Workshop Programme

14:30 – 15:00	**Linguistic richness and technical aspects of an incremental finite-state parser** Hrafn Loftsson, Eiríkur Rögnvaldsson
15:00 – 15:30	**Partial parsing in a simple MT system** Petr Homola, Vladislav Kuboň
15:30 – 16:00	**Shallow parsing in sentiment analysis of product reviews** Aleksander Buczyński, Aleksander Wawer
16:00 – 16:30	coffee break
16:30 – 17:00	**Why is this wrong? – Diagnosing erroneous speech recognizer output with a two phase parser** Bernd Ludwig, Martin Hacker
17:00 – 17:30	**Chunking and dependency parsing** Giuseppe Attardi, Felice Dell'Orletta

Workshop Organisers

Workshop Programme Committee

Table of Contents

Author Index

Linguistic richness and technical aspects of an incremental finite-state parser

Hrafn Loftsson[*], **Eiríkur Rögnvaldsson**[†]

[*]School of Computer Science, Reykjavik University
Kringlan 1, Reykjavik IS-103, Iceland
hrafn@ru.is

[†]Department of Icelandic, University of Iceland
Árnagarður við Suðurgötu, Reykjavik IS-101, Iceland
eirikur@hi.is

Abstract

We describe the linguistic richness and the technical aspects of an incremental finite-state parser for Icelandic. We argue that our parser outputs a linguistically rich annotation which in many simple sentences amounts to full parsing. Additionally, we provide arguments for various technical design and implementation decisions regarding the parser. Our description may be used as guidelines for other researchers developing similar parsers.

1. Introduction

Syntactic analysis for natural languages is often divided into two categories: *full parsing*, in which a complete analysis for each sentence is computed, and *partial (or shallow) parsing*, where sentence parts or chunks are analysed without building a complete parse tree. The aim of partial parsing is "to recover syntactic information efficiently and reliably from unrestricted text, by sacrificing completeness and depth of analysis" (Abney, 1996).

In many natural language processing (NLP) applications, it can be sufficient to analyse sentence parts or phrases. This can be the case, for example, in applications like information extraction, machine translation, and some types of grammar checking, in which identification of phrases is more important than a global parse. Additionally, in cases of low quality input or spoken language, a partial parsing method can be more robust than a full parsing method, because of noise, missing words, and mistakes in the input (Li and Roth, 2001).

We have developed an incremental finite-state parser, *IceParser*, for parsing Icelandic text. IceParser, the first parser published for the Icelandic language, is designed to be used both as a stand-alone application and as an integrated part of an NLP toolkit.

In (Loftsson and Rögnvaldsson, 2007a), we briefly described the annotation scheme used by the parser, its individual modules, evaluation results, and error analysis. In this paper, we describe in detail the linguistic richness (in Section 2.) and the technical aspects (in Section 3.) of IceParser.

2. Linguistic richness

In this section, we describe the main features of our annotation scheme and show how it is applied. We argue that through the interplay of phrase structure annotation and syntactic functions annotation, accompanied by relative position indicators for arguments, we obtain a linguistically rich annotation which in many simple sentences amounts to full parsing.

2.1. The annotation scheme

When designing a parser for a natural language, it is important to outline an annotation scheme. In the context of shallow parsing this includes deciding what kind of chunks/phrases and grammatical functions to annotate, and writing guidelines (general principles) on how to perform the annotation (Voutilainen, 1997). Additionally, since "the correct analysis" is not always clear, detailed instructions may be needed where the general principles are not unambiguously applicable.

Voutilainen points out that the annotation scheme (or the grammatical representation, as he calls it) can be specified with the help of a *grammar definition corpus* (GDC). A GDC is a representative collection of sentences, consistently analysed using the guidelines and the detailed instructions. The purpose of the GDC is to "provide an unambiguous answer to the question how to analyse any utterance in the object language" (Voutilainen, 1997). Furthermore, the GDC can be used to help with the development of the parser itself, because the parser should at least be able to annotate correctly the sentences in the GDC.

We have designed a shallow annotation scheme (a thorough description of which can be found in (Loftsson and Rögnvaldsson, 2006)) that follows the dominant paradigm in treebank annotation, i.e. it is "the kind of theory-neutral annotation of constituent structure with added functional tags" (Nivre, 2002). During the design, we focused on making the annotation rich enough to be of use in various NLP applications, in particular, grammar correction. In addition to the annotation scheme, we constructed a GDC, a corpus consisting of 214 sentences, selected from the Icelandic Frequency Dictionary (IFD) corpus (Pind et al., 1991). The selected sentences represent the major syntactic constructions in Icelandic. This GDC was used as the development corpus for IceParser.

It is assumed that input text, to be annotated according to our scheme, has been morphologically tagged using the detailed part-of-speech (POS) tagset of about 700 tags described in (Pind et al., 1991). Although these tags

are morphological in nature, they also carry a substantial amount of syntactic information and the tagging is detailed enough for the syntactic function of words to be more or less deduced from their morphology and the adjacent words (Rögnvaldsson, 2006).
Thus, for instance, a noun in the nominative case can reasonably safely be assumed to be a subject, unless it is preceded by the copula *vera* 'to be' which is in turn preceded by another noun in the nominative, in which case the second noun is a predicative complement. A noun in the accusative or dative case can in most cases be assumed to be a (direct or indirect) object, unless it is immediately preceded by a preposition. As is well known, Icelandic also has accusative and dative subjects, and even some nominative objects (Thráinsson, 2007), but these can easily be identified from their accompanying verbs. We have compiled a list of those verbs which the parser consults.

2.1.1. Phrase structure annotation

According to our annotation scheme, two labels are attached to each constituent. The first one denotes the beginning of the constituent, the second one denotes the end (e.g. [NP . . . NP]). The main labels are AdvP, AP, NP, PP and VP – the standard labels used for phrase annotation (denoting adverb, adjective, noun, preposition, and verb phrase, respectively).
Additionally, we use the labels CP, SCP, InjP, and MWE for marking coordinating conjunctions, subordinating conjunctions, interjections, and multiword expressions (of which we have a list), respectively. Furthermore, we use the labels APs and NPs, for marking a sequence of adjective phrases (agreeing in gender, number and case) and noun phrases (agreeing in case), respectively.
Our scheme subclassifies VPs. A finite verb phrase is labelled as [VP . . . VP] and consists of a finite verb, optionally followed by a sequence of AdvPs and supine verbs. Other types of VPs are labelled as [VPx . . . VPx], where x can have the following values: *i*, denoting an infinitive VP; *b*, denoting a VP which demands a predicative complement (i.e primarily a verb phrase consisting of the verb *vera*); *s*, denoting a supine VP; *p*, denoting a past participle VP; *g*: denoting a present participle VP. Our VPs do not comprise verbal arguments and hence are strictly speaking not verb phrases, but rather bare verbs or verbal clusters. We also distinguish between four kinds of MWEs, i.e. expressions that function as i) a conjunction (MWE_CP), ii) an adverb (MWE_AdvP), iii) an adjective (MWE_AP), and iv) a preposition (MWE_PP).

2.1.2. Syntactic functions annotation

From a linguistic point of view, our constituent structure bracketing is of course rather primitive and in many ways incomplete, compared to the description in (Thráinsson, 2007), for instance. The structure it marks is relatively flat, and, since it is designed for partial parsing, recursiveness is not shown. Thus, many strings that in a complete parse tree would be grouped together in a single multi-level structure are shown as two or more distinct chunks.
Two such examples are shown below (the morphological tags from the input text are underlined):

(1) [NP niðurstöður (*results*) <u>nvfn</u> NP] [NP þessara (*these*, gen.pl.) <u>favfe</u> rannsókna (*research*, gen.pl.) *nvfe* NP]
'The results of this research'

(2) [NP húsin (*the houses*) <u>nhfng</u> NP] [PP í (*in*) <u>aþ</u> [NP fæðingarbæ (*hometown*) <u>nkeþ</u> mínum (*mine*) <u>fekeþ</u> NP] PP] [VPb voru (*were*) <u>sfg3fþ</u> VPb]
'The houses in my hometown were . . . '

In (1), the NP *þessara rannsókna* (gen.pl.) 'this research' is a genitive qualifier of the NP *niðurstöður* 'results'. In (2), the PP *í fæðingarbæ mínum* 'in my hometown' modifies the NP *húsin* 'the houses'. The constituent structure bracketing does not indicate any connection between the two NPs in (1), or between the NP and the PP in (2).
The syntactic functions annotation (functional tags), however, substitutes for the lack of hierarchical constituent structure to a considerable extent. We use curly brackets for denoting the beginning and the end of a syntactic function, as carried out by (Megyesi and Rydin, 1999). Four of the function labels we use, *SUBJ, *OBJ, *IOBJ, *COMP, are relatively standard, denoting a subject, an object, an indirect object, and a predicative complement, respectively. To deal with certain important characteristics of Icelandic syntax, we have added four nonstandard labels to mark NPs bearing different functions; *OBJAP, *OBJNOM, *QUAL, *TIMEX, denoting an object of an AP, a nominative object, a genitive qualifier, and a temporal expression, respectively. Additionally, for some of the syntactic function labels, we use relative position indicators ("<" and ">"). For example, *SUBJ> means that the verb is positioned to the right of the subject, *SUBJ< denotes that the verb is positioned to the left, while *SUBJ is used when the accompanying verb cannot be located, either because it is missing (in gapping structures, for instance) or because the distance between the subject and the verb is more than a parser can cope with. The motivation behind using the indicators is to simplify grammar checking at later stages. Similar indicators are, for example, used in the Constraint Grammar Framework (Karlsson et al., 1995).
By using the syntactic function labels and the relative position indicators, we manage to show the most important relations between phrases. Thus, in a sequence of two adjacent NPs with one of them in the genitive case, as in (1), the genitive NP is marked as *QUAL in order to show that it forms a syntactic unit with the other (governing) NP, as demonstrated in (3):

(3) [NP niðurstöður (*results*) <u>nvfn</u> NP] {*QUAL [NP þessara (*these*, gen.pl.) <u>favfe</u> rannsókna (*research*, gen.pl.) <u>nvfe</u> NP] *QUAL}

In a complete sentence, this string as a whole will be marked as a subject, an object, etc., according to its role in the sentence, as shown in (4):

(4) {*SUBJ> [NP niðurstöður (*results*) <u>nvfn</u> NP] {*QUAL [NP þessara (*these*, gen.pl.) <u>favfe</u> rannsókna (*research*, gen.pl.) <u>nvfe</u> NP] *QUAL} *SUBJ>} [VPb eru (*are*)

sfg3fn VPb] {*COMP< [AP [AdvP mjög (*very*) aa AdvP] óvæntar (*surprising*) lvfnsf AP] *COMP<}
'The results of this research are very surprising'

2.1.3. The output of IceParser

IceParser generates output according to the annotation scheme described above[1]. In many simple sentences, such as in (4), the annotation made using this scheme in fact amounts to a full parse. The phrase structure annotation and the syntactic functions annotation, together with the relative position indicators, give us all the information we need about the structure and the internal dependencies in this sentence. In more complex sentences, of course, the parsing may not be as complete as in this one. This is especially evident in cases of long distance dependencies and in sentences with embedded clauses and clauses with "gaps" of some kind, such as relative clauses.
IceParser makes no attempt at resolving PP attachment ambiguities – all PPs are shown as independent constituents. In the case of a PP following a sentence-initial NP, as in (2) above, we could make use of the fact that Icelandic is a V2 language, which does not allow more than one constituent preceding the finite verb (Thráinsson, 2007). Hence, a PP following an NP in front of a finite verb must be a part of this NP. We could of course show this by closing the function tag of the NP after the PP, but we have not implemented this yet.

