NP chunking in Hungarian

by

Gábor Recski

(*Previous Degrees Go Here*)

A Thesis Submitted to the Graduate Faculty

of Eötvös Loránd University of Sciences in Partial Fulfillment

of the

Requirements for the Degree

Master of Arts

Budapest, Hungary

2010

© 2010

Gábor Recski

All Rights Reserved

NP chunking in Hungarian

by

Gábor Recski

Approved:

Major Professor: András Kornai

Committee: *Committee member I*
Committee member II

Electronic Version Approved:

Dean's Name
Dean of the Graduate School
Eötvös Loránd University of Sciences
May 2010

Acknowledgments

Firstly I would like to express my gratitude to those who introduced me to the fascinating nature of language structure. For their tedious work as my professors and for their everlasting faith in hard work and freedom of thought I would like to thank in particular Huba Bartos, Bea Gyuris, László Kálmán, Péter Rebrus, Péter Siptár, Péter Szigetvári, Miklós Törkenczy and László Varga.

For acquainting me with natural language processing and for offering all possible help in my studies of the field I owe my deepest thanks to my friends and mentors Péter Halácsy and Dániel Varga. They have played a key role in making the past years of my studies both rewarding and enjoyable.

For their helpfulness and patience over the last months I am thankful to my friends and colleagues at SZTAKI and in particular to Attila Zséder, who has been generous and kind to me beyond measure.

Finally, I would like to express the deepest gratitude to my advisor, András Kornai. András has made these past months bearable by his unlimited willingness to share his time and knowledge, as well as by his faith in my endeavours even at the most difficult of times.

Table of Contents

Chapter 1

Introduction

The following thesis will describe various approaches to the task of extracting noun phrases from Hungarian text. This introductory chapter will briefly outline the task of NP (Noun Phrase) chunking and give a survey of previous definitions of and approaches to the task. Chapter 3 will describe the process of creating an NP corpus for the purposes of evaluating our parser and for training and testing the machine learning system. Section 3.1 gives a short description of the KR formalism, a convention which we use to represent morphological information in order to make both systems compatible with the `hun*` toolchain. Section 3.2 will describe our methods for extracting NPs from a treebank.

Chapter 4 will describe our attempts at building an NP parser for Hungarian by implementing the NP grammar in (Kornai 1985) and revising it as necessary. Chapter 5 will describe statistical approaches to NP chunking. We give an outline of a supervised learning system which employs Maximum Entropy-based tagging and Hidden Markov Models to identify NP chunks (`hunchunk`, Recski and Varga 2010) and discuss its features and parameters. The chapter is concluded by an evaluation of the system's performance.

Chapter 6 of the thesis explores the possibility of using our parser's output as a source of features for `hunchunk` in an effort to obtain results superior to both individual systems.

Chapter 2

The chunking task

2.1 Definition

The term chunk and the task of text chunking do not have universally accepted definitions in NLP (Natural Language Processing) literature. The term chunk was first used by Abney, who usees it to describe non-overlapping units of a sentence that each consist of a single content word surrounded by a constellation of function words (1991). Based primarily on Gee and Grosjean (1983), 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 (Tjong Kim Sang and Buchholz 2000), where the Penn Treebank (Marcus et al. 1994) was used as a source of chunk data.

By far the most broadly covered subtask is that of NP chunking, and the rest of this section will discuss possible definitions of NP chunks only. One of the best known works on the extraction of NP chunks is that of Ramshaw and Marcus (1995), who define *baseNP*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 baseNP tagging based on the datasets proposed by Ramshaw and Marcus and adopted by CoNLL-2000. The architecture and performance of some recent statistical NP-chunkers will be briefly outlined in section 5.1.

2.2 Maximal NPs in machine translation

This thesis will also attempt to solve the task of finding the highest level NPs in a Hungarian sentence. Since chunking is often described as an algorithm that can supply, with high-accuracy, basic information concerning sentence structure, most existing tools were designed to identify non-recursive noun phrases. Even though the extraction of top-level NPs is a more difficult task than baseNP tagging, establishing maximal constituents of a sentence is essential for several tasks in natural language processing, such as information extraction, information retrieval, named entity recognition, and machine translation.

It is the latter application which can make the greatest use of maximal NPs. Statistical machine translation (SMT) systems which use multi-language corpora to create translation models often rely on various kinds of analyses which are available for both source and target languages. One issue which every SMT system must address is that the difference between the syntax of various languages makes it difficult to find regular correspondences between lexical categories. If top-level NPs can be found with a relatively high accuracy we may create an SMT system that consists of (1) a statistical translation system of noun phrases and (2) a rule-based system to account for the difference between the ordering of sentence constituents in various languages. For our recent work on the task of NP-alignment see (Recski et al. 2010), for progress on extracting word-level correspondences based on bilingual corpora cf. (Recski et al. 2009)

Chapter 3

Creating NP corpora

3.1 The KR formalism

The KR formalism for representing morphological information (Kornai 1989, Rebrus, Kornai and Varga 2010) 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 full KR 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 are given in (1a-e). As can be seen, KR encodes the entire chain of derivations that led to the word form under analysis.

