Acta Cybernetica **21** (2014) 461–479.

Hungarian Noun Phrase Extraction Using Rule-based and Hybrid Methods*

Gábor Recski†

Abstract

We implement and revise Kornai's grammar of Hungarian NPs [11] to create a parser that identifies noun phrases in Hungarian text. After making several practical amendments to our morphological annotation system of choice, we proceed to formulate rules to account for some specific phenomena of the Hungarian language not covered by the original rule system. Although the performance of the final parser is still inferior to state-of-the-art machine learning methods, we use its output successfully to improve the performance of one such system.

Keywords: natural language processing, parsing, machine learning

1 Introduction

This paper describes a rule-based system which extracts noun phrases (NPs) from morphologically analyzed Hungarian text. We implement and revise the grammar of Hungarian NPs in [11] to create a system that identifies NPs by means of bottom-up parsing. Although high performance on the standard task is already possible using state-of-the-art machine learning methods, we show that a rule-based approach contributes substantially to the performance of a hybrid system. Section 2 describes the task and provides a brief survey of the standard statistical approach. Section 3 documents the process of creating a Hungarian NP corpus, a resource crucial not only for machine learning approaches, but also for the evaluation of rule-based systems.

Section 4.1 describes the technical preliminaries of creating an NP parser. In section 4.2 we describe the process of grammar development, which involves examining the error classes in the output after every major change to the rule system. Section 5 proposes a simple hybrid system where the chunking task is performed by the learning-based system `hunchunk` [23], [24] using features derived from the output of the rule-based system.

*I would like to thank András Kornai and two anonymous reviewers for their many useful comments and suggestions. Work supported by OTKA grant #82333.

†Department of Theoretical Linguistics, Eötvös Loránd University and Hungarian Academy of Sciences, E-mail: `recski@budling.hu`

DOI: 10.14232/actacyb.21.3.2014.11

2 Chunking

2.1 The task

The task of extracting one or several types of phrases from a text is often referred to as *shallow parsing* or *chunking.* The term *chunk* and the task of text chunking, however, do not have universally accepted definitions in NLP (Natural Language Processing) literature. The term chunk was first used by Abney in [2], who uses it to describe non-overlapping units of a sentence that each consist of "a single content word surrounded by a constellation of function words" . Based primarily on [9], who introduce the term *performance structure* to describe psycholinguistic units of a sentence, Abney argues that chunks are units that do not neccessarily coincide with syntactic constituents. Recent works on the automated chunking of raw text, however, invariably use definitions of chunks that make it possible to extract them from parse trees in order to provide training data for supervised learning systems. In practice, these chunks usually coincide with some group of syntactic phrases. One complete set of definitions for various classes of chunks is given in the description of the chunking task of CoNLL 2000 [28], where the Penn Treebank [17] was used as a source of chunk data.

One of the best known works on the extraction of NP chunks is that of Ramshaw and Marcus [18], who define *base NP*s (or *non-recursive NP*s) as noun phrases that do not contain another noun phrase. It is this definition that was adopted by Tjong Kim Sang and Buchholz for the CoNLL 2000 shared task, and when the task of NP chunking is mentioned as a benchmark for some machine learning algorithm, it almost invariably refers to base NP tagging based on the datasets proposed by Ramshaw and Marcus and adopted by CoNLL-2000.

2.2 Overview of statistical methods

Besides defining the task of NP chunking as the identification of non-recursive (base) noun phrases, Ramshaw and Marcus attempt to solve the task by applying the method of transformation-based learning, which had been used before for the tasks of part-of-speech tagging [4] and parsing [5]. Using the datasets and method of evaluation that was later to become the CoNLL shared task and also the standard field of comparison for NP-chunker tools, Ramshaw and Marcus report precision and recall rates of 90.5% and 90.7% respectively. Their datasets used for training and testing purposes were derived from sections 15-18 and section 20 of the Wall Street Journal respectively, data which was available from the Penn Treebank.

During and after the CoNLL shared task in 2000, a wide variety of machine learning methods have been applied to the task of identifying base NPs. Kudo and Matsumoto reached an F-score of 93.79% by using Support Vector Machines [13], a result that was to increase to 94.22% a year later when they introduced weighted voting between SVMs trained using different chunk representations [14]. Probably the most popular method for NP chunking today is the Conditional Random Field (CRF, [15]) machine learning algorithm. CRFs have been used on the standard

CoNLL task by Sha and Pereira, who achieved an F-score of 94.3% [26], and more recently by Sun et al. ($F = 94.34\%$) [27].

A further notable result is that of Hollingshead and colleagues [10], who evaluated several context-free parsers on various shallow parsing tasks and report an F-score of 94.21% on the CoNLL task using the Charniak parser [6]. These results show that a rule-based system can be competitive with results obtained by using any advanced machine learning algorithm, a fact that clearly points us in the direction of hybrid systems.