2.2. Use of IceParser in grammar correction

Even though IceParser is designed and implemented as a partial parser, we believe that its output is sufficiently detailed to be of great use in many NLP applications, such as in information extraction, grammar correction, and machine translation. Here we will only briefly illustrate its potential use in grammar checking tools.
Among the most error-prone features of Icelandic grammar is morphological agreement and morphological government of various types. Verbs agree in person and number with their subject; predicative adjectives agree in gender, number, and case with the subject of the clause; all inflected words within a noun phrase agree in gender, number, and case; verbs govern the case of their subjects and objects; and so on.
In order to detect errors having to do with agreement or morphological government, it is especially important to nail down the relationship between verbs and their arguments. It has been shown that IceParser does a good job in correctly identifying subjects and objects (F-measure for subjects and objects is 90.5% and 88.2%, respectively (Loftsson and Rögnvaldsson, 2007a)).
In designing the parser, we deliberately chose to make only minimal use of the morphological information furnished by the POS tags in the input text. This was done in order to be able to use the parser in detecting grammatical errors. It is clear that if the parser relies too heavily on the morphological features, errors in the input will both result in parsing errors and also severely undermine the usefulness of the parser in grammar checking. Therefore, the parser mainly uses case features, but other nominal features such as gender and number only in exceptional cases.
Once it has been decided which arguments and predicative complements belong to a certain verb, it can be checked whether the subject NP and the verb agree in person and number. If the verb takes an adjectival complement, it can be checked whether the subject and the complement agree in gender and number.
Since the case features of the morphological tags are used in the parse, it might seem that the parser cannot be used in detecting errors in the case government of verbs, for instance. However, the case feature is mainly used to distinguish between subjects (bearing nominative case, except with certain verbs of which we have a list as mentioned above) and (direct and indirect) objects (bearing oblique case, i.e. accusative, dative, or genitive). Thus, the distinction between the three oblique cases is not crucial for the parse. Most case errors that we find in texts involve some mixup between the oblique cases, rather than between the nominative and one of the oblique cases. Hence, the parser can be used in detecting such errors.
Of course, full parsing, with pronoun resolution etc., would enable us to detect more grammatical errors than our shallow parsing. However, we feel confident that our method will bring us a long way towards useful NLP tools for Icelandic, due to the linguistic richness of the morphological tags and the syntactic annotation scheme.

3. Technical aspects

In this section, we discuss some technical aspects of IceParser. Our aim is to provide arguments for various technical design and implementation decisions, i.e. with regard to the development methodology used, the utilisation of the lexical analyser generator tool JFlex, optimisation, and the integration of IceParser into our NLP toolkit.

3.1. Development methodology

At the very beginning of the parsing project, we decided to base IceParser on the incremental finite-state approach, in which a parser comprises a sequence of finite-state transducers (Grefenstette, 1996; Abney, 1997). The purpose of the transducers is to add syntactic information into running text in an incremental manner. This method is sometimes referred to as the *constructive* approach to distinguish it from the *reductionist* approach by (Koskenniemi et al., 1992) .
The reason for selecting this approach was mainly threefold. First, no treebank exists for Icelandic, and using a data-driven parsing method was therefore not an option. Secondly, earlier incremental finite-state parsing work has proven successful for various languages, e.g. Spanish (Molina et al., 1999), Swedish (Megyesi and Rydin, 1999), German (Müller, 2004), and French (Aït-Mokhtar and Chanod, 1997). Lastly, parsers built using finite-state methods are usually robust and fast, because they are, in fact, just a pipeline of lexical analysers.
Incremental finite-state parsers are developed by specifying patterns, in the form of regular expressions, for matching specific syntactic constructions (substrings) in the input

[1]Strictly speaking, our annotation scheme is independent from IceParser, i.e. the scheme is designed with a general partial parser in mind.

text. Syntactic markers or labels can then be inserted into the input text by associating an action with the appropriate pattern. The syntactic patterns are usually handwritten and thus often demand both computer science knowledge (with regard to regular expressions and finite automata) and linguistic knowledge (of the language being parsed). In fact, our team consists of a computer scientist and a linguist.

Input to IceParser is POS tagged text, using the tagset mentioned in Section 2.1. The parser consists of two main components: a phrase structure module (13 transducers) and a syntactic functions module (9 transducers). The purpose of the modular architecture "is to facilitate the work during development, to allow different uses of the parser and to reflect the different linguistic knowledge that is built into the parser" (Megyesi and Rydin, 1999). In both modules, the output of one transducer serves as the input to the following transducer in the sequence.

Table 1 lists the 22 transducers (in the order in which they are executed) along with a short description of their purpose. The transducers in the upper half of the table belong to the phrase structure module, and the ones in the bottom half belong to the syntactic functions module. Please refer to (Loftsson and Rögnvaldsson, 2007a; Loftsson, 2007b) for a detailed description of all the transducers used in IceParser.

3.2. Utilisation of JFlex

The Xerox Finite-State Tool (XFST) (Karttunen et al., 1996) is often used to develop finite-state parsers (cf. (Megyesi and Rydin, 1999; Aït-Mokhtar and Chanod, 1997)). The XFST includes extensions to the standard regular expression calculus, which simplify the creation of finite-state transducers for syntactic processing. When using the XFST for parser development, the development team defines syntactic patterns, in the form of extended regular expressions, which are then compiled into finite-state transducers. When running the resulting parser (i.e. the set of transducers) on input text, the transducers are interpreted by a run-time engine built into the tool.

We decided not to use the XFST for the development of IceParser. There are mainly two reasons for this decision. First, since the transducers are interpreted by the XFST, its run-time engine needs to be distributed to all parties interested in using the parser. Hence, licensing issues may complicate the distribution of the parser. Secondly, we wanted our parser to be an integrated part of our NLP toolkit (Loftsson and Rögnvaldsson, 2007b), all modules of which are written in Java. In order to simplify the integration of IceParser into the toolkit, and to simplify distribution of the parser, we thus decided to write the parser in a utility which produces Java code.

Our parser is written using the lexical analyser generator tool JFlex (http://jflex.de/). Each transducer is written in a separate specification file, which is compiled into Java code using JFlex. The resulting Java code is a deterministic finite-state automaton, along with actions (Java code) to execute for each matched pattern. The purpose of the actions is to insert syntactic labels into substrings of the input text. The patterns for each transducer are written using the regular expressions language of JFlex. The only non-standard operator of JFlex is $\sim a$, which matches everything up to (and including) the first occurrence of the input matched by a.

As an illustration of the rule and action format of JFlex, consider the following example, taken from the Phrase_MWEP1 transducer which recognises specific MWEs consisting of the preposition *fyrir* followed by specific adverbs:

```
%{
  String Open=" [MWE_PP ";
  String Close=" MWE_PP] ";
%}

AdverbPart = {WS}+{AdverbTag}
PrepPart = {WS}+{PrepTag}

Pair = [fF]yrir{PrepPart}(aftan|austan
       |framan|neðan|norðan|ofan|sunnan
       |utan|vestan){AdverbPart}
%%
{Pair} {out.write(Open+yytext()+Close);}
```

The code included in %{ and %} is copied directly into the generated Java source code.

Two regular definitions[2], *AdverbPart* and *PrepPart*, define the adverb part and the preposition part of the <preposition, adverb> pair, respectively. For example, the adverb part consists of one or more white spaces ({WS}+) followed by an *AdverbTag*. *AdverbTag* is a name defined in a special file, which is included by most of the transducers (*PrepPart* is defined similarly):

```
AdverbTag = aa[me]?{WS}+
```

i.e. the letters *aa* optionally followed by the letters *m* or *e* and postfixed with one or more white spaces. Finally, the name *Pair* is defined as the preposition *fyrir* followed by specific adverbs.

Actions are included inside curly brackets. Thus, when the generated lexical analyser recognises the pattern *Pair* the action is simply to put the appropriate brackets and labels around it (obtained by the function call yytext()). For example, for the MWE *fyrir aftan* 'behind' the result is:

(5) [MWE_PP fyrir <u>ao</u> aftan <u>aa</u> MWE_PP]
(*ao* and *aa* are the POS tags denoting preposition and adverb, respectively.)

Note that the action described above is a simple implementation of the *left to right longest match markup* replace operator, described in (Karttunen et al., 1996). This operator is a part of the XFST, in which it is specified by using the following syntax:

```
  A @-> B ... C
```

A transducer using this operator then inserts the strings (markers) B and C around the longest string matched by regular expression A.

[2]Regular definitions are a sequence of definitions of the form: d_i -> r_i, where each d_i is a distinct name and each r_i is a regular expression which may refer to names $d_1 \ldots d_{i-1}$.

Name	Purpose
Phrase_MWE	Marks MWEs: common bi- and trigrams.
Phrase_MWEP1	Marks MWEs: specific <preposition, adverb> pairs.
Phrase_MWEP2	Marks MWEs: specific <adverb, preposition> pairs.
Phrase_AdvP	Marks adverb, conjunction, and interjection phrases.
Phrase_AP	Marks adjective phrases.
Case_AP	Adds case information to adjective phrases.
Phrase_APs	Groups together a sequence of adjective phrases.
Phrase_NP	Marks noun phrases.
Phrase_VP	Marks verb phrases.
Case_NP	Adds case information to noun phrases.
Phrase_NPs	Groups together a sequence of noun phrases.
Phrase_PP	Marks prepositional phrases.
Clean1	Corrects special kind of annotation errors.
Func_TIMEX	Marks temporal expressions.
Func_QUAL	Marks genitive qualifiers.
Func_SUBJ	Marks subjects.
Func_COMP	Marks complements.
Func_OBJ	Marks direct objects.
Func_OBJ2	Marks indirect objects.
Func_OBJ3	Marks dative objects of complement adjective phrases.
Func_SUBJ2	Marks “stand-alone” nominative noun phrases.
Clean2	Cleans up, e.g. superfluous white spaces.

Table 1: A brief description of all the transducers.

3.3. Optimisation

In the first version of IceParser (and presumably in most incremental finite-state parsers), the output file of one transducer is used as the input file to the next transducer in the sequence. This version processes about 14,900 word-tag pairs per second (running on a Dell Precision M4300, Intel Core 2 Duo CPU, 2.2 GHz).

Note that, by writing the parser in Java/JFlex, we have full control of the source code. This has enabled us to build an optimised version of IceParser. In the optimised version, instead of making the transducers read and write to files, we make them read from, and write directly to, memory.

The following Java function *parse* illustrates how the output of one transducer is used as input to the next transducer in the sequence, without reading and writing to files.

```
0) parse(String text) {
...
1) StringReader sr=new StringReader();
2) StringWriter sw=new StringWriter();
3) advp=new Phrase_AdvP(sr);
4) ap=new Phrase_AP(sr);
5) advp.yyreset(sr);
6) advp.parse(sw);
7) sr=new StringReader(sw.toString());
8) sw=new StringWriter();
9) ap.yyreset(sr);
10)ap.parse(sw);
...
}
```

Line 0) is the signature of the function *parse*, which is called once for every line in a file containing the text to be parsed. Lines 1), 2), 7), and 8) create instances of the *StringReader* and *StringWriter* Java classes. Lines 3) and 4) create instances of the Phrase_AdvP and Phrase_AP classes, whose source files were automatically created by the JFLex tool from a corresponding regular expression specification file (as discussed in Section 3.2.). Lines 5) and 9) reset the corresponding lexical analyser to read from a new input stream. Lines 6) and 10) call the parse method in the corresponding transducers (see below). Moreover, lines 6), 7), and 9) show how the output of the Phrase_AdvP transducer is used as input to the Phrase_AP transducer.

The *parse* method of the transducers tries to match the input to its patterns and carry out the associated action. Its implementation is simple:

```
parse(java.io.Writer _out)
{
  out = _out;
  while (!zzAtEOF)
      yylex();
}
```

Note that methods starting with the letters *yy* (like yyreset() and yylex()) and variables starting with the letters *zz* (like zzAtEOF) are automatically generated by the JFlex tool. The *out* variable, which is an instance of the Writer class, is used in the actions to transduce the output (as shown in the last line of the code example for the Phrase_MWEP1 transducer in Section 3.2.).

This optimised version of IceParser annotates the whole POS tagged IFD corpus, consisting of 590,297 word-tag pairs (36,922 sentences), in just over 23 seconds. This is equivalent to about 25,200 word-tag pairs per second, resulting in a speed increase of about 70% compared to the first version of the parser.

3.4. Integration into our NLP toolkit

Since IceParser is written in Java, we were able to integrate it easily into our NLP toolkit, *IceNLP*. The toolkit works in the following manner. First, the input text is tokenised. Secondly, sentence segmentation is carried out. Thirdly, POS tagging for each input sentence is performed (using *IceTagger*, a linguistic rule-based tagger (Loftsson, 2007a)), and, lastly, each POS tagged sentence is partially parsed with IceParser.

The optimised version of IceParser is used in IceNLP. A POS tagged sentence is not written to an output file, but is instead fed directly to IceParser in the manner described in Section 3.3. The result is an efficient combined POS tagging and partial parsing utility[3].