(1a)	(1b)
tanárunk	*óráján*
teacher-Poss1Pl	class-Poss3-ON
'our teacher'	'in his/her class'
`tanár/NOUN<POSS<1><PLUR>>`	`óra/NOUN<POSS><CAS<SUE>>`

(1c)

másodikkal

two-ORD-WITH

'with the second'

`kettö/NUM[ORD]/NUM<CAS<INS>>`

(1d)

vegyük

take-Imp-Pl1-Def

'let's take'

`vesz/VERB<SUBJUNC-IMP><PERS<1>><PLUR><DEF>`

(1e)

felértékelődése

up-value(V)-Med-Ger-Poss3

'the increase of its value'

`felértékel/VERB[MEDIAL]/VERB[GERUND]/NOUN<POSS>`

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 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 (2a/f)

(2a)	(2b)	(2c)
ez	*mindenki*	*valami*
this	everybody	something
`ez/NOUN<PRON<DEM>>`	`mindenki/NOUN<PRON<GEN>>`	`valami/NOUN<PRON<INDEF>>`

(2d)	(2e)	(2f)
aki	*ki*	*saját*
who (relative pron.)	who (interrogative pron.)	own
`aki/NOUN<PRON<REL>>`	`aki/NOUN<PRON<WH>>`	`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. e.g. Abeillé 2003). For this purpose we use the Szeged Treebank, 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 (Csendes et al. 2005). Since we are interested in the extraction of both baseNPs and maxNPs, we created two separate corpora. The maxNP corpus was created by extracting all tokens and tagging those dominated by a top-level NP as belonging to the same chunk. The baseNP corpus was created by tagging all non-recursive NPs as separate chunks.

The treebank contains morphological information about each word in the MSD format. 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 the 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 corpora

Having created a baseNP and a maxNP corpus by the methods 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 our tools, we need it to reflect the analyses which we expect our systems to produce. In this paper we do not wish to argue 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

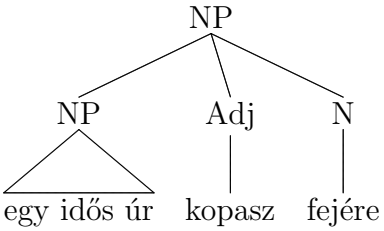
The largest number of cases where there is a discrepancy between the Szeged analysis and the one used here is related to the analysis of possessive constructions. The noun phrase in (3a) is represented in the treebank as in (3b).

(3a)

egy	idős	úr	kopasz	fejére
an	elderly	gentleman	bald	head-POSS-3-ON

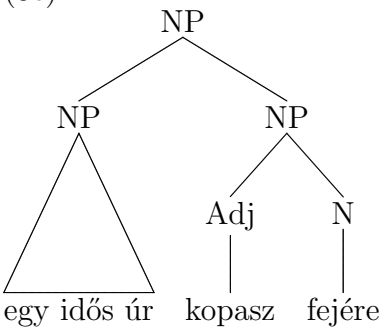
'on the bald head of an elderly gentleman'

(3b)



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 baseNP corpus in order to reflect the analysis in (3c), which we believe to be the correct one.

(3c)



3.3.2 *'Ez a'* demonstrative

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

(4a)

ez	a	pincér
this	the	waiter

'this waiter'

(4b)

ezek	a	hajók
this-PLUR	the	ship-PLUR

'these ships'

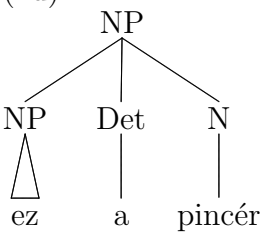
(4c)

attól	a	pasastól
that-FROM	the	bloke-FROM

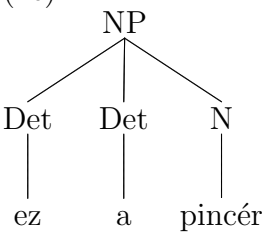
'from that bloke'

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

(4d)



(4e)



3.3.3 Other issues

The chunk corpora extracted from the Szeged Treebank still present a number of small anomalies that hinder the evaluation of both the rule-based and the machine learning-based 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 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 corpora) 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 baseNPs which our grammar is unable to parse.

3.4 Evaluation methods

In this thesis we shall discuss two systems which take as their input tokenized sentences with morphological information available for each word. The NP corpora, besides serving as training data for the machine-learning system, will be used to evaluate the performance of both systems at various stages of development. When evaluating the machine learning system, the corpora are each split into a train and test part. Our system is then trained on the former and evaluated on the latter.

The evaluation involves comparing two sets of chunks for each sentence, the one supplied by some system and the one present in the corpus, known as the *gold standard*. 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. Makhoul et al. 1999).