2.3 The `hunchunk` system

In the final section of this paper we shall combine the parser with our own learning-based NP-chunking tool. `Hunchunk` uses a combination of Maximum Entropy learning and Hidden Markov Models (HMM) to perform NP-chunking of a sentence that is tokenized and morphologically annotated. For a detailed description of `hunchunk` the reader is referred to [23]. Some past applications of `hunchunk` are documented in [25] and [22]. The tool is available for download under an LGPL license from `http://www.github.com/recski/HunTag`.

3 Creating NP corpora

A preliminary step of creating the NP corpus is choosing a method for representing morphological information. The morphological analyzer `hunmorph` [29] uses the `KR` formalism [20] and our grammar relies heavily on the kind of structured information that `hunmorph` provides and `KR` codes represent.

3.1 The KR formalism

The `KR` formalism for representing morphological information was developed with the intention of capturing the hierarchy between individual inflectional features and encoding the derivational steps used to arrive at the word form in question. The output of the analysis of a word starts with the stem and contains the category and features of the word as well as the category of the word from which the given form was derived, if any. This latter part of the code also contains in square brackets the type of derivation used to form the final word. The last part of the code represents the hierarchy between grammatical features of the word by means of bracketing similar to that used for the analysis of sentence structure.

Some examples of `KR`-codes in the Szeged Treebank [7] are given in Table 1. As can be seen, `KR` encodes the entire chain of derivations that led to the word form under analysis.

One great advantage of this formalism is that it explicitly encodes all pieces of information which one might think of as a grammatical feature, therefore any NLP application which relies on word level information can make use of the `KR` code

Table 1: KR examples

tanárunk
teacher-Poss1Pl
'our teacher'
`tanár/NOUN<POSS<1><PLUR>>`

óráján
class-Poss3-SUP
'in his/her class'
`óra/NOUN<POSS><CAS<SUE>>`

másodikkal
two-ORD-INS
'with the second'
`kettö/NUM[ORD]/NUM<CAS<INS>>`

vegyük
take-Imp-Pl1-Def
'let's take'
`vesz/VERB<SUBJUNC-IMP><PERS<1>><PLUR><DEF>`

felértékelődése
up-value(V)-Med-Ger-Poss3
'the increase of its value'
`felértékel/VERB[MEDIAL]/VERB[GERUND]/NOUN<POSS>`

without the need for any external knowledge about the meaning of various symbols or positions in the code.

The `KR` formalism straightforwardly encodes most grammatical features, but there are still some distinctions which it is unable to represent. One of these, which we must overcome in order to account for syntactic phenomena, is the distinction between pronouns and nouns as well as the various types of pronouns in Hungarian. Pronouns are tagged as nouns in the `KR` formalism because they take part in the same inflectional phenomena as nouns – although some of their paradigms are defective –, therefore introducing a new top-level category into the `KR` system would cause the loss of a well-founded generalization. The solution we implemented for use with our system is the introduction of the noun feature `PRON` which takes as its value `0` if the word is not a pronoun and the type of pronoun otherwise. This addition results in the analyses exemplified in Table 2, for a detailed description see [21].

Table 2: Pronoun types

ez this `ez/NOUN<PRON<DEM>>`
mindenki everybody `mindenki/NOUN<PRON<GEN>>`
valami something `valami/NOUN<PRON<INDEF>>`
aki who (relative pron.) `aki/NOUN<PRON<REL>>`
ki who (interrogative pron.) `aki/NOUN<PRON<WH>>`
saját own `aki/NOUN<PRON<POS>>`

3.2 Extracting NPs from a treebank

Having determined the way we wish to encode morphological information we may proceed to create an NP corpus by extracting sentences and syntactic information from a treebank (a corpus which contains the full syntactic analysis for all sentences, cf. [1]). For this purpose we use the Szeged Treebank [7], a syntactically annotated corpus of Hungarian which contains nearly 1.5 M tokens of text taken from a variety of genres including fiction, newspapers, legal text, software documentation and essays written by students between the age of 13 and 16.

The treebank contains morphological information about each word in the `MSD` format [8]. Converting `MSD`-tags to `KR` is insufficient because `MSD` codes do not contain data about the derivations that create a word form, a piece of information which `KR` can encode and which some of our rules rely on. Our morphological analyzer, **`hunmorph`**, is able to supply this information, but it will necessarily produce some sporadic tagging errors on sentences extracted from the Treebank. Such errors may be corrected in a machine learning system based on context, but will surely mislead the rule-based system, which has no other source of information at its disposal. In order to have all available data present in the corpus, and at the same time preserve the high precision provided by manually annotated tags, we merged our two sources of data. Information on the derivation of a word form, if any, was taken from the `KR`-codes provided by `hunmorph`, the remaining part of the tag, containing the category of the word as well as its grammatical features, was obtained from the Treebank. In case the Treebank could not provide any grammatical information (0.91% of all words), the output of `hunmorph` was entered into the corpus as is.