4. Conclusion

In this paper, we have described the linguistic richness and the technical aspects of IceParser, an incremental finite-state parser for Icelandic. We discussed the linguistic richness of the output generated by the parser and argued that for many simple sentences the output amounts to full parsing. Additionally, we described technical aspects of the parser, in particular, various design and implementation details. Our description may be used as guidelines for other researchers developing similar parsers.

5. Acknowledgements

Thanks to the Institute of Lexicography at the University of Iceland, for providing access to the IFD corpus used in this research.

The development of IceParser was partly supported by the Icelandic Research Fund, grant 060010021, “Shallow parsing of Icelandic text”.

6. References

S. Abney. 1996. Part-of-Speech Tagging and Partial Parsing. In K. Church, S. Young, and G. Bloothooft, editors, *Corpus-Based Methods in Language and Speech*. Kluwer Academic Publishers.

S. Abney. 1997. Partial Parsing via Finite-State Cascades. *Natural Language Engineering*, 2(4):337–344.

S. Aït-Mokhtar and J.-P. Chanod. 1997. Incremental Finite-State Parsing. In *Proceedings of Applied Natural Language Processing*, Washington DC, USA.

G. Grefenstette. 1996. Light Parsing as Finite State Filtering. In *Proceedings of the ECAI ’96 workshop on “Extended finite state models of language”*, Budapest, Hungary.

F. Karlsson, A. Voutilainen, J. Heikkilä, and A. Anttila. 1995. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin, Germany.

L. Karttunen, J.-P. Chanod, Grefenstette, G., and A. Schiller. 1996. Regular expressions for language engineering. *Natural Language Engineering*, 2(4):305–328.

K. Koskenniemi, P. Tapanainen, and A. Voutilainen. 1992. Compiling and using finite-state syntactic rules. In *Proceedings of the 14th International Conference on Computational Linguistics*, Nantes, France.

X. Li and D. Roth. 2001. Exploring Evidence for Shallow Parsing. In *Proceedings of the 5th Conference on Computational Natural Language Learning*, Toulouse, France.

H. Loftsson and E. Rögnvaldsson. 2006. A shallow syntactic annotation scheme for Icelandic text. Technical Report RUTR-SSE06004, Department of Computer Science, Reykjavik University.

H. Loftsson and E. Rögnvaldsson. 2007a. IceParser: An Incremental Finite-State Parser for Icelandic. In *Proceedings of NoDaLiDa 2007*, Tartu, Estonia.

H. Loftsson and E. Rögnvaldsson. 2007b. IceNLP: A Natural Language Processing Toolkit for Icelandic. In *Proceedings of Interspeech 2007, Special Session: “Speech and language technology for less-resourced languages”*, Antwerp, Belgium.

H. Loftsson. 2007a. Tagging Icelandic Text using a Linguistic and a Statistical Tagger. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, Rochester, NY, USA.

H. Loftsson. 2007b. *Tagging and Parsing Icelandic Text*. Ph.D. thesis, University of Sheffield, Sheffield, UK.

B. Megyesi and S. Rydin. 1999. Towards a Finite-State Parser for Swedish. In *Proceedings of NoDaLiDa 1999*, Throndheim, Norway.

A. Molina, F. Pla, L. Moreno, and N. Prieto. 1999. APOLN: A Partial Parser of Unrestricted Text. In *Proceedings of SNRFAI99*, Bilbao, Spain.

F-H. Müller. 2004. Annotating Grammatical Functions in German Using Finite-State Cascades. In 20th *International Conference on Computational Linguistics*, Geneva, Switzerland.

J. Nivre. 2002. What kinds of trees grow in Swedish soil? A Comparison of Four Annotation Schemes for Swedish. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories*, Sozopol, Bulgaria.

J. Pind, F. Magnússon, and S. Briem. 1991. *Íslensk orðtíðnibók [The Icelandic Frequency Dictionary]*. The Institute of Lexicography, University of Iceland, Reykjavik, Iceland.

E. Rögnvaldsson. 2006. The Corpus of Spoken Icelandic and its Morphosyntactic Annotation. In *Treebanking for Discourse and Speech. Proceedings of the NODALIDA 2005 Special Session on Treebanks for Spoken Language and Discourse*, Copenhagen, Denmark.

H. Thráinsson. 2007. *The Syntax of Icelandic*. Cambridge University Press, Cambridge.

A. Voutilainen. 1997. Designing a (Finite-State) Parsing Grammar. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*. MIT Press.

[3]Please visit `http://nlp.ru.is` for tagging and parsing Icelandic text.

Partial Parsing in a Simple MT System

Petr Homola, Vladislav Kuboň

Institute of Formal and Applied Linguistics
Malostranské náměstí 25
CZ-110 00 Praha 1
Czech Republic
{homola|vk}@ufal.mff.cuni.cz

Abstract

The paper describes an architecture for a hybrid approach to the machine translation of closely related languages. It learns from the previous experiments performed for closely related Scandinavian, Slavic, Turkic and Romanic languages. The main part of the paper introduces a rule-based shallow parsing component for Czech which tries to analyze parts of input sentences. It thus solves to a certain extent the problem of morphemic ambiguity of Czech word forms. All results of the shallow parser are translated into the target language (Slovak) and a ranker based on a stochastic model of Slovak chooses the best translation. The results presented and discussed in the last section of the paper indicate better translation quality compared to the existing MT system for the same language pair.

1. Introduction

Shallow parsing is a term covering a wide variety of techniques, whose aim is to deliver a partial syntactic structure of an input sentence in an efficient way. It has been used for various application areas, including information retrieval, document search, dialogue systems etc. All applications have one thing in common — they aim at exploiting the shallow parsing for isolating syntactically relevant units using relatively simple means. In this paper we would like to address the issue of exploiting a shallow parsing formalism in a hybrid machine translation system with statistical post-processing.

1.1. MT between related languages

Although the automatic translation of closely related languages is a subject considered by many linguists as slightly inferior compared to the full-fledged MT of unrelated language pairs, it is at the same time a stimulating field providing a number of interesting research topics. It has been investigated recently for numerous language groups — for Slavic langauges in (Marinov, 2003) and (Homola and Kuboň, 2004), for Scandinavian languages in (Dyvik, 1995), for Turkic languages in (Altintas and Cicekli, 2002) and for languages of Spain in (Corbi-Bellot et al., 2005).

The MT of closely related languages not only provides a translation quality unattainable by neither stochastic nor rule-based general MT systems (cf. the results presented in (Hajič et al., 2003) or in (Corbi-Bellot et al., 2005), with a correct translation of about 90% of the text in both cases), it may also serve as a kind of testing ground for methods which may be exploited in full-fledged MT systems.

The MT systems for closely related language mentioned above are all based on "shallow" methods. The close relatedness of languages from the same language group guarantees that there are usually only minor syntactic differences which can be handled by shallow parsing of a source language or by "shallow transfer", a method used in (Corbi-Bellot et al., 2005). Let us present some linguistic phenomena where shallow parsing represents an adequate tool solving the translation problems. The concrete examples are taken from Balto-Slavic langauges, a language group which has proved to be suitable for experiments with MT between closely related languages.

1.1.1. Word order problems

Shallow parsing may play an important role in solving the problem of a different word-order in Slavic nominal groups. With congruent attributes, Russian, Czech and Slovak prefer in most cases the order <Adj N>, adjective noun, while Polish typically uses the order <N Adj> for adjectives defining a "species" of the nominal head, while the order <Adj N> is reserved for adjectives defining a "feature" of the noun. This problem requires word-order adjustments during the translation. The knowledge of what actually belongs to the nominal group, provided by a shallow parsing module, is very important, although it helps only partially. The general solution of this problem is very complicated; full solution would require even semantic analysis of the source text, a phenomenon definitely beyond the intended scope of a simple MT system exploiting the similarity of related languages.

1.1.2. Differences in the system of genders

Unlike most of the Slavic languages, their close relative Lithuanian language (Baltic langauges have similar syntactic properties as Slavic langauges and thus they are likely candidates for "shallow MT") does not have the neuter gender. This poses no problem for nouns, since it can be handled easily by the bilingual dictionary, but it is a serious problem for adjectives and adjectival pronouns that syntactically depend on a noun in neuter gender in the source language. If the text is translated word for word (i.e., no shallow parsing is used), no dependencies between words are created. If a neuter noun with depending adjectives occurs in the source sentence, the morphological tag specifying gender is changed only for the noun. All adjectives keep their morphological tags unchanged and thus they are a source of translation errors. When a shallow parser module is employed, most occurrences of this problem are solved by the shallow syntactical analysis of noun phrases.

1.1.3. Prepositional constructions

Even between the closely related languages it is very often the case that some prepositions require a different case than their counterparts in the target language. For example, the Czech preposition *pro* [for] (and the Slovak preposition *pre*) requires the use of the accusative case, while the corresponding Polish preposition *dla* (and the Russian preposition для) requires the genitive case. The shallow parser helps to identify the whole prepositional phrase and thus it is possible to adapt the case of the nominal head of the phrase to the case required by the preposition. The parser may be able to help even with more complicated phenomenon when a preposition used in the source language is not used in the target one and it's function is expressed by a case of the nominal head of the phrase.

1.2. Named entity recognition

Named entities (NE) are atomic units such as proper names, temporal expressions (e.g., dates) and quantities (e.g., monetary expressions). They occur quite often in various texts and carry important information. Hence, proper analysis of NEs and their translation has an enormous impact on MT quality (see (Babych and Hartley, 2004)).

NE translation involves both semantic translation and phonetic transliteration. Each type of NE may be handled in a different way. For instance, person names should not undergo semantic translation (only transliteration is required), while certain titles and part of names should (e.g. for Czech and English *first lady Laura Bush* → *první dáma Laura Bushová*). In case of organizations, application of regular transfer rules for NPs seems to be sufficient (e.g. again for Czech and English, *Ústav formální a aplikované lingvistiky* → *Institute of Formal and Applied Linguistics*), although an idiomatic translation may be probably preferable sometimes.

In the following paragraphs we are going to present an approach solving many of the above mentioned issues. It is based on regular expressions that process typed feature structures. The grammar framework, similarly as the formally weaker platform SProUT (Bering et al., 2003), uses finite-state techniques and unification, i.e., a grammar consists of pattern/action rules, where the left-hand side is a regular expression over typed feature structures (TFS) with variables, representing the recognition pattern, and the right-hand side is a TFS specification of the output structure.

2. An architecture of the MT system

As Czech and Slovak are very similar at all linguistic levels, the architecture of the system differs from most rule-based or hybrid MT systems. In particular, no full-fledged analysis is needed. A full syntactic analysis cannot be done with sufficient precision as for now, and the errors we would introduce by trying to create a full syntactic tree for the sentence would lower the quality of the translation significantly (as reported in (Oliva, 1989) for a Czech-to-Russian MT system). Thus we have adopted the simplistic and rather naïve approach of ignoring syntactic differences and focusing on morphology and lexics. Nevertheless, since Czech is a language with rich inflection, which implies a very high degree of morphological ambiguity, it seemed helpful to integrate a partial ('shallow') parser into the system. This module became a must-have component as we have decided not to use the stochastic tagger in the system because it causes too many errors that have a negative impact on the subsequent phases of the translation process. The main goal of the partial parser is to restrict the ambiguity of morphological annotation by using local context. For example, the adjective *hlavní* "main" has 27 different tags in Czech which would result in many different forms in Slovak (*hlavný, hlavná, hlavné, hlavní* etc.) because the target language does not exhibit the same degree of case syncretism of soft adjectives. Fortunately, this immense ambiguity can often be constrained if the adjective is followed by a noun that governs it. Due to the agreement, only the intersection of possible tags of both words is valid in the given local context.

The partial analysis is followed by lexical transfer. In this phase, lemmas of all words are translated to their Slovak equivalents according to a bilingual dictionary. This dictionary contains no additional morphological information. Since the partial parser produces complex syntactic structures, they have to be linearized so that we obtain the target sentence. The linearization is a reverse process of parsing which means that the complex structures are decomposed while preserving the original word order in the source language. Finally, all words are morphologically processed (word forms are generated from lemmas and tags).

In the following subsections, we describe the components and data structures they use in detail.

