) and yields 90.28% F-measure (2nd column of Table 6). This was the baseline in our tests. Hungarian test data are based on the Szeged Treebank (Csendes et al., 2005), the biggest Hungarian manually annotated (MSD morphosyntactic descriptions (Erjavec, 2010)) corpus which contains about 80,000 sentences with 326,000 annotated maximal NP chunks.

Tagging systems can be compared if they work on the same corpus, with the same training/test set splittings, and also with the same part-of-speech tagset. Therefore, a different train/test set splitting from a previous NP chunker comparison for Hungarian based on the MSD tagset (Miháltz, 2011) was tested as well, which was reused to be able to compare the results (1st column of Table 6).

In all tests, IOBES labels were used in conjunction with the original chunk types conforming to the referred papers. We optimized the features on a development set, which is a held-out set of 10% from the training set. Using trigrams and smoothing, we achieved some gain in F-scores and window size three produced the best results for Hungarian on the previously used train/test splittings.

Our best F-score, which is the state-of-the-art score so far was 93.59% on Miháltz's test set (Miháltz, 2011) (see 1st column of Table 6). Our detailed experiments on the MSD dataset show that the most notable improvement was the correct recognition of neighboring chunks. The features mentioned in Section 4.2. alone increased the F-score with 1.70% (3rd column of Table 6). This result confirmed our hypothesis to look for new categories suggested by WordNet (Section 4.3.). The rewritten HunTag3 engine itself resulted in 0.19% improvement in average (2nd column of Table 6). We evaluated the original TNT as well, to compare it to our methods. In agreement with the previous English results (Shen and Sarkar, 2005), TNT is greatly outperformed by HunTag3 with trigram for Hungarian. In each train/test we measured the best score with HunTag3 and trigrams on IOBES labels. Despite the results for English, CRFsuite did not produce better F-score for Hungarian (5th and 6th row of Table 6). The number of inconsistent IOB sequences were 0 in all tests based on the MSD dataset (1st column of Table 6).

A 10-fold cross-validation was done to examine the

method \ train/test	MSD	KR	KR +Sect. 4.2.
TNT	68.52	70.95	-
prev. results by others	81.71	88.72/90.28	-
HunTag (baseline)	93.20	88.96	90.78
HunTag3, bigram	93.43	89.10	90.72
HunTag3, trigram	**93.59**	**89.83**	**91.50**
HunTag3, CRFsuite	92.27	89.12	89.77

Table 6: Hungarian results on Szeged Treebank, F-scores with various POS tags and test sets. The best results were reached with the help of new POS category suggestions, and explicit features against typical HunTag mistakes

validity of our best results on MSD tagset. It showed the following average F-scores: HunTag: 86.9±4.9%, HunTag3 with CRFsuite: 89.2±3.8%, HunTag3 with bigram: 90.1±3.7% and HunTag3 with trigram: 90.3±3.8%. These results have the same trend as Table 6, however, the differences are significant only between CRFsuite and HunTag3 with bigram (according to the t-test).

5.3. Hungarian NER

HunTag3 was evaluated on the Szeged NER corpus (Szarvas et al., 2006) as well (tokens: 226,000, NE: 14,500). We used previous HunTag results as a baseline (Simon, 2013), with the same development set, and 10-fold cross-validation on the test set, omitting the development set. All the aforementioned methods were tested. First, with *Simons's original featuresets*, which include *a reduced featureset* suggested by Simon that yields better results on the test set. Second, the featuresets were augmented with our features (Section 4.2.).

Our results show that on the development set, the best method is trigrams. With the same features applied to the test set, our features have no effect. However, on the cross-validated test set, the bigram solution has turned out to be the best. Significance test showed no difference between the updated methods, but compared to the baseline, there is a significant improvement. Our features have slight overall negative effect on performance. However, all methods have outperformed the original HunTag version (details can be seen in Table 7).

5.4. Discussion

Previously, HunTag was used as a flexible, open source sequential tagger for maximal NP chunking and NER achieving the state-of-the-art performance on Hungarian. We updated the codebase and fine-tuned the engine (HunTag3, bigram), updated to use trigrams in a TnT-like manner (HunTag3, Trigram), made it ready to use with CRFsuite (HunTag3, CRFsuite) and added new features in two steps. First, automatically generated ones from resources (WordNet and MetaMorpho), then grammar oriented ones from manual analysis of the errors of HunTag. The program is now capable of feature evaluation and is open to test different algorithms easily. We tested the aforementioned methods on multiple training/test set combinations, in some cases with 10-fold cross validation for the Hungarian NP chunking, NER tasks and English chunking. The latter is included to show the performance of our tool compared to CRFs, when applied to an other language.

In all tests for Hungarian, the CRF method was outperformed by HunTag3, however, for English CRF has turned out to be significantly better by 0.01%, but still cannot beat the state of the art. We can clearly conclude that the reimplemented HunTag3 has outperformed HunTag. For different tasks in Hungarian, different methods turned out to be the best: for NP chunking, the trigram solution augmented with our features reached the best score (3% improvement), however, for NER, the bigram version achieved the best results, and our features had no or negative effect.

Acknowledgments.
We would like to express our gratitude to Dr. Nóra Wenszky and Borbála Siklósi for their constructive proofreading during the writing of this article.