3.3 Mending the corpus

Having created a base NP corpus by the method described in section 3.2, we proceeded to apply two further changes to the data in order to handle syntactic analyses in the Treebank with which we do not agree. Since we intend to use these corpora as a standard of evaluation for the parser, we need it to reflect the analyses which we expect our system to produce. In this paper we do not wish to argue extensively for one analysis over the other, we simply describe the changes we have made to the data in order to ensure that our experiments can be replicated.

3.3.1 Adjectives in possessive constructions

The largest number of cases where there is a discrepancy between the Szeged analysis and the one used here is related to the analysis of possessive constructions. The noun phrase in Table 3 is represented in the treebank as in Figure 1.

Table 3: Possessive construction

egy	*idős*	*úr*	*kopasz*	*fejére*
an	elderly	gentleman	bald	head-Poss3-SUBL
‘on the bald head of an elderly gentleman’				

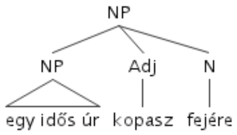

Figure 1: Original analysis of the possessive construction

We believe this analysis to be false since the noun and preceding adjective modifying it form a constituent in Hungarian and the possessive construction does not change this fact: the possessor NP can be followed directly by any NP with the `POSS` feature. Therefore we modified our base NP corpus in order to reflect the analysis in Figure 2, which we believe to be the correct one. We will expect our system to parse such structures as two consecutive NPs.

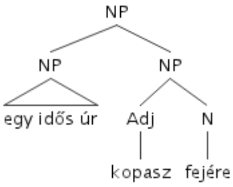

Figure 2: Revised analysis of the possessive construction

3.3.2 Demonstratives

Another structure which we intend to treat differently from the analysis in the Treebank is the special demonstrative construction of Hungarian exemplified in Table 4. Note that in this structure the demonstrative pronoun *ez/az* must be marked for both the case and number of the following noun.

Table 4: Demonstrative NP

ez	*a*	*pincér*
this	the	waiter
'this waiter'		
ezek	*a*	*hajók*
this-PL	the	ship-PL
'these ships'		
attól	*a*	*pasastól*
that-ABL	the	bloke-ABL
'from that bloke'		

For these structures the Treebank gives the analysis in Figure 3. We believe that the demonstrative pronoun cannot project a noun phrase of its own, therefore we change the corpus to reflect the analysis in Figure 4.

3.3.3 Other issues

The chunk corpus extracted from the Szeged Treebank still present a number of small anomalies that hinder the evaluation of both the rule-based and the statistical system as well as the training of the latter. One notable example is a construction which involves an NP containing an adjective that precedes the noun and is enclosed

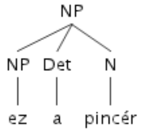

Figure 3: Original analysis of demonstrative NPs

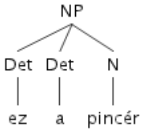

Figure 4: Revised analysis of demonstrative NPs

in parentheses and which occurs often in legal text (e.g. *A Gt. (új) 3. paragrafusa* 'The (new) 3rd section of the Gt. Act'). This case falls under the same questionable analysis as those described in section 3.3.1. We believe that arbitrary modification of the analysis of problematic structures (which are, unfortunately, overrepresented in our corpus) is not a measure we can take in good conscience. Therefore, we leave these occurrences, as well as any smaller anomalies, untouched. We note that this phenomenon accounts for ca. 5% of those base NPs which our grammar is unable to parse.

3.4 Evaluation methods

The corpus created in the manner described above is used to evaluate our parser at various stages of development. The statistical system `hunchunk` also uses two (non-overlapping) sections of this corpus for training and testing. Finally, performance of the hybrid system (to be introduced in the final section of this paper) is also measured using this data as gold standard.

In each of these cases, evaluation involves comparing two sets of chunks for each sentence, the one supplied by the system in question and the one present in the corpus. Our evaluation method follows the guidelines of CoNLL-2000: a chunk identified by our system is considered correct iff it corresponds to a chunk in the gold standard and a chunk in the corpus is considered found iff it corresponds to a chunk in our tagging. A system's performance can be described by two values: the *precision* of a system is the number of correctly identified chunks divided by the number of all chunks in the output, while the *recall* rate is obtained by dividing the same number by the number of chunks in the gold standard. As customary, we measure the overall quality of the tagging by calculating the harmonic mean of these two values, also called the F-score:

$$F = \frac{2PR}{P+R}$$

where P and R stand for precision and recall respectively (cf. e.g. [16].