2.1. Feature structures

In the system, the basic data structure for representing linguistic data is a feature structure. It is an attribute-value-matrix (AVM); the values of its attributes are atoms, strings or complex values (sets or embedded feature structures). All feature structures in the system are typed, i.e., there is a global type hierarchy and each feature structure is assigned a type, for instance:

(1) $\begin{bmatrix} \textit{adv} & \\ \text{LEMMA} & \text{`quickly'} \\ \text{POS} & \text{adv} \end{bmatrix}$

Each linguistically significant entity has a set of relevant features. The value of a feature may be underspecified, i.e., its value may not be fully known until a more specific context is given (e.g., the morphological analyzer classifies the word *IBM* as a noun but specifies neither number nor case for it). Ambiguous feature values usually get resolved after having taken context into account.

The most typical operation on feature structures is unification which is a combination of mutually compatible attribute values. What is often used in the rules is a partial unification, i.e., only specified attributes are unified (e.g. *case, gender, number*), which reflects the linguistic notion of agreement between a head and some of its dependents.

2.2. Chain graphs

The basic data structure that represents text segments and their local contexts is a chain graph. A chain graph is a continuous graph with designated initial and end nodes. It represents all hypotheses that are valid up to a certain point in the parsing process. The application of syntactic rules is implemented by adding new edge to the graph. For example, implementing the rule of an adjective that depends on a subsequent noun and agrees with it in gender, number and case, means finding two adjacent edges in the chain graph (for the adjective and its governing noun, respectively) and adding a new edge that spans the found edges, if both words agree in the required attributes. The original edges are marked as used by a rule which means that they will be removed from the graph after the application of all possible rules. At the end of the parsing process, the remaining edges represent a partial parse of the source segment (the parsing process is described in detail in (Colmerauer, 1969)).

There are some simple workarounds that allow for a more effective processing of chain graphs like reducing morphological ambiguity by means of shackles, removing of falsified hypotheses etc. (see below).

The initial chain graph of the Czech sentence “I thought that I would be a writer” is a linear sequence of edges labelled with morphologically annotated words of the source sentence. After the application of grammar rules, the chain graph is extended with new edges (see Figure 2). The last step is to remove deprecated edges, i.e., all edges that were spanned over are deleted from the graph. Furthermore, all edges that do not belong to a path from the initial node to the end node are removed as well. As one can easily see in Figure 2, there is only one remaining edge which spans the whole sentence.

The use of chain graphs bears one specific problem. Since we do not aim to parse whole sentences, the result of the parser usually are not long edges from the initial node to the end node, but rather edges that cover simple noun and prepositional phrases. If such edges overlap and there is no other edge that would cover both of them, there is no path in the graph from the initial graph to the end graph, resulting in an empty graph. This is an inherent property of the formalism, described in more detail in (Colmerauer, 1969). We do not solve this problem explicitly, suggesting to translate such sentences once more with the shallow parser switched off.

2.3. Rules

2.3.1. The structure of the rules

The grammar for analysis and synthesis consists of declarative rules that prescribe how to combine phrases to complex structures or how to decompose them. In our system, all rules are context-free.

A rule can be applied if its right-hand side matches the categories of a subchain in the chain graph and all conditions associated with the rule are met. The conditions are defined by means of unification over the associated feature structures and/or their attributes (which can be atomic values or recursively embedded features structures). In such a case, a new edge or a subchain of new edges is added to the chain graph which spans the edges that are covered by the right-hand side of the rule. The feature structure the new edge is labelled with is usually based on one of the feature structures of the covered edges and extended by means of unification according to the conditions associated with the rule (an exception may be, for example, a feature structure for coordination).

2.4. NP/PP rules

NP/PP rules are used to identify simple noun and prepositional phrases and their internal syntactic structure. A simple but very frequent example may be the combination of an adjective and a noun that are adjacent and agree in gender, case and number.

Rules used for partial (‘shallow’) analysis do not usually reflect the relationship (mainly dependencies) between the main verb and its complements. Such techniques are used for instance for named entity recognition. Here is an example of a simple NP/PP rule[1]:

(2) PP → P **NP**, $\uparrow$ CASE = $\downarrow$ CASE & $\uparrow$ PREP = $\downarrow$

```
(pp -> p np / R (
  (^ (case) = . (case) )
  (^ (prep) = . )
))
```

The rule attaches a preposition to a noun phrase. The first part (before the comma) declares the categories of the subchain the rule will be tentatively applied to. The bold font denotes that the feature structure of the right element will be propagated as the head (the core of the feature structure) of the phrase. It takes a preposition and a noun phrase to the right of it that agree in case which is declared in the other part of the rule — the conditions. Thus the resulting feature structure is the feature structure of the noun phrase extended with a new attribute — *prep* — which is unified with the feature structure of the preposition.

2.5. Helper rules

The remaining rules concern mainly verb phrases. However these rules typically have no linguistic motivation, their goal is to constrain morphological ambiguity by using local context. For example, a noun phrase adjacent to certain verb forms may sort out some of the morphological tags. This approach is obviously not bullet-proof, sometimes such a rule applies although both words are not related to each other. Nevertheless in most cases, it helps to constrain morphological ambiguity. For example, in the sentence

(3)

Auta	*jezdila*
cars-NEUT,PL,NOM	go-LPART,IMPF,NEUT,PL
rychle.	
quickly	

“The cars were moving quickly.”

[1]We use the LFG notation ((Bresnan, 2002)) although the rules are interpreted in a slightly different way (see below).

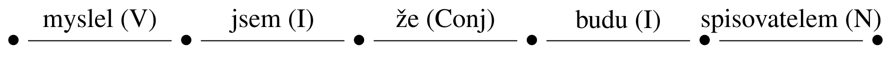

Figure 1: Initial chain graph

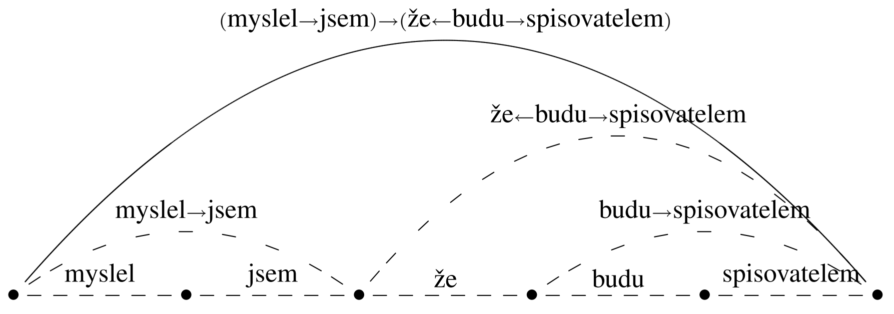

Figure 2: Chain graph with new edges

the word *jezdila* can be translated as *jazdila* or *jazdili* if the context is unknown since the form of the *l*-participle is different in Slovak for fem.pl and neut.sg whereas there is a syncretism in Czech. Assuming that the preceding noun (*auta*) is the subject of the verb form helps to sort out irrelevant tag for both words — for the noun, it is sg.gen and pl.acc (the correct tag is pl.nom).
In general, the result of the parsing process up to here is a set of features structures that cover continuous segments of the input sentence. In a full-fledged MT system, the result would be a set of feature structures that cover the whole sentence.

2.5.1. Named entities

The recognition of named entities is an additional module that helps to constrain morphological ambiguity. It is implemented in the same formalism as described above, however most rules have a clear semantic meaning.
In the sequel, we give an example of a grammar fragment which is capable of recognizing Czech date expressions of the format *12. listopadu 2006* "November 12th, 2006". The first (unary) rule marks a feature structure as being semantically significant (in the

(4) MONTH → **NP**, ↑ LEMMA = 'listopad' &
↑ MONTH = '11'

(5) ORD → **CARD** NOTE, ↓ LEMMA = fullstop

(6) DATE → ORD **MONTH**, ↑ DAY = ↓ lemma &
↑ CASE = gen

(7) DATE_WITH_YEAR → **DATE** CARD, ↑ YEAR =
↓ lemma

The symbol CARD denotes cardinal numbers, ORD denotes ordinal numbers. These categories come from a preprocessing phase which takes place between morphological analysis and parsing. The result of the parser would be the following feature structure (we give only semantically relevant features):

(8)
$$\begin{bmatrix} \textit{date\_with\_year} & \\ \text{DAY} & 12 \\ \text{MONTH} & 11 \\ \text{YEAR} & 2006 \end{bmatrix}$$

NB: A full semantical corectness would, of course, require to check the number whether it is in the appropriate range according to the month. We assume that the semantics of named entites is correct in the text, i.e., the rules only cover syntax.
In simple cases, for each lemma of the source language the bilingual glossary contains its counterpart in the target language, typically with additional morphological information (such as gender which may differ in both languages). Furthermore, the glossary contains structural transfer rules for phraseologisms etc. Here is an example of a simple lexical entry:

```
(t_vp -> vp (
  (^ (lemma) ! 'come')
  (^ (t_lemma) = 'kommen')
))
```

3. Principles of the partial rule-based syntactic parsing

In this section, we describe the parser from a more general perspective than it is used in the presented system. The power of the parser component has also been tested in our recent experiments with the Czech-to-German language pair and it turned out that it is also capable of performing deep syntatic analysis (including valence).

3.1. Tasks of the parser

The main task of the shallow parser in an MT system is to deliver an information about the sentence structure to the transfer module so that language specific structural properties could be handled and transferred properly. Without the parser, morphological differences may only be considered which is, of course, not sufficient for most language pairs. Hence the parser may provide an add-on value which is supposed to improve the target sentence. If the source

sentence is left untouched by the parser (because it is too short or too complex), the system translates it as if there were no parsing module.
The output of the parser is a set of *c*-trees. It is important to mention that a *c*-tree does not represent the structure of the sentence as such but a concrete rule application sequence. What is passed to the transfer module are *f*-structures that are assigned to constituent phrases during the parsing process.
We would like to underline once again that the shallow Czech grammar is not supposed to parse whole sentences. Of course, if the syntactic structure of the sentence is simple enough, the result will be one tree (or a set of trees) covering the whole sentence. Nevertheless in most cases, the result is a set of trees which only represent fragments of the sentence. One reason for such behavior may be non-projectivity which is very frequent in languages with free word order. But projective sentences also may be parsed only partially since the grammar focuses on the level of noun and prepositional phrases. The coverage of verbal phrases is rather small, the rules on this level are meant to capture only the syntactic construction which may cause serious problems in the target sequence.

3.2. The computational formalism

The transformational formalism used is based on a chunk parser. What is very important is the fact that the derivational process is context-free (in the sense of Chomsky's hierarchy) which has the crucial consequence for Slavonic languages that it is not able to handle non-projective constructions (at least not directly).
In the following subsections, we give a brief overview of what should be coped with within the grammar.

3.2.1. Ambiguous input

The input of the parser can be morphologically ambiguous. In such a case, the parser tries to use all available data to construct a complete tree. If it succeeds, all complete trees create the result set whereas all input items which are not contained in a complete tree are discarded.
It is necessary to parse the whole sentence in order to disambiguate it morphologically. Even then, some words may keep more than one morphological tag (due to case syncretism). In case of shallow parsing only, the morphological ambiguity seems to be one of the most serious problems. The best case scenario would be to get an disambiguated input. Unfortunately, at the moment the only possibility is to use a stochastic tagger which introduces too many errors that make it impossible for the parser to recognize important dependencies. It is a general problem of highly inflected languages that their taggers work with lower precision and at the same time it is impossible to disambiguate the input text morphologically by means of shallow rules only (as shown, e.g., for Czech in (Žáčková, 2002)).

3.2.2. Agreement

One of the essential rule principles is the agreement of morphological categories between the governor and its dependent. For example, an adjective which depends on a noun, has to agree with it in gender, case and number. We understand the term *agreement* in broader sense, i.e., a dependent agrees with its governor if a set of conditions which are defined for the particular type of syntactic construcion, is satisfied. In most cases, the conditions are simply equivalences of category values, as in the following phrase:

(9) *mladší* *sestře*
younger-FEM,SG,DAT sister-FEM,SG,DAT
"to the younger sister"

Nevertheless, the conditions may be more complicated sometimes, for instance, in Polish noun phrases if the governor is in dual form[2]:

(10) *czarnymi* *oczyma*
black-NEUT,PL,INS eyes-NEUT,DU,INS
"with black eyes"

(11) *w* *swoim* *ręku*
in REFL-POSS,SG,LOC hands-FEM,DU,LOC
"in his/her hands"

Another example can be found in Russian:

(12) два больших города
two big-MASC,PL,GEN towns-MASC,SG,GEN
"two big towns"

Thus we see that a precise definition of agreement depends on the syntactic type and also on morphological properties. For instance, we say that an adjective agrees with a noun in dual if the following conditions are satisfied:

1. $\text{CASE}_{adj} = \text{CASE}_{noun}$
2. $\text{GENDER}_{adj} = \text{GENDER}_{noun}$
3. NUMBER_{adj} = 'pl' & NUMBER_{noun} = 'dual'

Another example concerning non-trivial agreement between subject and verb (in Spanish or Slovenian):

(13) *los* *checos* *amamos...*
the Czech-MASC,PL like-1SG
"we Czech people like..."