Chapter 4

Rule-based approaches

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 (Kornai 1985, Kornai 1989) in order to account for even the most complicated structures. Section 4.1 will describe the technical preliminaries of our system. In section 4.2 we turn our attention to grammar development and evaluate the performance of the parser using different versions of our grammar.

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 (Bird et al. 2009). 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 (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.

(5a)

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

(5b)

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

(5c)

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

(5d)

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

(5e)

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

(5f)

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

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 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 needing to parse 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 give 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 potential NPs of the sentence. When searching for maximal NPs, we discard all fragments which are contained in a larger fragment (when tagging baseNPs, we must first discard all fragments which contain more than one noun and only then proceed with this step). The second step of 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 by Kornai (1985) (repeated in Appendix A). We evaluate each version of the grammar on a test corpus which contains 1000 sentences picked randomly from all genres in the baseNP 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 (6).

(6)

```
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. (7))

(7a)

minden *pofon*

all punch

'all punches'

(7b)

néhány *villanykörte*

some light-bulb

'some light-bulbs'

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 (8).

(8a)

```
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]
```

(8b)

```
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 special "*ez a*"-structure described in section 3.3.2. To allow the parser to recognize the structure we implement the rule in (9):

(9)

```
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],
```

thus achieving an F-score of 86.68.

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

(10a)

a	*korsónak*	*támasztott*	*könyvet*	*olvasta*
the	jug-DAT	prop-PERF_PART	book-ACC	read-PAST-DEF-3

'she read the book propped up against the jug.'

(10b)

az	*ókori*	*mór*	*hódítóktól*	*származó*
the	ancient	moor	conqueror-Pl-FROM	originate-IMPERF_PART
esküvést	*hallották*			
oath-ACC	hear-PAST-DEF-3			

'They heard the oath originating from ancient moor conquerors'

Since our terminal symbols encode the 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 (11) to our grammar.

(11)

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

This addition caused an increase in the performance of the system to 87.87%. In the end the greatest 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 B) 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 (12)

(12)

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

As can be seen form these figures our development of the grammar corrected nearly half of the errors made by the system.

Chapter 5

Statistical methods

We shall now turn our attention to statistical approaches to noun phrase chunking. First we shall give a brief overview of some recent solutions that make use of various machine learning algorithms. Section 5.2 will describe `hunchunk`, an NP chunker that uses Maximum Entropy learning and Hidden Markov Models to find the most probable chunking for a given sentence (Recski and Varga 2010). At the end of the chapter we shall describe the evaluation of our system and discuss the results as well as examine the output of the tagger.

5.1 Overview of statistical methods in chunking

One of the most widely known works on chunking is that of Ramshaw and Marcus (1995), the paper that introduced NP chunking as a machine learning task. Besides defining the task of NP chunking as the identification of non-recursive (base) noun phrases, they 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 (Brill 1993a) and parsing (Brill 1993b). 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 (Marcus et al. 1994).

During and after the CoNLL shared task in 2000, a wide variety of machine learning methods have been applied to the task of identifying baseNPs. Kudo and Matsumoto reached

an F-score of 93.79% by using Support Vector Machines (2000), a result that was to increase to 94.22% a year later when they introduced weighted voting between SVMs trained using different chunk representations (Kudo & Matsumoto 2001). Probably the most popular method for NP chunking today is the Conditional Random Field (CRF, Lafferty et al. 2001) machine learning algorithm. CRFs have been used on the standard CoNLL task by Sha and Pereira, who achieved an F-score of 94.3% (2003), and more recently by Sun et al. (2008, 94.34%).

A further notable result is that of Hollingshead et al. (2005), 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 (Charniak 2000). These results show that a rule-based system proves to be competitive with results obtained by using any advanced machine learning algorithm, a fact that clearly points us in the direction of hybrid systems.