4 Rule-based method

This chapter describes our efforts to use a rule-based parser for the extraction of noun phrases. We improve the context-free feature grammar of Hungarian NPs [11] [12] in order to account for even the most complicated structures.

4.1 Building a parser

Our system uses the NLTK parser, a tool which supports context-free grammars and a wide variety of parsing methods [3]. To parse a text we must first give a feature representation of all words. We implement the context-free grammar of Kornai to create a parser which takes as its input the series of `KR`-codes of words in a sentence and produces, by means of bottom-up parsing, charts containing the possible rule applications that may produce some fragment of the sentence. A chunking is then derived from this chart through a series of recognition steps which we shall describe at the end of this section.

4.1.1 Preparing the data

When using the NLTK parser with a CF grammar, the system accepts nonterminal symbols that consist of a category symbol such as `NOUN` or `VERB` followed by a set of features in square brackets. Feature values can be strings, integers, non-terminals of the grammar and variables that bind the value of the feature to that of some other feature of the same type in the rule. Thus a rule to encode agreement in number between verb and object would be `VP -> V[PL=?a] N[PL=?a]`, which is equivalent to the more standard 'Greek variable' notation `VP -> V[`α`PL] N[`α`PL]`. Converting `KR` codes to such representations, i.e. supplying the terminal rules for our grammar, is a straightforward mechanical process. Some examples are given in Table 5. Notice that the grammar does not use different symbols for various projection levels of the same syntactic category, but encodes this information in the feature `BAR`; the notation `NOUN[BAR=0]` will then simply represent a bare noun. Information on the source of derivation is represented by the feature `SRC` which takes as its value a set of two features: `STEM` encoding the features of the source word and `DERIV` the type of derivation.

As we have described in section 3.1, the bulk of any `KR`-style code lends itself to such a representation, e.g. the code `NOUN<POSS><PLUR>` needs only to be rewritten as `NOUN[POSS=1, PLUR=1]` in order to produce input for `NLTK`. Still, a number of problems must be addressed when transforming `KR` codes into such feature structures. First of all, `KR` features are privative: the fact that a noun is singular, for example, can be concluded from the absence of the `<PLUR>` feature. Similarly, the default case is nominative (there is no `<CAS<NOM>>` feature), the default person is the third, etc. Since our grammar should be able to refer to such default features

Table 5: Terminal rules

```
NOUN[POSS=[1=1, PLUR=1] -> NOUN<POSS<1><PLUR>>

NOUN[POSS=1, CAS=[SUE=1]] -> NOUN<POSS><CAS<SUE>>

NOUN[ANP=0, CAS=0, PLUR=0, POSS=[1, PLUR=1], PRON=0] ->
-> 'NOUN<POSS<1><PLUR>>

NUM[CAS=[INS=1], SRC=[STEM=NUM, DERIV=ORD]] ->
-> NUM[ORD]/NUM<CAS<INS>>

VERB[SUBJUNC-IMP=1, PERS=[1=1], PL=1, D=1] ->
-> VERB<SUBJUNC-IMP><PERS<1>><PLUR><DEF>

NOUN[POSS=1, SRC=[STEM=VERB[SRC=[STEM=VERB, DERIV=MEDIAL]],
DERIV=GERUND]] ->
-> VERB[MEDIAL]/VERB[GERUND]/NOUN<POSS>
```

in a straightforward manner, the process of transforming KR-codes involves explicating these features by adding the feature values PERS=0, CAS=0, PLUR=0, etc. Similarly, a word which has not been identified as the product of some derivation will receive the feature SRC=0.

4.1.2 Implementing NP-chunking

Having established a method for creating the terminal rules of our grammar we are now able to parse, based on the NP-grammar of Kornai, any sentence tagged according to the KR formalism. Since we do not have a complete grammar of Hungarian, we employed a bottom-up parser, which can provide an analysis of fragments of a sentence without parsing the full sentence. The output obtained for each sentence is a chart which contains *edges*, individual entries which describe a step in the parsing process by representing a particular application of a rule in the grammar, and gives the location of the sentence fragment to which it can be applied.

The absence of an S-grammar means that we cannot automatically discard the majority of chart edges based on their lack of ability to function as part of a parse-tree for the full sentence. Therefore we must compile a list of rules to post-process the set of parse edges in order to produce non-overlapping NP sequences. First, we take all fragments of the sentence which correspond to a *complete* NOUN edge, thereby selecting the word sequences that the parser considers potential NPs of the sentence. Secondly, since we are trying to extract base NPs only, we discard all fragments which contain more than one noun. Next, we discard all fragments which are contained in a larger fragment. The final and most complicated step in finding NPs is dealing with overlapping fragments: we implement a heuristic approach in which we choose of two overlapping NPs the one which cannot be parsed as a phrase of some other category based on the parse chart. This process is preferable since most overlaps are produced by SLASH-rules, i.e. rules which allow NPs with elliptic heads to be parsed as NPs. In most cases, these rules falsely generate phrases which are not NPs but AdjPs, NumPs, etc. In case this process fails to select exactly one of the two fragments – i.e. both or neither of them can be parsed as a phrase of some other category – we discard them both.