(14) *Slovenci* *volimo...*
Slovenes-MASC,PL like-1SG
"we Slovene people like..."

For this reason, we use predefined patterns to identify agreement between sentence elements.

3.3. Implementation of the formalism

Let us make a couple of short notes on the underlying implementation of the formalism. Both feature structures and rules are written in form of (Lisp-like) *s*-expressions that are automatically trans-compiled to LFG-like rules. The implementation has been inspired mostly by LFG ((Bresnan, 2002)) and the SProUT framework ((Becker et al., 2002)).

[2]There are rests of dual in Polish for pairwise nouns.

Apart from rules used to build syntactic trees, we use in our grammar some rules the aim of which is to modify the chain graph or to control the parsing process. Since the formalism is declarative, the *control rules* use a workaround to achieve a particular modification of the graph. Let us briefly explain at least the most important one of these rules.

As has been already described above, the input of the parser is typically morphologically highly ambiguous and the main task of the parser is to disambiguate the sentence (or at least to reduce its ambiguity). Let us consider the sentence *Starý hrad stojí na kopci* "There is an old castle on the hill". The phrase *starý hrad* is morphologically ambiguous (nominative and accusative). After having recognized this phrase as the subject of the main verb, we know that the case is nominative in this context. And since there is no other reading where it would be accusative, we want to remove this wrong reading. In fact, it is removed automatically by the algorithm of the parser. But what would happen if we had the bare phrase *staré hrady*? There are two possible readings (nominative and accusative) which cannot be resolved without context. Nevertheless, there are still other meanings for each of the words independently (unregarding the dependence between them). In this case, the contextually incorrect edges would not be removed although the parser has analyzed the phrase. This is one negative property of the parser framework which has to be solved explicitly. We use a workaround: we insert a dummy edge (*shackle*) between adjacent edges. If there is at least one analysis which connects two words from adjacent edge bunches, the parser marks the shackle as used, i.e., it will be removed by the system later. As a side effect of this, the 'wrong' edges do not belong to a valid path from the initial node to the end node any more and will be deleted, too.

4. Statistical postprocessing and evaluation

An essential part of the whole MT system is the statistic postprocessor. The main problem with our simple MT process described in the previous sections is that both morphological analyzer and transfer introduce a huge number of ambiguities into the translation. It would be very complicated (if possible at all) to resolve this kind of ambiguity by hand-written rules. That is why we have implemented a stochastic post-processor which aims at selecting one particular sentence that is best in the given context.

We use a simple language model based on trigrams (trained on word forms without any morphological annotation) which is intended to sort out "wrong" target sentences (these include grammatically ill-formed sentences as well as inappropriate lexical mapping). The current model has been trained on a corpus of 18.8 million words which have been randomly chosen from the Slovak Wikipedia[3].

Let us present an example of how this component of the system works. In the source text we had the following Czech segment (matrix sentence):

(15) *Společnost* *ve* *zprávě*
company-FEM,SG,NOM in report-FEM,SG,LOC
uvedla
inform-LPART,FEM,SG

"The company stated in the report, ..."

The rule-based part of the system has generated two target segments:

1. *Spoločnosť vo zpráve uviedli*,
2. *Spoločnosť vo zpráve uviedla*.

The word *uvedla* is ambiguous (fem.sg and neu.pl). According to the language model, the ranker has (correctly) chosen the second sentence as the most probable result.

There are also many homonymic word forms that result in different lemmas in the target languages. For example, the word *pak* means both "then" and "fool-pl.gen", the word *tři* means "three" and the imperative of "to scrub", *ženu* means "wife-sg.acc" and "(I'm) hurrying out" etc. The ranker is supposed to sort out the contextually wrong meaning in all these cases.

We have evaluated the system on approx. 300 text segments from technical and news domain. We use smaller text segments than whole sentences, i.e., we translate matrix and embedded sentences separately for efficiency reasons (the ranker has less sequences to evaluate). The metrics we are using is the Levenshtein edit distance between the automatic translation and a reference translation. There are three basic possibilities of the outcome of translation of a segment.

1. The rule-based part of the system has generated a 'perfect'[4] translation (among other hypotheses) and the ranker has chosen this one.
2. The rule-based part of the system has generated a 'perfect' translation but the ranker has chosen another one.
3. All translations generated by the rule-based part of the system need post-processing.

In the first case, the edit distance is zero, resulting in accuracy equal to 1. In the second case, the accuracy is $1 - d$ with d meaning the edit distance between the segment chosen by the ranker and the correct translation divided by the length of the segment. In the third case, the accuracy is calculated as for (2) except that we use the reference translation to obtain the edit distance.

Given the accuracies for all sentences we use the arithmetic mean as the translation accuracy of the whole text. The accuracy is negatively influenced by several aspects. If a word is not known to the morphological analyzer, it does not get any morphological information which means that it is practically unusable in the parser. Another possible problem is that a lemma is not found in the dictionary. In such a case, the original source form appears in the translation, which of course penalizes the score. Finally, sometimes the morphological synthesis component is not able to generate the proper word form in the target language (due to partial incompatibility of tagsets for both languages). In such a case, the Slovak lemma appears in the translation.

[3]http://sk.wikipedia.org

[4]By 'perfect' we mean that the result does not need any human post-processing.

In our test data, 35% of segments have been translated perfectly. For 12% of segments, the system has generated a perfect translation but the ranker has chosen a different one. In general, the accuracy of the translation is 96.45%. With the original architecture (i.e., using a tagger only), the accuracy is 93.92%. When we left out the parser, the result was 96.10%.

5. Conlusions

The results achieved in the experiments with machine translation between two very closely related languages (Czech and Slovak) described in this paper seem to support the hypothesis that a rule-based shallow parser in combination with a stochastic ranker of the target language sentences generated by the system performs better than the simpler architecture used in previous experiments, which exploited the stochastic tagger and a direct translation of lemmas and morphological tags. In the future we would like to improve the results of the system by solving at least some of the issues mentioned in the previous sections by means of the presented shallow parser and to extend the system to other language pairs of more or less similar languages.

Acknowledgments

The research presented in this paper has been supported by the grant No. 1ET100300517 of the GAAV ČR.

6. References

Kemal Altintas and Ilyas Cicekli. 2002. A Machine Translation System between a Pair of Closely Related Languages. In *Proceedings of the 17th International Symposium on Computer and Information Sciences (ISCIS 2002)*, pages 192–196, Orlando, Florida.

B. Babych and A. Hartley. 2004. Selecting translation strategies in MT using automatic named entity recognition. In *Proceedings of the Ninth EAMT Workshop, Valetta, Malta*.

M. Becker, W. Drożdżyński, H.U. Krieger, J. Piskorski, U. Schaefer, and F. Xu. 2002. SProUT — Shallow Processing with Typed Feature Structures and Unification. *Proceedings of ICON 2002*.

C. Bering, W. Drozdzynski, G. Erbach, C. Guasch, P. Homola, S. Lehmann, H. Li, H.-U. Krieger, J. Piskorski, U. Schaefer, A. Shimada, M. Siegel, F. Xu, and D. Ziegler-Eisele. 2003. Corpora and evaluation tools for multilingual named entity grammar development. *Proceedings of the International Workshop: Multilingual Corpora - Lingusitic Requirements and Technical Perspectives, Lancaster, UK.*

Joan Bresnan. 2002. *Lexical-functional syntax*. New York.

Alain Colmerauer. 1969. Les systèmes Q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur. Technical report, Mimeo, Montréal.

Antonio Corbi-Bellot, Mikel Forcada, Sergio Prtiz-Rojas, Juan Antonie Perez/Ortiz, Gema Remirez-Sanchez, Felipe Sanchez Martinez, Inaki Alegria, Aingeru Mayor, and Kepa Sarasola. 2005. An Open-Source Shallow-Transfer Machine Translation Engine for the Romance Languages of Spain. In *Proceedings of the 10th Conference of the European Association for Machine Translation*, Budapest.

Helge Dyvik. 1995. Exploiting Structural Similarities in Machine Translation. *Computers and Humanities*, 28:225–245.

Jan Hajič, Petr Homola, and Vladislav Kuboň. 2003. A simple multilingual machine translation system. In *Proceedings of the MT Summit IX*, New Orleans.

Petr Homola and Vladislav Kuboň. 2004. A translation model for languages of acceding countries. In *Proceedings of the IX EAMT Workshop*, La Valetta. University of Malta.

Svetoslav Marinov. 2003. Structural Similarities in MT: A Bulgarian-Polish case. http://www.gslt.hum.gu.se/ svet/courses/mt/termp.pdf.

Karel Oliva. 1989. A parser for Czech implemented in Systems Q. Technical report, MFF UK, Prague.

Eva Žáčková. 2002. *Parciální syntaktická analýza (češtiny)*. Ph.D. thesis, Fakulta informatiky Masarykovy univerzity, Brno.

Shallow parsing in sentiment analysis of product reviews

Aleksander Buczyński, Aleksander Wawer

Institute of Computer Science, Polish Academy of Sciences
Ordona 21, 01-237 Warsaw, Poland
Aleksander.Buczynski@uw.edu.pl, axw@ipipan.waw.pl

Abstract

The article discusses a practical application of shallow parsing to sentiment polarity analysis of product reviews in Polish. Examples on how partial parsing can help the task on different levels are presented, ranging from disambiguation between mophosyntactic interpretations with different sentiment polarity, through detection of structures expressing negation or lack of a certain sentimentally polarised property, to capturing idioms. All the stages are expressed and implemented in the same, coherent shallow parsing formalism.

1. Introduction

The article presents an attempt to apply shallow parsing to improve the accuracy of automatic recognition of product review sentiment polarity (Turney, 2002) in Polish. Examples of application on various levels are presented, ranging from disambiguation, through detection of negation, to capturing idioms. All the stages are expressed and implemented in the same, coherent shallow parsing formalism.

Section 2. contains the data overview, including the preprocessing technique. Section 3. briefly introduces Spejd, the shallow parsing formalism and engine used for experiments. Section 4. presents the connection between marking consistency grammar structures and disambiguating sentiment polarity. Section 5. shows how to improve the results by identifying constructions describing negation or lack of a certain sentimentally polarised property. Finally, section 6. describes an attempt to cover idiomatic expressions in the same formalism.

2. Data Overview

The evaluation dataset consists of 4175 product or service reviews downloaded from various Polish e-commerce websites and Internet shops. Reviewed products included books, games, printers, monitors, cameras, phones, cosmetics, tools, holidays.

Each review has a corresponding numeric score (number of stars), assigned by the review's author. Most of the websites have scores ranging from 0 (worst) to 10 (best), but some are based on 5 or 6 point scale. For evaluation purposes, in order to obtain a common, coherent metrics, we decided to rescale all scores into three categories: negative (-1), neutral (0), positive (+1).

As it often happens with data collected from the Internet, reviews were typed in a rather loose manner, sometimes omitting Polish diacrits which futher increases ambiguity of the input on top of the "natural" ambiguity of language. A dedicated procedure has been applied to guess the missing diacrits, which improved the detection ratio of identified positive sentiment words by 5% and the number of negative words by 3%.

As a baseline, we took the bag of words approach, disregarding grammar and word order. We do not account for presence of a particular lexeme, but rather presence of a specific category of lexems. Such an abstraction originates in content analysis systems, most notably the classic General Inquirer (Stone, et al 1966). Lexical categories used in this work include two sets of lexemes (dictionaries): 1580 positive and 1870 negative ones. A string is considered to have a positive/negative sentiment if at least one of its morphosyntactic interpretations belongs to a positive or negative dictionary, respectively.[1] After the addition of missing diacrits, the application of the lexicons resulted in sentiment tags for 19370 words in the reviews (13768 positive and 5602 negative).