5.2 The `hunchunk` system

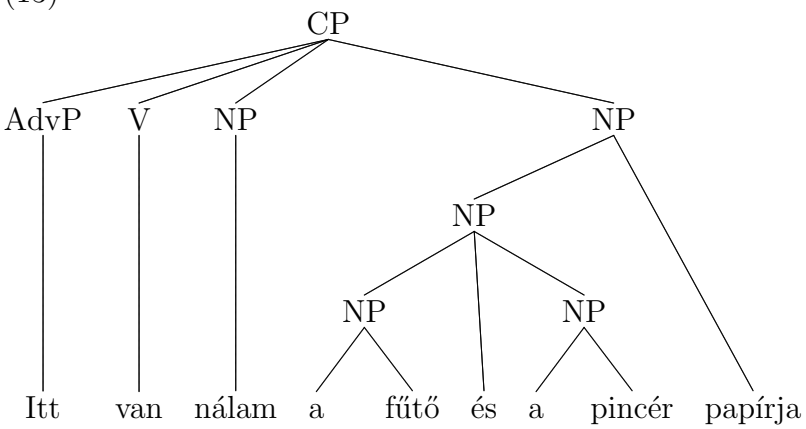
5.2.1 Labeling

The first step of solving the chunking task was turning it into a sequence labeling task. Given a sequence of words, each token receives one of five tags. `B-NP` and `E-NP` mark words at the beginning and end of NPs respectively. The tag `1-NP` is given to words which constitute an NP themselves, `I-NP` marks all other tokens within a chunk, and `O` is given to all tokens which are not part of an NP sequence. This labeling convention, called *Start-End tagging* (Uchimoto et al 2000), requires twice as many tags than the more widely used IO and IOB conventions (Tjong Kim Sang and Veenstra 1999), which enables our algorithms to learn typical features of words in all chunk positions. In contrast, the IOB system, which is by far the most widely used tagging system, and was chosen for CoNLL 2000 as well, makes use of only three tags (`B-NP, I-NP, O`), thus allowing for less fine distinctions between various contexts. We have also extracted from the corpus information about the complexity of NPs. A maximal NP which does not dominate any lower-level NP received a complexity measure of 1, while every other chunk received the tag `2+` to indicate complexity of 2 or greater. This

distinction was beneficial as it allowed for even finer distinctions to be made by the machine learning system. As there is no need for a tool to supply such complexity information about identified chunks in its output, this information is discarded at the end of the chunking process.

The labeling process detailed above is exemplified below. The sentence represented in the treebank as (13) will be labeled in the chunk corpus as in (14).

(13)



(14)

Itt	van	nálam	a	fűtő	és	a	pincér	papírja
O	O	1-N_1	B-N_2+	I-N_2+	I-N_2+	I-N_2+	I-N_2+	E-N_2+

5.2.2 Features

Before training a Maximum Entropy model we must first decide on the set of features we wish to use to represent each token in the training data. The form and part-of-speech tag of a word are the two features employed by virtually all similar systems based on any machine learning algorithm, often exclusively (e.g. Sha & Pereira 2003). In the case of most such systems the POS-tag feature of a word takes a single value, the tag itself. The KR formalism, however, which we have described in detail in section 3.1, encodes all morphological features in separate strings, therefore it is possible to encode all morphological features as separate feature values. A word with the KR-tag `NOUN<PLUR><POSS>`, for example, will receive the

features `kr=NOUN, kr=PLUR` and `kr=POSS`. This enables the maxent model to learn about the effect of each grammatical feature separately.

In order to make it possible for the model to learn some typical morphemes of Hungarian we also use character trigrams. A token's feature list is compiled by adding the values of all these features for each word in a 5-token radius of the word in question.

A final feature which we have introduced encodes information about the sequence of part-of-speech tags in a given context. If a word in position i of a sentence is denoted by w_i and its POS-tag by p_i then the values for the POS pattern feature for w_i will be all subintervals of the series $p_{i-r} \ldots p_{i+r}$. Intervals of length 1 are omitted since they are equivalent to plain KR-codes which are already represented. Feature values are prefixed with numbers indicating the start and end position of the given interval relative to w_i. Given the sentence and POS-tag sequence in (15) the word *csapos* will receive the POS pattern features in (16) ($r = 3$).

(15)

A	csapos	mesélte	,	hogy	milyen	szép	kés	van	bennem	.
ART	NOUN	VERB	PUNCT	CONJ	ADJ	ADJ	NOUN	VERB	NOUN	PUNCT

(16)

-1_0_ART+NOUN
-1_1_ART+NOUN+VERB
-1_2_ART+NOUN+VERB+PUNCT
-1_3_ART+NOUN+VERB+PUNCT+CONJ
0_1_NOUN+VERB
0_2_NOUN+VERB+PUNCT
0_3_NOUN+VERB+PUNCT+CONJ
1_2_VERB+PUNCT
1_3_VERB+PUNCT+CONJ
2_3_PUNCT+CONJ

Increasing the value of r yields higher F-score values for the chunking task. Unfortunately, setting it to 4 or greater leads to an increase in the number of features to a level which does not allow the training of a Maximum Entropy model anymore. The contribution of each feature class used will be discussed in section 5.3.

5.2.3 Tagging

One advantage of the Maximum Entropy method is that, given a set of features for a token, a Maxent model not only returns the most probable tag but supplies a complete probability distribution over all possible tags. This means that for any word w and tag t the model computes $P(t|w)$, i.e. the probability that the word w receives tag t. This enables us to model the labeling task with a Hidden Markov Model (HMM, cf. e.g. Rabiner 1989) where the hidden states are chunk-tags and the surface states are features sets corresponding to words. The transition probabilities, i.e. $p(t_i|t_{i-1})$ are estimated by the relative frequency

$$\frac{\hat{p}(t_{i-1}, t_i)}{\hat{p}(t_{i-1})}.$$

where the frequencies are those observed in the training corpus. The emission probabilities, on the other hand, are of the form $p(w|t)$, i.e. the probability that given the chunk-tag t the observed word will be w. Since the Maxent model supplies the value of $p(t|w)$ we can compute the emission probability using Bayes' theorem:

$$p(w|t) = \frac{p(t|w)p(w)}{p(t)}$$

The prior probability of a tag t is once again estimated from the observed frequency $\hat{p}$. The prior probability of a given word $p(w)$, however, may be disregarded when calculating the most probable tag sequence for a given sentence, as all probabilities $p(w_1), \ldots p(w_i)$ are constant. During labeling, the system has to find the most likely tag sequence for a given sentence, i.e. the tags $t_1 \ldots t_k$ for which the value of the following product is highest:

$$\prod_i \frac{\hat{p}(t_i|w)\hat{p}(t_i, t_{i-1})}{p(t_i)}.$$

The maximum of this formula (that is, our estimate of the best labeling) can be efficiently found by the Viterbi algorithm.

5.3 Evaluation

5.3.1 The CoNLL task

We begin the evaluation of the `hunchunk` system by training and testing on the standardized datasets of the CoNLL-2000 task. On this baseNP chunking task our system achieves an F-score of 93.73%. This performance is competitive compared to state-of-the art CRF-based NP chunkers: Sha and Pereira (2003) and most recently Xu et al. (2008) report F-scores of 94.29% and 94.34% respectively. These systems perform slightly better than `hunchunk`, however their training time is an order of magnitude longer.

5.3.2 Results on Hungarian NPs

In order to test the performance of our NP chunker we used .5 million tokens from each chunk corpus. Tagging was performed with the language model weight set to various values between 0 and 1, thus we could establish the optimal value at 0.3 for maximal NPs and 0.8 for baseNPs. Evaluation of the output was done according to the standards of the CoNLL task: a chunk is tagged correctly if and only if all its tokens and no other are identified correctly as belonging to the chunk. The baseline method involved assigning each token the most probable chunk-tag based on its part-of-speech tag, using the simplest tagset B-I-O, which makes use of three tags only (`B-NP`, `I-NP` and `O`). The performance of the baseline method and hunchunk on the task of tagging baseNPs are listed in table 5.1.

	Precision	Recall	F-score
baseline	53.99%	24.83%	34.02%
`hunchunk`	**94.61%**	**94.88%**	**94.75%**

Table 5.1: Results of baseNP-tagging

In the case of maxNP-chunking we also trained and tested two state-of-the-art machine learning systems which achieved competitive results on the CoNLL task. The `Yamcha` system uses Support Vector Machines (SVMs) (Kudo and Matsumoto 2000) while the `mallet` chunker makes use of the CRF algorithm (McCallum 2002). When using either of these algorithms, training of a model may take up to an order of magnitude longer than in case of a Maximum Entropy Model. Therefore it is virtually impossible to experiment with a large number of different feature sets and training parameters. Also, there are much more severe computational limitations to the number of features and the size of training data that these algorithms allow, making it impossible to train them with the same set of features as the one we used for `hunchunk`. In our comparison each system was trained with the largest training data and feature sets possible. The performance of both of these systems as well as those of `hunchunk` and the baseline system are shown in table 5.2

	Precision	Recall	F-score
baseline	56.87%	35.44%	43.66%
mallet	80.93%	86.66%	83.70%
yamcha	78.59%	87.47%	82.79%
`hunchunk`	**89.34%**	**88.12%**	**88.72%**

Table 5.2: Results of maxNP-tagging

5.3.3 Effects of feature groups

In order to gain insight about the contribution of individual feature classes to the performance of the chunker we proceeded to train models using various subsets of the 4 feature classes (form, KR, n-gram, and POS-pattern)[1]. Results obtained by using three out of four feature groups indicate that all feature classes contribute to the final performance (cf. table 5.3). We can also safely say that neither of the four groups is truly essential for the task: even the

[1]The data in this section were obtained using a slightly different dataset than those used for the performance measures. The latter involve a train and test corpus which are representative of the entire corpus in terms of genre, therefore they give a better indication of the general performance of the system. The figures in this section are nevertheless comparable to each other as they were all obtained using the same train and test set, extracted randomly from our maxNP corpus.

Features	F-score
form, KR, n-grams, POS-patterns	89.68%
form, KR, n-grams	88.39%
form, KR, POS-patterns	88.89%
form, n-grams, POS-patterns	87.72%
KR, n-grams, POS-patterns	88.90%

Table 5.3: Results of maxNP-tagging using 3 feature groups

Features	F-score
form, KR	87.68%
form, n-grams	83.45%
form, POS-patterns	86.73%
KR, n-grams	87.31%
KR, POS-patterns	83.73%
n-grams, POS-patterns	87.36%

Table 5.4: Results of maxNP-tagging using 2 feature groups

omission of KR-codes, i.e. discarding all morphological information except what is implicitly encoded in the POS-pattern features, will lead only to a $\sim 2\%$ decline in performance.