4.2 Developing the grammar

In this section we describe our additions to the grammar of Hungarian NPs published in [11]. We evaluate each version of the grammar on a test corpus which contains 1000 sentences picked randomly from all genres in the base NP corpus, following the principles described in section 3.4.

Implementing the initial grammar of Kornai our system achieves an F-score of 81.76%. By observing the output it is clear that the greatest shortcoming of our system is its lack of knowledge about the internal structure of adjectival, numeral and adverbial phrases, all of which can form components of an NP. Therefore our first step does not involve touching the NP grammar but rather the addition of some simple rules to account for complex AdjPs, NumPs and AdvPs. These rules can be seen in Table 6.

Table 6: Basic rules for AdjPs and NumPs

ADJ -> ADJ ADJ
ADJ -> ADV ADJ
NUM -> NUM NUM
NUM -> ADV NUM
NUM -> ADJ NUM

After the addition of these rules our system produces chunkings with an F-score of 84.18%. The next step involved the treatment of pronouns. We have discussed in section 3.1 that Hungarian pronouns behave very similarly to nouns, and in fact the parser can only distinguish them from nouns with the help of a feature which we have added to the KR-system. In the vast majority of cases, treating pronouns as nouns is entirely justified. There are, however, a handful of phenomena which make it necessary for us to refer to them separately in the grammar. General pronouns (e.g. *minden* 'all') and indefinite pronouns (e.g. *néhány* 'some') may combine with a following noun constituent to form an NP (cf. Table 7)

These pronouns are not in complementary distribution with numerals, however we choose to keep the grammar simple and adjoin them to nouns of bar-level 1. The resulting rules are shown in Table 8.

Table 7: General and indefinite pronouns

minden	*pofon*
all	punch
‘all punches’	
néhány	*villanykörte*
some	light-bulb
‘some light-bulbs’	

Table 8: Rules for general and indefinite pronouns

```
NOUN[POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f] ->
-> NOUN[PRON=GEN]
NOUN[BAR=1, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]

NOUN[POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f] ->
-> NOUN[PRON=INDEF]
NOUN[BAR=1, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]
```

The addition of these two rules result in an increase of the system’s F-score to 85.45. A third type of pronoun, the demonstrative *ez/az*, etc. also needs treatment when it comes to the demonstrative structure described in section 3.3.2. To allow the parser to recognize the structure we implement the rule in Table 9, thus achieving an F-score of 86.68.

Table 9: Rule for demonstrative NPs

```
NOUN[POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e] ->
-> NOUN[PRON=DEM, BAR=0, POSS=?a, PLUR=?b, ANP=?c, CAS=?d]
ART NOUN[PRON=0, BAR=2, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=0],
```

The next structure which caused numerous parsing errors is that of adjectival phrases containing a noun followed by an adjective derived from a verb (called a *deverbal adjective*), either in perfect or imperfect participle form. An example of both of these structures can be seen in Table 10.

Since our terminal symbols encode information about the source of derivation which produced any given word form, we can once again treat these structures properly by adding the two rules in Table 11 to our grammar. This addition caused an increase in the performance of the system to 87.87%. In the end the greatest

Table 10: Sentences with deverbal adjectives

a	*korsónak*	*támasztott*
the	jug-DAT	prop-PERF_PART
könyvet	*olvasta*	
book-ACC	read-PAST-DEF-3	
‘He read the book propped up against the jug.’		
az	*ókori*	*mór*
the	ancient	moor
hódítóktól	*származó*	*esküvést*
conqueror-Pl-FROM	originate-IMPERF_PART	oath-ACC
hallották		
hear-PAST-DEF-3		
‘They heard the oath originating from ancient moor conquerors’		

Table 11: Rules for deverbal adjectives

```
ADJ -> NOUN ADJ[SRC=[STEM=VERB[], DERIV='PERF_PART']]
ADJ -> NOUN ADJ[SRC=[STEM=VERB[], DERIV='IMPERF_PART']]
```

error classes – besides those caused by genuinely ambiguous structures – remained those which involved the incorrect parsing of punctuation marks and conjunctions. With the addition of several rules describing their behaviour in and around NPs (see Appendix A) we further increased the F-score of the system to 89.36%.

The progress of the system’s performance as a result of our steps of grammar development are summarized in Table 12. As can be seen from these figures our development of the grammar corrected nearly half of the errors made by the system. For the final version of the grammar see Appendix A.