Baseline accuracy was calculated by running a C5.0 classifier (a commercial successor of C4.5 (Quinlan, 1993)) on the tagged reviews, taking as input variables the number of positive and negative tags in a review. Such a classifier predicted the sentiment of the reviews with accuracy of 74,9%.

3. Shallow Parsing of Polish

For detecting syntactic structures we decided to use Spejd — a tool for simultaneous morphosyntactic disambiguation and shallow parsing (Przepiórkowski, 2007). The Spejd formalism is essentially a cascade of regular grammars. Unlike in the case of other shallow parsing formalisms, the rules of the grammar allow for explicit morphosyntactic disambiguation statements, independently or in connection with structure-building statements, which facilitates the task of the shallow parsing of ambiguous and/or erroneous input. An example of a simple Spejd rule is:

```
Match: [pos~~prep][base~"co|kto"];
Eval:  unify(case,1,2);
       group(PG,1,2);
```

The rule means: **1)** find a sequence of two tokens such that the first token is an unambiguous preposition, and the second token is a form of the lexeme CO 'what' or KTO 'who'; **2)** if there exist interpretations of these two tokens with the same value of case, reject all interpretations of these two tokens which do not agree in case; **3)** if the above unification did not fail, mark the identified sequence as a syntactic group of type PG (prepositional group), whose syntactic

[1]Therefore it is theoretically possible for the same string to have both positive and negative sentiment.

Table 1: Examples of pairs of words in Polish with different sentiment polarity, which have common forms. Tags :spos and :sneg denote `sentiment` value. Lack of tag means neutral sentiment.

obraz (image)	obraza (insult:sneg)
ok (ok:spos)	oko (eye)
płytka (tile, CD/DVD)	płytki (shallow:sneg)
lub (or)	lubić (to like:spos)
kupa (poo:sneg)	kupić (to buy)
wina (guilt:sneg)	wino (wine)

head is the first token and whose semantic head is the second token.
Although Spejd was originally designed for morphosyntactic disambiguation, it is also highly flexible. Therefore we extended the morphosyntactic tagset with a semantic category (`sentiment`), expressing properties of positive or negative sentiment (`spos` and `sneg` respectively). We called this hybrid approach Sentipejd.

4. Sentiment disambiguation

Since both morphosyntactic tagging and partial constituency parsing involve similar linguistic knowledge, shallow parsing can be a powerful tool for simultaneous morphosyntactic disambiguation, as discussed in (Przepiórkowski, 2007). But different morphosyntactic interpretations often imply also different semantic interpretations, including sentiment polarity (especially when it comes to disambiguating between different base forms). Therefore, a tool for disambiguating between various morphosyntactic interpretations can also help to disambiguate the sentiment polarity of an interpreted unit.
For example, strings like obraz and obrazy can be forms of the word obraza (insult), which has a definitely negative sentiment polarity, as well as obraz (image, painting), with no sentiment connotations at all. Table 1 shows a few more examples of such ambiguities in Polish.
For testing the application of shallow parsing to sentiment disambiguation, we used a preliminary shallow grammar of Polish, developed at the Polish Academy of Sciences, Institute of Computer Science. The grammar is written in the Spejd formalism, allowing to encode structure building and disambiguation in the same rules. It contains 58 rules for syntactic group identification. Among these, the rules identifying noun groups turned out to be particularly useful for sentiment analysis, because of their case unification or strict requirements.
Let us examine the sentence:
Najlepszy obraz uzyskamy, podłączając go do cyfrowego wyjścia karty graficznej.[2]
(The best image we achieve, connecting it to a digital output of graphic card.)
Najlepszy is definitely a superlative form of singular adjective 'good', but can be assigned four possible combinations of case and gender:

- NOM:M1
- NOM:M2
- NOM:M3
- ACC:M3

The string obraz can be:

- either a form of the word OBRAZ (image) in nominative or accusative case,
- or a genitive of the word OBRAZA (insult).

After the grammar identifies "Najlepszy obraz" as a noun group, it enforces case, number and gender unification between the words constituting the group. The result is:

Najlepszy	obraz
~~good:adj:sg:nom:m1:sup:spos~~	image:subst:sg:nom:m3
~~good:adj:sg:nom:m2:sup:spos~~	image:subst:sg:acc:m3
good:adj:sg:nom:m3:sup:spos	~~insult:subst:pl:gen:f:**sneg**~~
good:adj:sg:acc:m3:sup:spos	

Although the interpretations are still somewhat ambiguous (the parser has not yet decided whether the phrase is nominative or accusative), for sentiment analysis it is important that the invalid interpretation of "obraz" as a genitive form of 'insults' has been discarded, therefore removing the only interpretation with negative sentiment polarity, which could lead to a wrong conclusion about the sentiment of the whole review.
Let us consider another example:
Trochę zajmuje mu odczyt płytki.
(Some [time] takes him reading the CD.)
The word "płytki" has eight interpretations, four with the base form PŁYTKA:SUBST:

- SG:GEN:F
- PL:NOM:F
- PL:ACC:F
- PL:VOC:F

and four with base form PŁYTKI:ADJ:

- SG:NOM:M1
- SG:NOM:M2
- SG:NOM:M3
- SG:ACC:M3

Spejd has identified the pronoun "mu" as a NG, and following that — part of the sentence "odczyt płytki" as a NG with a genitive postmodifier. The rule identifying the latter group has decided to discard all non-genitive interpretations for "płytki", correctly leaving only PŁYTKA:SUBST:SG:GEN:F.
On the test data the aforementioned shallow parsing rules allowed to assign correct, unambiguous sentiment tags to 144 semantically ambiguous segments, generating no false positives or negatives. Although comparing to the total number of sentiment tags it may not seem much, one has to take into consideration that the grammar used is indeed very shallow and still in developement.

[2]Diacrits were missing in the original input and have been added in preprocessing.

5. Sentiment Phrases

5.1. Rules extraction

Our rules for sentiment extraction were created semi-automatically with the help of statistical methods of collocation extraction. First, a list of word bigrams with the highest value of Frequency biased Symmetric Conditional Probability (Buczyński, 2006) was created, to find collocations which are both common in the corpora and strongly dependent. A simple heuristics was used to discard proper names from the results — if all occurences of both words forming a collocation started with a capital letter, the pair of words was considered a proper name. Then, the remaining collocations were manually generalised into two kinds of rules — sentiment reversibility and feature extraction.

5.2. Sentiment reversibility

We paid special attention to structures expressing reversion or cancellation of sentiment polarity. Although the work presented here is a pioneering effort for Polish, the problem of recognising phrase level sentiment polarity reversal has been addressed in English (Whitelaw et al, 2005). Several of the rules presented below could be implemented using a window based approach, but the precision of such techniques can be problematic in inflected languages.
For our experiments we used the following types of sentiment modifying structures:

Negation — reversing the polarity as from "polecam" ('I recommend') to "nie polecam" ('I don't recommend'). The example generic rule captures also statements including the optional verb 'to be' (`[base~być]?`), like "nie jest dobry" ('isn't good'):

```
Match: [orth~nie/i]
       [base~być]?
       [sentiment~spos];
Eval:
  word(3, neg:sneg, "nie " base);
```

Nullification — expressing lack of a certain quality or property (usually of negative sentiment), for example "nie mam zastrzeżeń" ('I have no objections') or "zero wad" ('zero defects'). An example of a nullification rule[3] is:

```
Match:
  ([base~"bez|brak|zero|żaden"]
   | [orth~nie/i] [base~mieć])
  [base~żaden]?
  [sentiment~sneg];
Eval: word(2, spos, );
```

The second, optional specification in match (`[base~żaden]?`) serves capturing typical double negative constituents, expressing a single negation. Negative concord is quite common in Polish (Przepiórkowski, 1997), and also in the product reviews, for example "nie miałem żadnego problem" ('I didn't have no problem')[4]

Limitation — a limiting expression tells us that an expression of positive or negative sentiment has only a very limited extend, therefore hinting that the general sentiment of the review is the opposite of the expression. Examples: "jedyny problem" ('the only problem'), "jedyna zaleta" ('the only advantage').

```
Match: [base~"jeden|jedyny|1"]
       [sentiment~sneg];
Eval:
  agree(case number gender,1,2);
  word(2, spos, );
```

Negative modification — an adjective of negative sentiment preceeding a noun of usually positive sentiment, for example "koszmarna jakość" ('nightmarish quality'), "nieprzyjemne doświadczenie" ('unpleasant experience')[5]

```
Match:
  [sentiment~sneg && pos~adj]
  [sentiment~spos && pos~subst];
Eval:
  agree(case number gender,1,2);
  word(2, sneg, );
```

5.3. Feature extraction

Among the captured collocations, many were product specific, like "wysoki kontrast" ('high contrast') or "duży wyświetlacz" ('large display'). We have chosen to ignore these, to make the rules product-independent. However, there seem to be sentiment polarised features that are common for many different products. The following features were included in the rules:

- high/low price,
- high/low quality,
- easy/difficult to use.

5.4. Captured Structures

The set of rules described above captured 1774 structures. Table 2 presents the most common of them. Although the structures provide only less than 10% of all sentiment tags, they often change the polarity of tags, therefore having a significant impact on the results. As shown in Table 3, the structures increased the classification accuracy by up to 2,5% comparing to the baseline bag of words approach, using the same c5.0 classifier.

[3]The rule is very generic and does not force any case requirements or unification. It performed well on the reviews data, but for other aplications it may be more suitable to split the rule into a few more sophisticated ones.

[4]In the early version of the system, double negation used to cancel itself, therefore not giving any improvement over the bag of words approach.

[5]It is worth noting that the structures captured by the opposite of the rule, ie. a positive modifier of a negative subject, are very hard to assign an unambiguous sentiment polarity.

Table 2: Most commonly applied Sentipejd rules.

Value	Count	%
negation of positives	493	27,8%
negation of negatives	341	19,2%
nullification of negatives	320	18,0%
feature: ease of use	147	8,3%
nullification of positives	146	8,2%
limitation of negatives	119	6,7%

6. Idioms

This section deals with multiword entries, including phraseological units and idioms, with non compositional sentiment value. The calculation of sentiment present in such structures in then a matter of accurate and efficient recognition, which can be reduced to a more general question - of how to encode and recognize multiword entries in Polish, an inflected language.

In principle, encoding of multiword expressions for natural language processing falls in two general groups (Moszczyński, 2006): encode them in an existing formal grammar, such as Debusmann (Debusmann, 2004) or use a specialized formalism such as IDAREX (Segond, 1995) or Phrase Manager (Pedrazzini, 1994). A formal grammar approach makes the lexicon of multiword sentiment expressions heavily dependant on a particular grammar, which might make its reusability questionable. The expressive power of such grammars might be largely unused in the context of sentiment analysis. The other method, specialized formalisms, seems more promising, but the overview of existing approaches proves that none of them meets the requirements of Polish. IDAREX, which is a regular grammars based formalism, does not allow for handling expressions that have a very variable word order and allow many modifications. Expressing sentiment-bearing idioms in IDAREX has to include all the possible variations which leads to a description that suffers from overgeneration. Moreover, IDAREX does not support unification. This fact alone renders it unsuitable for any reliable recognition, as Polish requires to enforce agreement between constituents of a phraseological expression. Phrase Manager is not suitable for Polish multiword structures as it enforces membership of such a structure to predefined syntactic classes, which in turn leads to an unnecessary overhead and classes proliferation.

We found out that the sentiment carrying idioms can be conveniently described in the same Spejd formalism as the polarity reversing structures. Examples of commonly used non-compositional sentiment phrases used in dialogs involve popular euphemistic expressions such as ‘have somebody somewhere’:

```
Match: [base~mieć]
       [base~"to|ty|ten" && case~acc]
       [pos~~adv]?
       [base~gdzieś];
Eval:  leave(case~acc, 2);
       set(qub:sneg, , 4);
```

where somewhere is an euphemism for a more abusive “arse”, the negative meaning being only recognizable in the context of the whole expression.