Let us now examine the results achieved by using any two of the four feature groups in table 5.4. These figures tell us that a substantial drop in performance occurs when the two feature classes encode the same type of information: surface data (form and n-grams) or grammatical features (KR and POS-patterns). This behaviour is hardly surprising: to achieve near-optimal performance we need at least one feature group for both types of information.

Finally let us take a look at the performance of models trained with a single feature type (table 5.5). We now experience a substantial decrease in performance, yet still we achieve F-scores 10-20% above the baseline method. It is notable that using only word form as a feature for training we can achieve nearly as high numbers as when using all available morphological information. Since morphological information is also entailed by the POS-pattern features,

it is only the n-gram features that achieve significantly lower F-scores when used exclusively. This might not be the case if we used n-grams longer than 3 character, since in that case we could except character sequences that coincide with morphemes of the Hungarian language to encode grammatical information implicitly. However, as we described earlier, we have already pushed our system to its computational limitations when implementing the POS-pattern feature.

Features	F-score
form	78.19%
KR	79.39%
POS-patterns	77.13%
n-grams	68.19%

Table 5.5: Results of maxNP-tagging using 1 feature group

Chapter 6

Hybrid solutions

This final chapter of the thesis will describe some experiments aimed at creating an NP chunking system with both rule-based and statistical components. We will show that such a hybrid system is capable of achieving results superior to those of the two individual systems described in previous chapters. We will also describe an attempt at using baseNPs in the task of extracting maximal noun phrases.

6.1 Features from the parser

We attempted to improve the performance of `hunchunk` by supplying it with the output produced by our NP-parser. Our simplest experiment involved adding a single feature to the set used by the Maximum Entropy system. Besides the form, KR , n-gram and POS-pattern features each word was assigned one of the five chunk tags (`B-NP, I-NP, E-NP, 1-NP, O`) based on its position in the chunking produced by the parser.

In case of baseNP-chunking we started by parsing the entire chunk corpus and then creating a train and test set from the data obtained in the same way as we have done when evaluating `hunchunk` on its own. Adding the feature defined above to our original set produces the results shown in table 6.1.

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

Table 6.1: baseNP-tagging with parser features

As can be seen from the above figures, the addition of information from a rule-based system lead to a 15% decrease in the number of errors made by the statistical system. In the case of maximal noun phrases the parser feature also causes some increase in performance (cf. table 6.2)

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

Table 6.2: Results of maxNP-tagging with parser features

6.2 Multi-level parsing

In an attempt to increase the performance of our maximal noun phrase tagger we devised a method of maxNP-chunking which makes use of an existing baseNP tagging of a given text.

Given a tokenized and morphologically analyzed text which has undergone baseNP-tagging and now has a chunk-tag associated with each word, we take all sequences of tokens which have been marked as constituting a non-recursive noun phrase and merge them to create a single token of the corpus. This transformation is exemplified in (17a-b)

(17a)

Kérje	VERB<SUBJUNC-IMP><DEF>	O
meg	PREV	O
erre	NOUN<CAS<SBL>><PRON<DEM>>	B-NP
a	ART	I-NP
szívességre	NOUN<CAS<SBL>>	E-NP
valamelyik	ADJ<PRON<INDEF>>	B-NP
matrózt	NOUN<CAS<ACC>>	E-NP
.	PUNCT	O

(17b)

Kérje	VERB<SUBJUNC-IMP><DEF>
meg	PREV
erre_a_szívességre	NP
valamelyik_matrózt	NP
.	PUNCT

Once the corpus has been transformed this way, we may use it to train `hunchunk` for maxNP tagging. We expect such a model to produce better results than the original maxNP-tagger even if the baseNP chunks are generated automatically. This method, however, will produce results about five percentage points lower than the regular tagger, since the model is not trained for incorrect baseNP taggings (cf. table 6.1).

baseNP source	F-score
parser	76.61%
`hunchunk`	83.10%
hybrid	83.84%
gold standard	90.47%
without multi-level	88.72%

Figure 6.1: Results of maxNP chunking with baseNP data

The hybrid system described in the previous section is trained on parse tags that are sometimes incorrect, therefore the maxent model learns to be somewhat distrustful of such information. In the case of gold multi-level tagging, the training data contains tokens representing baseNPs and a gold standard maxNP chunking. Since a perfect maxNP chunking can only be given for a correct tokenization, i.e. one based on the gold standard baseNP tagging, it is not possible to train the model with noisy data and the quality of maxNP chunking will suffer seriously from errors in the baseNP chunking. Still, the fact that gold baseNP information leads to an increased F-score shows that the treatment of baseNPs as tokens is a step which makes automated maxNP chunking easier.

Chapter 7

Conclusion

In this thesis we have described various approaches to the task of extracting noun phrases from Hungarian text. We have attempted to solve two individual tasks, that of finding non-recursive (base) NPs and that of finding maximal NPs. We have described the process of creating NP-corpora from a treebank.