Table 12: Stages of grammar development

Development stage	F-score
Kornai 1985	81.76%
AdjPs, AdvPs, NumPs	84.18%
Pronouns	85.45%
Demonstrative NPs	86.68 %
Deverbal adjectives	87.87%
Punctuation and conjunctions	89.36%

5 Features from the parser

Although the performance of our parser is still inferior to statistical systems, in this final section we will demonstrate, using a very simple example, how a machine learning system may benefit from the output provided by the parser.

`Hunchunk` handles the task of chunking as a type of word-tagging and attempts to assign the correct chunk-tag to each word in the sentence: the five tags `B-NP, I-NP, E-NP, 1-NP, O` indicate the position of a word within a chunk and each possible chunking of a sentence corresponds to a sequence of word tags. The system uses Maximum Entropy learning [19] to determine for each word the probability distribution over this tagset, based on a set of binary features of the word such as character ngrams, morphological features, position in the sentence, etc. (see [23] for details). Using these distributions as observation probabilities and a simple bigram model as an estimate for transition probabilities, the Viterbi algorithm can efficiently compute the most probable sequence of tags, i.e. the most probable chunking for a sentence.

We improve the system by first converting the output of the NP-parser to such a sequence of tags and then using the tag for each word as an extra feature that the maxent model has access to. In other words, when trying to guess what the chunk-tag of a word should be, the `hunchunk` system may use the answer the NP-parser gives to the same question. In order to evaluate this hybrid system we parse the entire chunk corpus and then create a train and test set from the data obtained in the same way as we would do when evaluating `hunchunk` on its own. Table 13 shows the results of the evaluation for both the original `hunchunk` model and the new hybrid system.

	Precision	Recall	F-score
`hunchunk`	94.61%	94.88%	94.75%
`hunchunk`+parser features	95.29%	95.68%	95.48%

Table 13: The role of parser features in base NP chunking

As can be seen from the above figures, the addition of information from a rule-based system leads to a 15% decrease in the number of errors made by the statistical system. We also measured the impact of parser features on a different chunking task which `hunchunk` performs: that of extracting `maximal NPs`, i.e. noun phrases that are not contained by a higher level NP. In the case of maximal noun phrases the parser feature also causes some increase in performance (cf. Table 14).

	Precision	Recall	F-score
`hunchunk`	89.34%	88.12%	88.72%
`hunchunk`+parser features	89.46%	88.76%	89.11%

Table 14: The role of parser features in maxNP-chunking

6 Conclusion

This paper described the process of implementing a grammar for Hungarian noun phrases to create an NP-parser and using its output to enhance the performance of a state-of-the-art statistical system. Firstly, we described the technical preliminaries of implementing a context-free grammar and also documented additions and amendments made to both the data and the grammar. Having reached a sufficient parsing quality we proceeded to use the output of the rule-based system to create new features for use with the learning-based `hunchunk` system.

The improved F-scores indicate that hybrid systems in NP-extraction may produce results superior to those of a stand-alone machine learning system. However, it falls beyond the scope of this paper to explore the various possibilities of combining rule-based and statistical approaches to NP-chunking. Also, cross-analysis of errors made by each system – possibly on larger corpora – could help us gain a better understanding of what the strengths and weaknesses of the two approaches are.

References

[1] Abeillé, A., editor. *Treebanks: Building and using parsed corpora.* Kluiwer, Dodrecht, 2003.

[2] Abney, S. Parsing by chunks. In Berwick, Robert, Abney, Steven, and Tenny, Carol, editors, *Principle-based parsing*, pages 257–278. Kluwer Academic Publishers, 1991.

[3] Bird, S., Klein, E., and Loper, E. *Natural language processing with Python.* O'Reilly Media, 2009.

[4] Brill, E. A simple rule-based part of speech tagger. In *Third Conference on Applied Natural Language processing*, pages 152–155, Trento, Italy, 1992.

[5] Brill, E. Automatic grammar induction and parsing free text: A transformation-based approach, 1993. *Proceedings of the Workshop on Human Language Technology*, pages 237–242, Association for Computational Linguistics, Stroudsburg, PA, USA, 1993.

[6] Charniak, E. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference (NAACL-2000)*, pages 132–139. Association for Computational Linguistics, 2000.

[7] Csendes, D., Csirik, J., Gyimóthy, T., and Kocsor, A. The Szeged Treebank. In *Lecture Notes in Computer Science: Text, Speech and Dialogue*, pages 123–131. Springer, 2005.

[8] Erjavec, T. MULTEXT-east version 3: Multilingual morphosyntactic specifications, lexicons and corpora. In Lino, Maria Teresa, Xavier, Maria Francisca, Ferreira, Fátima, Costa, Rute, and Sila, Raquel, editors, *Fourth Inernational Conference on Language Resources and Evaluation (LREC 2004)*, pages 1535–1538, Paris, 2004. European Language Resources Association (ELRA).