Sentiment carrying non-compositional expressions are very infrequent in the reviews, nevertheless a careful examination revealed several such multiword structures. The two most common examples are presented below:

- ‘Almost makes a great difference’ (meaning: to fail to meet some requirements):

```
Match:
  [base~prawie]
  [base~robić & person~ter]
  [base~"duży|spory|wielki"]
  [base~różnica && case~acc];
Eval:
  unify(case number gender,3,4);
  leave(case~acc, 4);
  word(qub:sneg,
    "prawie robi dużą różnicę");
```

- ‘Nothing to add, nothing to lessen’ (meaning: perfectly, accurately):

```
Match: [base~nic]
       [base~dodać && pos~inf]
       ([pos~interp] ns?)?
       [base~nic]
       [base~ująć && pos~inf];
Eval: word(qub:spos,
        "nic dodać nic ująć");
```

The recognition of sentiment carrying multiword structures or idioms might not introduce substantial improvements on product review sentiment recognition accuracy, as the reviews language is very simple, and idioms are very rare (few occurences per idiom in the sample). However, it seems that the recognition of sentiment-bearing idiomatic expressions can contribute to sentiment analysis in other language domains, like informal dialogs or literary language. Once properly encoded, the same set of rules for idiom recognition can be used across multiple domains. However, introducing new domains may require extending the idiom set.

7. Results

For evaluation purposes, we grouped the shallow parsing rules into four disjoint sets: disambiguation (described in 4.), sentiment reversibility (5.2.), feature extraction (5.3.) and idioms (6.). Table 3 displays impact of each set on the classifier accuracy, including ‘bag of words’ (empty ruleset, baseline approach) and all rules (all sets combined).

On the test data, disambiguation actually lowered the accuracy of the classification. An examination of the results revealed that the disambiguation rules are sometimes too strict, not taking into account errors often made by the reviewers. The most significant improvement was achieved by detecting sentiment reversing structures.

Table 3: Accuracy of c5.0 classifier on reviews, depending on the shallow parsing ruleset used.

Method	Accuracy
Bag of words	74,49%
Disambiguation	74,47%
Reversibility	77,01%
Feature extraction	74,56%
Idioms	74,49%
All rules	77,05%

8. Conclusions and Future Work

In the paper we presented an approach to improve automatic sentiment polarity extraction from noisy and ambiguous product reviews in Polish by shallow parsing techniques. We demonstrated that shallow parsing can affect the accuracy of sentiment classification compared to a baseline bag of words approach. The most significant improvement has been achieved by detecting negation-like structures which reverse sentiment polarity. On less noisy data morphosyntactic disambiguation of the phrases can also help by removing certain sentiment ambiguities. Finally, although idioms with clear sentiment polarity were rare in the data set under consideration, they can be described and recognised in the same formalism. Further reasearch is needed to investigate the usability of the formalism in idioms recognition on different types of corpora.

The work so far focused on considering product review as a whole, assigning general sentiment polarity to a product. It remains an open question how shallow parsing can contribute to extracting attitudes towards specific properties or dimensions of a product.

9. References

A. Buczyński. 2006. Wybrane zastosowania programu Kolokacje do badań lingwistycznych. A. Duszak, E. Gajek, U. Okulska, Korpusy w angielsko-polskim językoznawstwie kontrastywnym. Teoria i Praktyka. Kraków 2006, pp. 427-448.

R. Debusmann. 2004. Multiword expressions as dependency subgraphs. In Proceedings of the ACL 2004 Workshop on Multiword Expressions: Integrating Processing, Barcelona, Spain.

R. Moszczyński. 2006. Formalisms for encoding Polish multiword expressions. Report 994 of IPI PAN (Institute of Computer Science, Polish Academy of Sciences).

S. Pedrazzini. 1994. Phrase Manager: A System for Phrasal and Idiomatic Dictionaries. Georg Olms Verlag, Hildeseim, Zurich, New York.

A. Przepiórkowski, A. Kupść. 1997. Negative Concord in Polish. Research Report 828 of IPI PAN. Institute of Computer Science, Polish Academy of Sciences.

A. Przepiórkowski, A. Buczyński. 2007. Shallow Parsing and Disambiguation Engine. Proceedings of the 3rd Language and Technology Conference, Poznań, Poland.

J. R. Quinlan. 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.

F. Segond, E. Breidt. 1995. IDAREX: Formal description of German and French multi-word expressions with finite state technology. Technical Report MLTT-022, Rank Xerox Research Centre, Grenoble.

P. J. Stone, D.C. Dunphy, M. S. Smith, D. M. Ogilvie. 1966. The General Inquirer: A Computer Approach to Content Analysis. MIT Press.

P. Turney. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. Proceedings of the 40th Annual Meeting of the ACL, pp. 417-424.

C. Whitelaw, N. Garg, S. Argamon. 2005. Using Appraisal Groups for Sentiment Analysis. Proceedings of the 14th ACM international conference on Information and knowledge management. Bremen, Germany.

Why is this Wrong? – Diagnosing Erroneous Speech Recognizer Output with a Two Phase Parser

Bernd Ludwig, Martin Hacker

Chair for Artificial Intelligence (University of Erlangen-Nuremberg)
Haberstraße 2, D-91058 Erlangen
`ludwig@cs.fau.de`, `hacker@cs.fau.de`

Abstract

A major problem of understanding language in spoken dialog systems is to detect recognition errors in the output of a speech recognizer. Such a capability is the basis of implementing repair strategies that allow a dialog system to handle communication about misunderstandings similarly to other clarifications. In this paper we present a two-phase approach that combines chunk and dependency parsing and takes the global syntactic structure of recognizer output into account. This enables us to identify dependencies between chunks and detect syntactical errors caused by word confusions in case dependency constraints are violated. Finally, we apply these diagnostics to dialog modeling and discuss how the resulting error information can be used by clarification strategies.

1. Introduction

A major problem of understanding language in spoken dialog systems is to detect recognition errors in the output of a speech recognizer. While the signal processing community works on improvements of feature extraction algorithms in order to decrease the average word error rate of a speech recognizer, less attention is paid to the linguistic side of errors and how to detect them.
The capability to locate errors in a speech recognizer output (word chain, word lattice, or word confusion matrix) enables a dialog system to engage in clarification dialogs on acoustic misunderstandings. Our paper presents a study how to identify these misunderstandings.
As an example application for this paper, we choose a natural language dialog system for controlling digital equipment for home entertainment, such as TV sets or DVD players. In this domain, a typical utterance is

Die Sparte Serie auf RTL ist ausgewählt.
(The genre series is selected for RTL).

Normally, speech recognizers will misrecognize some words producing hypotheses which may be inconceivable to a human hearer and – in particular – to an automatic dialog system. Figure 1 lists hypotheses for the above utterance. As discussed in detail later, they are ill-formed and every German native speaker will ask a number of questions in order to clarify the misunderstandings.
The contribution of this paper is a parsing algorithm that computes error diagnostics similar to those a native speaker would find. The paper then explains how such diagnostics can be used in a dialog system for clarification sub-dialogs.
The paper is structured as follows: First, we discuss some corpus studies that exemplify the way in which humans communicate about misunderstandings on the acoustic, syntactic, and semantic level. In the section to follow, we report on other computational approaches to localize and identify errors in ill-formed utterances. Then we explain our two-phase approach that combines a chart-based chunk parsing step with a constrained-based dependency step to a global syntactic analysis which results in a data structure containing possible readings of the input or detailed error reports with respect to a given topological model. This section is followed by a worked example. We conclude with a presentation of the system's performance, with a discussion of the results and with an analysis of open issues and an outlook to future work.

2. Clarification of Misunderstandings

Current (commercial) state-of-the-art systems do not implement strategies revealed in corpus analyzes, but rather heuristic approaches to clarifications. Often, feature extraction that is needed for implementing recovery or clarification strategies is difficult to compute. Secondly, the features are unclear themselves. Researchers report different feature sets they used in their studies (e.g. Gabsdil and Lemon (2004), Bohus and Rudnicky (2005), Skantze (2004)). Mostly, the features are not specified semantically, so it is very hard to compare them. Ginzburg (1998) provides an overview of types of clarifications which may follow an utterance.
In contrast to such ad-hoc technical solutions for clarification strategies, our approach is to find operational semantics for these strategies and for algorithmic detection of features that allow to diagnose non-understanding and misunderstanding errors at run-time. In our view, clarification has to be implemented differently and separately on all levels of perception of and reacting to an utterance. A similar approach is discussed in Schlangen (2004).

2.1. Clarification of Acoustic and Syntactic Information

Acoustic and syntactic errors are hard to distinguish in speech recognizer output. Often enough, they are generated during the speech recognition process – either because the speech recognizer did not classify correctly or because the user transgressed the lexical and grammatical limitations of the recognizer's language model. Examples of such utterances are:

Boulevard Bio nimm anschauen (Boulevard Bio take watch) *ich möchte eine komödie laufen werden (I want a comedy run will)*

These types of error are the focus of our paper.

2.2. Clarification of Semantic Information

Another type is misunderstanding on the semantic level:

ich möchte eine komödie auswählen (I want to select a comedy).

ich möchte eine komödie aufnehmen (I want to record a comedy).

Such misunderstandings may be caused by acoustic problems as well, but cannot be detected on syntactic level as both utterances are grammatically perfect. They are only detectable with information about context and therefore beyond the scope of this paper.

2.3. Clarification of Pragmatic Information

The same holds for pragmatic misunderstandings. For clarifications of pragmatics the context or state of the dialog has to be considered in parallel to the state of the application which is addressed in the interaction. Pragmatic misunderstandings are discussed e.g. in Krum et al. (2005) or Ludwig (2006) and are beyond our scope as well.

2.4. Error Classification

In a corpus of spoken user input collected with the EMBASSI dialog system (see Görz and Ludwig (2005)), the utterance

Die Sparte Serie auf RTL ist ausgewählt. (The genre series is selected for RTL).

is recognized as shown in Figure 1. None of the five hypotheses computed by the speech recognizer contains the transcription (see above).

Although some words are recognized correctly, every native speaker of German would not find a hypothesis out of the five to be conceivable as a complete sentence. The speaker would argue that in each hypothesis there are sequences of words that are conceivable for themselves, but along with the other sequences of the hypothesis they cannot be integrated in a way that a hearer can make sense out of the hypothesis. The reason for that observation is that in a correct sentence there are several grammatical functions that have to be fulfilled by certain phrases. To give an example, in the simplest case there must be a verb phrase serving as predicate, a noun phrase serving as subject and (depending on the verb) other noun phrases serving as direct and indirect object. Having identified these, the hearer is able to construct a hypothesis for the meaning of the utterance.

However, in the hypotheses in Figure 1 some of the functions are missing (e.g. a predicate in hypothesis 1 and 2), some are fulfilled more than once (e.g. the predicate in hypothesis 3, 4, and 5), and some are located in unusual positions (e.g. *what* in hypothesis 4). These facts are obstructive for the hearer to understand the hypothesis.

In most dialog systems, there is a work-around for the very complicated problem addressed above: those sequences of words that have meaning in isolation are extracted from the hypothesis and syntactic functions are ignored completely. This approach works well of course when the speech recognizer made no severe errors. However, if an error affects a word that is crucial for the meaning of the utterance (or the underlying intention of the user – such as the non-understanding of the finite verb *ist* as in the example above), the approach fails as a work-around. Its performance degrades further when errors in recognition lead to conflicting information (e.g. VOX versus ARD in hypothesis 5). In such cases, it is hard for the dialog system to take a decision.

3. Related Work

In the literature, the analysis of clarification dialogs has always attracted the interest of many researchers. While there are many corpus studies on which patterns are used for clarification in human-human dialog (Ginzburg (1998) provides an overview of types of clarifications which may follow an utterance), it is difficult to define an efficient and tractable decision procedure for error classification and selection of an appropriate repair strategy. Indeed, there is even no consensus about an adequate feature set. However, researchers report different feature sets they used in their studies (e.g. Gabsdil and Lemon (2004), Bohus and Rudnicky (2005), Skantze (2004)). Mostly, the features are not specified semantically, so it is hard to compare them.

Our approach differs from the cited publications in that we want to classify errors in natural language output of a speech recognizer by employing as much knowledge about language use as possible instead of abstracting immediately to data that involves the (error prone) interpretation of the recognizer output. Another main difference is that the cited approaches lead to an acceptance or rejection of the user utterance as a whole. However, we are interested in finding the type of misunderstanding in order to provide a computational basis for (interactive) clarification also on parts of an utterance.

In the area of speech recognition, confidence scores are defined by an entropy-based measure of confusion in a word graph. While the idea of word confusion graphs goes back at least to Mangu et al. (2000) for the purpose of minimizing the word error rate of a speech recognizer, Hakkani-Tür and Riccardi (2003) report about experiments to use posterior probabilities or posterior entropy as a confidence measure and to localize errors in positions with confidence below a given threshold. In Falavigna et al. (2002) anti models for phones are proposed as another possibility to compute confidence scores.