The thesis described our efforts for revising the context-free grammar of Hungarian NPs (Kornai 1985) and implementing it to create a parser for the task of NP-chunking. We also described the `hunchunk` system (Recski and Varga 2010), a statistical system using Maxiimum Entropy learning and Hidden Markov Models. Having tested `hunchunk` on a standardized task of English NP chunking we have also shown that we have created a state-of-the-art machine learning system.

Finally we have shown that a system combining the knowledge encoded by each of the two systems will produce superior results on both tasks of NP-chunking. In order to achieve these results, we pushed both the statistical and the hybrid system to the limits of our current computational capacities. Therefore, we expect to reach even better results in the future by running our tools on hardware with larger memory capacity.

Appendix A

The initial grammar of Hungarian NPs

(1F) N< 1 αPOS βPL γANP δCAS ϵD>→

(A< n -POS -PL -ANP -CAS>) N< 0 αPOS βPL γANP δCAS ϵD>

(1G) N< 1 αPOS βPL γANP δCAS ϵD>/N →

(A< n -POS -PL -ANP -CAS>) N< 0 αPOS βPL γANP δCAS ϵD>/N

(2F) N< 2 αPOS -PL βANP γCAS δD>→

(Num< n -POS -PL -ANP -CAS>) N< 1 αPOS -PL βANP γCAS δD>

(2G) N< 2 αPOS +PL βANP γCAS δD>→ N< 1 αPOS +PL βANP γCAS δD>

(2H) N< 2 αPOS -PL βANP γCAS δD>/N →

(Num< n -POS -PL -ANP -CAS>) N< 1 αPOS -PL βANP γCAS δD>/N

(2I) N< 2 αPOS +PL βANP γCAS δD>/N →

N< 1 αPOS +PL βANP γCAS δD>/N

(3F) N< 3 αPOS βPL γANP δCAS ϵD>→

Art< ϵD> N< 2 αPOS βPL γANP δCAS $\pm$D>

(3G) N< 3 αPOS βPL γANP δCAS +D>→

N< 0 αPOS βPL γANP δCAS +D>

(3H) N< 3 αPOS βPL γANP δCAS ϵD>/N →

Art< ϵD> N< 2 αPOS βPL γANP δCAS $\pm$D>/N

(5F) N< 3 αPOS βPL ANP< γ PL> δCAS ϵD>=

N< 3 αPOS βPL -ANP -CAS $\pm$D> +

+ N< 2 +POS γPL -ANP δCAS ϵD>/N

(6F) N<3 -POS αPL βANP γCAS +D>→

N<3 ±POS ±PL -ANP -CAS ±D> N<2 +POS αPL βANP γCAS ±D>

(7F) N<4 -POS αPL βANP γCAS +D>→

N<3%4 ±POS ±PL ±ANP +DAT ±D> N<3 +POS αPL βANP γCAS +D>

(8F) N<3 -POS αPL βANP γCAS +D>→Art<1 +D δME ϵYOU ζPL>

N<2 POS< δME ϵYOU ζPL> αPL βANP γCAS ±D>

(8G) N<3 -POS αPL βANP γCAS +D>→Art<0 ±D>

N<2 POS< ±ME ±YOU ±PL> αPL βANP γCAS ±D>

(8H) Art<1 +D αME βYOU γPL>→Art<+D> Pro< αME βYOU γPL>

Appendix B

The revised grammar of Hungarian NPs

(1F) N< 1 αPOS βPL γANP δCAS ϵD>→

(A< n -POS -PL -ANP -CAS>) N< 0 αPOS βPL γANP δCAS ϵD>

(1G) N< 1 αPOS βPL γANP δCAS ϵD>/N →

(A< n -POS -PL -ANP -CAS>) N< 0 αPOS βPL γANP δCAS ϵD>/N

(2F) N< 2 αPOS -PL βANP γCAS δD>→

(Num< n -POS -PL -ANP -CAS>) N< 1 αPOS -PL βANP γCAS δD>

(2G) N< 2 αPOS +PL βANP γCAS δD>→ N< 1 αPOS +PL βANP γCAS δD>

(2H) N< 2 αPOS -PL βANP γCAS δD>/N →

(Num< n -POS -PL -ANP -CAS>) N< 1 αPOS -PL βANP γCAS δD>/N

(2I) N< 2 αPOS +PL βANP γCAS δD>/N →

N< 1 αPOS +PL βANP γCAS δD>/N

(3F) N< 3 αPOS βPL γANP δCAS ϵD>→

Art< ϵD> N< 2 αPOS βPL γANP δCAS $\pm$D>

(3G) N< 3 αPOS βPL γANP δCAS +D>→

N< 0 αPOS βPL γANP δCAS +D>

(3H) N< 3 αPOS βPL γANP δCAS ϵD>/N →

Art< ϵD> N< 2 αPOS βPL γANP δCAS $\pm$D>/N

(5F) N< 3 αPOS βPL ANP< γ PL> δCAS ϵD>=

N< 3 αPOS βPL -ANP -CAS $\pm$D> +

+ N< 2 +POS γPL -ANP δCAS ϵD>/N

(6F) N<3 -POS αPL βANP γCAS +D>→

N<3 ±POS ±PL -ANP -CAS ±D> N<2 +POS αPL βANP γCAS ±D>

(7F) N<4 -POS αPL βANP γCAS +D>→

N<3%4 ±POS ±PL ±ANP +DAT ±D> N<3 +POS αPL βANP γCAS +D>

(8F) N<3 -POS αPL βANP γCAS +D>→Art<1 +D δME ϵYOU ζPL>

N<2 POS< δME ϵYOU ζPL> αPL βANP γCAS ±D>

(8G) N<3 -POS αPL βANP γCAS +D>→Art<0 ±D>

N<2 POS< ±ME ±YOU ±PL> αPL βANP γCAS ±D>

(8H) Art<1 +D αME βYOU γPL>→Art<+D> Pro< αME βYOU γPL>

(9A) A → A A

(9B) A → Adv A

(10A) Num → Num Num