6. References

Baldridge, Jason, 2005. The OpenNLP project.

Bird, Steven, Ewan Klein, and Edward Loper, 2009. *Natural language processing with Python*. O'Reilly Media, Inc.

Brants, Thorsten, 2000. TnT: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*. Association for Computational Linguistics.

Brill, Eric, 1992. A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics.

Charniak, Eugene, 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*. Association for Computational Linguistics.

Csendes, Dóra, János Csirik, Tibor Gyimóthy, and András Kocsor, 2005. The Szeged Treebank. In *Lecture Notes in Computer Science: Text, Speech and Dialogue*. Springer.

Erjavec, Tomaž, 2010. Multext-east version 4: Multilingual morphosyntactic specifications, lexicons and corpora. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias (eds.), *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. Valletta, Malta: European Language Resources Association (ELRA).

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning, 2005. Incorporating non-local information into information extraction systems by Gibbs sampling.

features \ method		HunTag	HunTag3, CRFsuite	HunTag3, Bigram	HunTag3, Trigram
best features on the development set - *Simon's full featureset*	test set	89.37 ± 4.29	97.17 ± 1.58	97.70 ± 1.32	97.32 ± 1.48
	devel set	85.83	87.99	97.25	97.87
best features on the test set - *Simon's reduced featureset*	test set	89.40 ± 4.43	96.75 ± 1.67	**97.75 ± 1.30**	97.29 ± 1.51
	devel set	85.83	87.58	97.34	**98.05**
best features on the development set **+ sect. 4.2.** - *Our full featureset*	test set	-	96.57 ± 2.02	97.43 ± 1.50	97.06 ± 1.56
	devel set	-	93.22	97.24	96.98
best features on the test set **+ sect. 4.2.** - *Our reduced featureset*	test set	-	96.72 ± 1.87	97.45 ± 1.42	97.06 ± 1.53
	devel set	-	81.37	97.34	**98.05**

Table 7: NER results of HunTag3 in Hungarian with 10-fold cross-validation tests

In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Jones, Eric, Travis Oliphant, Pearu Peterson, et al., 2001–. SciPy: Open source scientific tools for Python.

Kornai, András, Péter Rebrus, Péter Vajda, Péter Halácsy, András Rung, and Viktor Trón, 2004. Általános célú morfológiai elemző kimeneti formalizmusa (The output formalism of a general-purpose morphological analyzer). In *Proceedings of the 2nd Hungarian Computational Linguistics Conference*.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira, 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

McCallum, Andrew, Dayne Freitag, and Fernando CN Pereira, 2000. Maximum entropy Markov models for information extraction and segmentation. In *ICML*, volume 17.

Miháltz, Márton, 2011. Magyar NP-felismerők összehasonlítása (Comparing Hungarian NP-chunkers).

Miháltz, Márton, Csaba Hatvani, Judit Kuti, György Szarvas, János Csirik, Gábor Prószéky, and Tamás Váradi, 2008. Methods and results of the Hungarian WordNet project. In *Proceedings of the Fourth Global WordNet Conference. GWC*.

Miller, George A, 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Molina, Antonio and Ferran Pla, 2002. Shallow parsing using specialized HMMs. *The Journal of Machine Learning Research*, 2:595–613.

Novák, Attila, László Tihanyi, and Gábor Prószéky, 2008. The MetaMorpho translation system. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08. Stroudsburg, PA, USA: Association for Computational Linguistics.

Okazaki, Naoaki, 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs).

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., 2011. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830.

Recski, Gábor, 2014. Hungarian noun phrase extraction using rule-based and hybrid methods. *Acta Cybernetica*, 21(3):461–479.

Recski, Gábor and Dániel Varga, 2009. A Hungarian NP Chunker. *The Odd Yearbook. ELTE SEAS Undergraduate Papers in Linguistics*:87–93.

Recski, Gábor and Dániel Varga, 2012. Magyar főnévi csoportok azonosítása (Identifying Hungarian noun phrases). *Általános Nyelvészeti Tanulmányok*:81–95.

Roth, Luzia and Simon Clematide, 2014. *Tagging Complex Non-Verbal German Chunks with Conditional Random Fields*. Universitätsbibliothek Hildesheim.

Shen, Hong and Anoop Sarkar, 2005. Voting between multiple data representations for text chunking. In Balázs Kégl and Guy Lapalme (eds.), *Advances in Artificial Intelligence, 18th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2005, Victoria, Canada, May 9-11, 2005, Proceedings*, volume 3501 of *Lecture Notes in Computer Science*. Springer.

Simon, Eszter, 2013. *Approaches to Hungarian Named Entity Recognition*. Ph.D. thesis, Budapest University of Technology and Economics Budapest.

Szarvas, György, Richárd Farkas, László Felföldi, András Kocsor, and János Csirik, 2006. A highly accurate named entity corpus for Hungarian. In *Proceedings of International Conference on Language Resources and Evaluation*.

Tjong Kim Sang, Erik F. and Sabine Buchholz, 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, CoNLL '00. Stroudsburg, PA, USA: Association for Computational Linguistics.

Tjong Kim Sang, Erik F and Fien De Meulder, 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics.

Van Der Walt, Stefan, S Chris Colbert, and Gaël Varoquaux, 2011. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.