[9] Gee, J. P. and Grosjean, F. Performance structures: A psycholinguistic and linguistic apprasial. *Cognitive Psychology*, 15:411–458, 1983.

[10] Hollingshead, K., Fisher, S., and Roark, B. Comparing and combining finite-state and context-free parsers. In *Proceedings of the Conference on Human Language Techonologies and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 787–794, 2005.

[11] Kornai, A. The internal structure of Noun Phrases. In *Approaches to Hungarian*, 1:79–92, 1985.

[12] Kornai, A. A főnévi csoport egyeztetése. In Kiefer, Ferenc, editor, *Általános Nyelvészeti Tanulmányok*, volume 17, pages 183–211. Akadémiai Kiadó, 1989.

[13] Kudo, T. and Matsumoto, Y. Use of support vector learning for chunk identification. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning - Volume 7*, page 144. Association for Computational Linguistics, 2000.

[14] Kudo, T. and Matsumoto, Y. Chunking with support vector machines. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*, pages 1–8. Association for Computational Linguistics, 2001.

[15] Lafferty, J., McCallum, A., and Pereira, F. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.* Morgan Kaufmann, 2001.

[16] Makhoul, J., Kubala, F., Schwartz, R., and Weischedel, R. Performance measures for information extraction. In *Proceedings of the DARPA Broadcast News Workshop*, page 249, Herndon, VA, 1999. Morgan Kaufmann.

[17] Marcus, M., Santorini, B., and Marcinkiewicz, M. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330, 1993.

[18] Ramshaw, L. and Marcus, M. Text chunking using transformation-based learning. In Yarowsky, D. and Church, K., editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Cambridge, MA, 1995. SIGDAT/ACL.

[19] Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution.* PhD thesis, University of Pennsylvania, 1998.

[20] Rebrus, P., Kornai, A., and Varga, D. Egy általános célú morfológiai annotáció. [A general purpose morphological annotation system] *Általános Nyelvészeti Tanulmányok* , volume 24, pages 47-80. Akadémiai Kiadó 2012.

[21] Recski, G. Egy általános célú morfológiai annotáció kiterjesztése [Extending a general-purpose morphological annotation system]. In Váradi, T., editor, *VII. Alkalmazott Nyelvészeti Doktoranduszkonferencia.* pages 168-174. Budapest, MTA Nyelvtudományi Intézet, 2013.

[22] Recski, G., Rung, A., Zséder, A., and Kornai, A. Np alignment in bilingual corpora. In Calzolari, Nicoletta, Choukri, Khalid, Maegaard, Bente, Mariani, Joseph, Odijk, Jan, Piperidis, Stelios, Rosner, Mike, and Tapias, Daniel, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, 2010. European Language Resources Association (ELRA).

[23] Recski, G. and Varga, D. A Hungarian NP Chunker. *The Odd Yearbook. ELTE SEAS Undergraduate Papers in Linguistics*, pages 87–93, 2009.

[24] Recski, G. and Varga, D. Magyar főnévi csoportok azonosítása. *Általános Nyelvészeti Tanulmányok*, volume 24, pages 81-95. Akadémiai Kiadó 2012.

[25] Recski, G., Varga, D., Zséder, A., and Kornai, A. Főnévi csoportok azonosítása magyar-angol párhuzamos korpuszban [Identifying noun phrases in a parallel corpus of English and Hungarian]. *VI. Magyar Számitógépes Nyelvészeti Konferencia [6th Hungarian Conference on Computational Linguistics]*, 2009.

[26] Sha, F. and Pereira, F. Shallow parsing with conditional random fields. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 134–141. Association for Computational Linguistics, 2003.

[27] Sun, X., Morency, L.-P., Okanohara, D., and Tsujii, J. Modeling latent-dynamic in shallow parsing: a latent conditional model with improved inference. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 841–848. Association for Computational Linguistics, 2008.

[28] Tjong Kim Sang, E. F. and Buchholz, S. 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*, pages 127–132. Association for Computational Linguistics, 2000.

[29] Trón, V., Gyepesi, Gy., Halácsy, P., Kornai, A., Németh, L., and Varga, D. Hunmorph: open source word analysis. In Jansche, Martin, editor, *Proceedings of the ACL 2005 Software Workshop*, pages 77–85. ACL, Ann Arbor, 2005.