(10B) Num → Adv Num

(10C) Num → A Num

(11A) N<αPOS βPL γANP δCAS ϵD ζPRON>→N<PRON<+GEN>>

N<1 αPOS βPL γANP δCAS ϵD ζPRON>

(11B) N<αPOS βPL γANP δCAS ϵD ζPRON>→ N<PRON<+INDEF>>

N<1 αPOS βPL γANP δCAS ϵD ζPRON>

(11C) N<αPOS βPL γANP δCAS ϵD>→

N<0 PRON<+DEM> αPOS βPL γANP δCAS ϵD> Art

N<2 αPOS βPL γANP δCAS ϵD>

(12A) A → N A<SRC<STEM<+VERB> DERIV<+PERF_PART>>>

(12B) A → N A<SRC<STEM<+VERB> DERIV<+IMPERF_PART>>>

Bibliography

[1] A. Abeillé. *Treebanks: Building and using parsed corpora*. Springer Netherlands, 2003.

[2] S. Abney. Parsing by chunks. *Principle-based parsing*, pages 257–278, 1991.

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

[4] E. Brill. A corpus-based approach to language learning, 1993.

[5] E. Brill. Automatic grammar induction and parsing free text: A transformation-based approach, 1993.

[6] E. Charniak. A maximum-entropy-inspired parser. In *ANLP00*, 2000.

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

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

[9] K. Hollingshead, S. Fisher, and B. Roark. Comparing and combining finite-state and context-free parsers. In *Proceedings of HLT-EMNLP*, pages 787–794, 2005.

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

[11] A. Kornai. A fonévi csoport egyeztetése [Agreement in noun phrases]. *Általános Nyelvészeti Tanulmányok*, pages 183–211, 1989.

[12] T. Kudo and Y. Matsumoto. 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.

[13] T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *Proceedings of NAACL 2001*, 2001.

[14] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel. Performance measures for information extraction. In *Broadcast News Workshop'99 Proceedings*, page 249. Morgan Kaufmann Pub, 1999.

[15] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19:313–330, 1994.

[16] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[17] R. L. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proc. IEEE*, volume 77, pages 257–286, 1989.

[18] L. A. Ramshaw and M. P. Marcus. Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpora, Cambridge, MA, USA*, 1995.

[19] P. Rebrus, A. Kornai, and D. Varga. Egy általános célú morfológiai annotáció [a general-purpose annotation of morphology]. *Általános Nyelvészeti Tanulmányok*, 2010.

[20] G. Recski, A. Rung, A. Zséder, and A. Kornai. NP-alignment in bilingual corpora. In *Tenth International Conference on Language Resources and Evaluation, LREC'10*, 2010.

[21] G. Recski and D. Varga. Magyar fonevi csoportok azonosítása [Identifying Hungarian noun phrases]. *Általános Nyelvészeti Tanulmányok*, 2010.

[22] G. Recski, D. Varga, A. Zséder, and A. Kornai. Fonevi 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.

[23] E. F. Tjong Kim Sang and J. Veenstra. Representing text chunks. In *EACL*, pages 173–179, 1999.

[24] F. Sha and F. Pereira. 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 on Human Language Technology*, pages 134–141, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[25] X. Sun, L.-P. Morency, D. Okanohara, and J. Tsujii. 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, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

[26] E. F. Tjong Kim Sang, S. Buchholz, and K. Sang. Introduction to the CoNLL-2000 shared task: Chunking, 2000.

[27] K. Uchimoto, Q. Ma, M. Murata, H. Ozaku, and H. Isahara. Named entity extraction based on a maximum entropy model and transformation rules. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 326–335, Morristown, NJ, USA, 2000. Association for Computational Linguistics.