A Final grammar of the NP parser

```
NOUN[POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f] ->
  NOUN[PRON=POS]
  NOUN[BAR=2, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]
NOUN[POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e] ->
  NOUN[PRON=DEM, BAR=0, POSS=?a, PLUR=?b, ANP=?c, CAS=?d]
  ART NOUN[PRON=0, BAR=2, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=0]
NOUN[POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f] ->
  NOUN[PRON=GEN]
  NOUN[BAR=2, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]
NOUN[POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f] ->
  NOUN[PRON=INDEF]
  NOUN[BAR=2, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]
NOUN[BAR=1, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f] ->
  ADJ
  NOUN[BAR=0, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]
NOUN[BAR=1, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f] ->
  NOUN[BAR=0, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]
NOUN[BAR=1, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]/NOUN[BAR=0] ->
  ADJ
  NOUN[BAR=0, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]/NOUN[BAR=0]
NOUN[BAR=1, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]/NOUN[BAR=0] ->
  NOUN[BAR=0, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]/NOUN[BAR=0]
NOUN[BAR=2, POSS=?a, PLUR=0, ANP=?c, CAS=?d, D=?e, PRON=?f] ->
  NUM
  NOUN[BAR=1, POSS=?a, PLUR=0, ANP=?c, CAS=?d, D=?e, PRON=?f]
NOUN[BAR=2, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f] ->
  NOUN[BAR=1, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]
NOUN[BAR=2, POSS=?a, PLUR=0, ANP=?c, CAS=?d, D=?e, PRON=?f]/NOUN[BAR=0] ->
  NUM
  NOUN[BAR=1, POSS=?a, PLUR=0, ANP=?c, CAS=?d, D=?e, PRON=?f]/NOUN[BAR=0]
NOUN[BAR=2, POSS=?b, PLUR=0, ANP=?c, CAS=?d, D=?e, PRON=?f]/NOUN[BAR=0] ->
  NOUN[BAR=1, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e]/NOUN[BAR=0, PRON=?f]
NOUN[BAR=3, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f] ->
  ART[D=?e]
  NOUN[BAR=2, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, PRON=?f]
NOUN[BAR=3, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=1, PRON=?f] ->
  NOUN[BAR=0, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=1, PRON=?f]
NOUN[BAR=3, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]/NOUN[BAR=0] ->
  ART[D=?e]
  NOUN[BAR=2, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, PRON=?f]/NOUN[BAR=0]
NOUN[BAR=3, POSS=0, PLUR=?a, ANP=?b, CAS=?c, D=1, PRON=?f] ->
  NOUN[BAR=3, ANP=0, CAS=0]
  NOUN[BAR=2, POSS=1, PLUR=?a, ANP=?b, CAS=?c, PRON=?f]
NOUN[BAR=4, POSS=0, PLUR=?a, ANP=?b, CAS=?c, D=1, PRON=?f] ->
  NOUN[BAR=3, CAS=[DAT=1]]
  NOUN[BAR=3, POSS=1, PLUR=?a, ANP=?b, CAS=?c, D=1, PRON=?f]
NOUN[BAR=3, POSS=0, PLUR=?a, ANP=?b, CAS=?c, D=1, PRON=?f] ->
```

```
  ART[BAR=1, D=1, ME=?d, YOU=?e, PLUR=?f]
  NOUN[BAR=2, POSS=[ME=?d,YOU=?e,PLUR=?f], PLUR=?a,ANP=?b,CAS=?c,PRON=?f]
NOUN[BAR=3, POSS=0, PLUR=?a, ANP=?b, CAS=?c, D=1, PRON=?f] ->
  ART[BAR=0]
  NOUN[BAR=2, POSS=[], PLUR=?a, ANP=?b, CAS=?c, PRON=?f]
NOUN[POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f, BAR=?g] ->
  PUNCT[TYPE='DQUOTE']
  NOUN[BAR=?g, POSS=?a, PLUR=?b, ANP=?c, CAS=?d, D=?e, PRON=?f]
  PUNCT[TYPE='DQUOTE']
NOUN/NOUN ->

ART[BAR=1, D=1, ME=?a, YOU=?b, PLUR=?c, PRON=?f] ->
  ART[D=1] PRO[ME=?a, YOU=?b, PLUR=?c, PRON=?f]
ART[D=1] -> DET
ADJ -> ADJ ADJ
ADJ -> ADV ADJ
ADJ -> NOUN ADJ[SRC=[STEM=VERB[], DERIV='PERF_PART']]
ADJ -> NOUN ADJ[SRC=[STEM=VERB[], DERIV='IMPERF_PART']]
ADJ -> PUNCT[TYPE='DQUOTE'] ADJ PUNCT[TYPE='DQUOTE']
ADJ -> ADJ PUNCT[TYPE=COMMA] ADJ
ADJ -> ADJ PUNCT[TYPE=COMMA] CONJ ADJ
NUM -> NUM NUM
NUM -> ADV NUM
NUM -> ADJ NUM
```

Received 15th June 2011