The approach in this paper takes the main ideas just outlined into account, but anti models are built on a word or even POS basis in order to highlight linguistic instead of acoustic information.

Hogan (1998) aimed to use data-driven methods to build a simple model of grammaticality in language and to compare it with machine translated text with the hope of uncovering ungrammatical sequences of words. The corpus used in this study consisted of about 300,000 words. Our analyses show that this size is not sufficient for providing a stable language model. In addition, HOGAN was interested in finding differences in the distribution of existing trigrams

```
die sparte gelaufen ab elf uhr gewählt      (the genre run from 11 clock selected)
die sparte gelaufen ab elf uhr ausgewählt   (the genre run from 11 clock selected)
die sparte gelaufen das will vox gewählt    (the genre run that want VOX selected)
die sparte gelaufen was wähle vox gewählt   (the genre run what choose VOX selected)
die sparte gelaufen ard wähle vox gewählt   (the genre run ARD choose VOX selected)
```

Figure 1: 5 best hypotheses for a speech signal as generated by a speech recognizer

```
PP  ->  P NP.                        DP  ->  DET NP.
        P:1 position = prepos,               DET:1 kasus = NP:2 kasus,
        P:1 kasrek = NP:2 kasus,             DET:1 numerus = NP:2 numerus,
        PP:0 = P:1.                          DET:1 genus = NP:2 genus,
                                             DP:0 = NP:2.
```

Figure 2: Example chunk rules producing prepositional phrases and definite noun phrases

within the "correct" and the machine translated text. Our focus on the contrary is to identify trigrams that are rare in the language model, but occur in ill-formed input and therefore can serve as indicators for errors.

While conventional parsers are constructed for parsing and completely disambiguating grammatically error free input, our goal is fault-tolerant parsing of sentences that include syntactic errors. For diagnostics the parser must provide useful information about what is wrong. Furthermore the language model should be robust as we are concerned with spoken language as input. However, efficient parsing is essential for applying the method in real time dialog systems. The demand of robustness and the phenomenon of variable word order in German sentences ask for the use of a dependency grammar because other ones like phrase structure grammar are hardly able to deal with those. Another argument in favor of dependency grammars is that error diagnostics call for information about semantic relations provided by a dependency analysis.

There is a number of dependency parsers for German, but none of them meets all demands of error handling, robustness and time complexity. Schröder (2002) and Foth et al. (2004) present a parser that is based on a Weighted Constraint Dependency Grammar (WCDG). Tests show that—even for short sentences—parsing takes too long for real time needs. Regrettably, the parser works case-sensitive, that is it detects nouns only if they are capitalized (in accordance with German spelling). However, this information is not available in spoken language. Without this restriction the parser would even take longer. For every sentence a lot of violations of defeasible grammar constraints are reported. As a whole they may give a rough clue for rating syntactic correctness, but most violations are not constructive for error diagnostics (compare Figure 3). What is more, some of them indicate topological errors. However, these never result from acoustic misunderstandings, but only from the parser choosing a wrong interpretation, provided that the word order of the utterance is correct.

4. The Two-Phase Parsing Approach

The parsing is composed of two phases: In a preprocessing step a chunk parser generates a chart graph containing all possible chunks (compare Figures 2 and 9) on a speech recognizer output, that is a word chain or a word confusion graph. In a second phase a dependency parser searches the

002 : 7.500e-01 : Schlagzeile
006 : 9.950e-01 : mod
008 : 9.950e-01 : mod
010 : 9.950e-01 : mod
012 : 9.970e-01 : Komplementdistanz
008 : 9.999e-01 : direction
010 : 9.999e-01 : direction

Figure 3: Non-constructive violations returned by NATS parser (cf. Schröder (2002) and Foth et al. (2004)) when parsing the first hypothesis from Figure 1.

schmecken (subj(NP_nom))
schmecken (subj(NP_nom),iobj(NP_dat))
schmecken (subj(NP_nom),obj(NP_acc))
schmecken (subj(NP_nom),p-obj(PP_nach_dat))

Figure 4: Readings of the verb *schmecken (to taste)* and their complements

chart graph for a path that can be parsed according to a rule-based linguistic model. If no path can be found that fits the model, it chooses a path as close to the model as possible. The deviation from the linguistic model is described by conflicts caused during the parsing process. This description serves as input to a subsequent diagnostics procedure specified in section 5.

The underlying linguistic model is based on a dependency grammar combined with a topology model. The grammar contains subcategorization rules and lexical data as shown in Figure 4 to describe admissible relations between chunks. These relations are called dependencies. They imply a certain grammatical function between a superordinated chunk, the regent, and a subordinated chunk, the dependent. Unlike common dependency theories that define relations on word level, our approach lifts the relations to a higher level, the chunk level. As chunk graphs contain atomic chunks for every word, all relations between words can be redefined on chunk level. But additionally, partial parses from the chunk parser can be utilized as short cuts for the search process and abstractions within parse trees.

As dependency grammars do not make statements on word order, we need a topology model that describes admissible linearizations of certain parse trees. It is based on topo-

$$
\begin{aligned}
(predicative\_noun, \mathit{VL}, *) &\rightarrow (R, left)\\
(predicative\_noun, \neg\mathit{VL}, finiteVerb) &\rightarrow (\mathit{RB}, right)\\
(predicative\_noun, \neg\mathit{VL}, \neg finiteVerb) &\rightarrow (R, left)\\
(predicative\_noun, \neg\mathit{VL}, *) &\rightarrow (\mathit{VFF}, right)
\end{aligned}
$$

Figure 5: Topology rules specifying target fields for dependents (depending on their grammatical function, the verb order and the regent's properties)

```
 NP_es < NP_ACC : 0
NP_NOM < NP_DAT : 2
```

Figure 6: An absolute and a weighted precedence rule

logical field theory in German linguistics that subdivides every sentence into five fields (see Altmann and Hofmann (2004)). Similarly to Gerdes and Kahane (2001), who suppose that every word (here: chunk) induces a new subfield, we defined a dynamic hierarchical field model that is capable of handling complex phenomena like subordinate clauses, scrambling and partial verb fronting. While target field rules (see Figure 5) specify in which field a certain dependent can be located, precedence rules (see Figure 6) constrain the relative order of chunks within a field. Unlike the common ones, our topology model works interactively since the field boundaries can be extended outwards when a new chunk is assigned to the field. Thus, whensoever a partial parse is to be extended by a new dependency, we can validate that the position of the new dependent fits the topological structure built up by the previous dependencies. The dependency parser works top down using an A* search algorithm to find a path fitting the linguistic model best. For this, it must work fault-tolerant as to a certain degree also erroneous dependencies must be taken into account. The deviation from the linguistic model can be described by a set of conflicts of the following three types:

- *Incongruency*: A dependency is presumed while features of the dependent do not exactly meet the specifications. E. g. in the hypothesis

  he sleep

  he can be interpreted as subject to *sleep* though regent and dependent do not correspond in number.
- *Vacant grammatical function*: A mandatory grammatical function is not fulfilled because there is no adequate chunk not occupied by another function. E. g.

  he gives to her

  causes a conflict because there is no chunk like *it* that can serve as direct object to *gives*.
- *Spare chunk*: At the end of the parsing process there is a chunk whose grammatical function could not be identified. E. g. the hypothesis

  he sleeps it

  contains the chunk *it* that cannot be incorporated into a semantic interpretation of the whole sentence.

Efficient parsing as it was postulated in section 3. can be achieved by relaxing the parser's precision: As we only need to find out if the input is erroneous, exact resolution of ambiguities is only required for error-related sequences. For all other ones it is sufficient to validate that there are admissible interpretations, no matter which one to choose.

5. Error Diagnostics

A subsequent diagnostics procedure analyzes the set of conflicts detected by the parser. It decides whether the speech recognizer output is accepted or rejected or whether an error model can be used to locate and analyze the confusion with the objective of correcting the sentence automatically or by means of a clarification dialog. Assuming that the linguistic model contains the original utterance, all those conflicts are caused by acoustic confusions. There are five types of errors a speech recognizer can produce (the examples refer to the utterance *I don't understand you*):

- *Simple confusion:* A word is replaced by another one:

  I don't understand who
- *Omission:* A word is omitted or replaced by a break:

  don't understand you
- *Insertion:* A word not contained in the utterance is inserted:

  I don't understand it you
- *Contraction:* A sequence of words is replaced by a single word:

  I don't understanding
- *Separation:* A single word is replaced by a sequence:

  I don't thunder stand you

In order to find out location and type of the confusion we need an error model that allows us to infer confusions from conflicts. However, in most cases there is no clear relation between cause and effect: A single confusion can cause more than one conflict. Often conflicts are bidirectional, that is either the dependent or the regent can be an erroneous word. In the end, several confusion types can cause same conflict. For an adequate handling of these ambiguous relations, we need a statistical error model.

6. Worked Example

In this section, we discuss all the steps involved in processing the 5 best hypotheses in Fig. 1 resulting in a localization and analysis of the detected errors.
The first step is to transform the single hypotheses into a joint word confusion graph as shown in Fig. 7. In this word confusion graph, sub-sequences of words are identified that are very unusual for German. The result of this step is a disambiguated word chain with markers for critical regions. In these regions, syntactic errors are likely to be found. First of all, a chunk parsing step tries to find all chunk readings for the given hypotheses or the whole word confusion graph. An extract of the chunk graph for the input in Fig. 8 can be seen in Fig. 9. This chart graph is the basis for the following dependency analysis. This step aims

```
(0,1,die) (1,2,sparte) (2,3,gelaufen) (3,4,wähle) (4,5,ard) (5,6,was) (6,7,gewählt)
                                      (3,4,das)   (4,5,vox) (5,6,uhr) (6,7,ausgewählt)
                                      (3,4,ab)    (4,5,elf)
```

Figure 7: Word Confusion Graph

```
(0,1,die)  (1,2,sparte)  (2,3,UNSURE)  (3,4,ab)  (4,5,elf)  (5,6,uhr)  (6,7,UNSURE)
```

Figure 8: Initial chart containing all error localization labels

at identifying how all chunks are related taking topological fields for German and the subcategorization frames of the involved words into account.
For going step by step through the second parsing phase we discuss another example from the home entertainment domain illustrating clearly the linguistic background and how error diagnostics can be made. The original utterance was

Ich möchte morgen abend einen Krimi im Fernsehen anschauen (Tomorrow night I would like to watch a detective story on TV)

The 5 best hypotheses of the parser—none of them contains the transcription—were

H1: *mich möchte morgen abend einen krimi im fernsehen anschauen (tomorrow night me would like to watch a detective story on TV)*

H2: *ich möchten morgen abend einen krimi im fernsehen anschauen (tomorrow night I would like to watch a detective story on TV)*
["would" being second-person plural]

H3: *ich möchte morgen abend einen krimi im ersehnt anschauen (tomorrow night I would like to watch a detective story on desired)*

H4: *ich möchte sorgen abend einen krimi im fernsehen anschauen (sorrow night I would like to watch a detective story on TV)*

H5: *ich möchte morgen haben einen krimi im fernsehen anschauen (tomorrow have I would like to watch a detective story on TV)*

In this example, parsing the word confusion graph would result in automatic correction of the sentence because it includes the transcription as a path. As no other path can be parsed without any conflicts, the search algorithm will definitely choose the transcription as best interpretation.
Parsing each word chain independently illustrates how error detection works if the error cannot be eliminated. First of all, the dependency parser performs the following steps:

1. Possible sentence readings and verb orders are taken into account. One of them is *statement* with *Verb-Zweit-Stellung (verb-second order)*. This reading calls for the finite verb to constitute the left bracket[1]. Finite verb candidates are *möchte*, *fernsehen* and *anschauen*. The former obtains the highest priority as the left bracket is expected to be early in the sentence.

2. The verb form *möchte* is an first-person singular modal auxiliary with subcategorization frame

   verbal_part(infinitive), subj(first-person sing. NP)

   The topology model allows the infinitive to be in the right bracket or the Vorfeld. As the left bracket has already been located in step 1, the Vorfeld is constituted by *ich*, which cannot be an infinitive. Thus *fernsehen* and *anschauen* are the candidates for the infinitive constituting the right bracket. This time the latter obtains higher priority.

3. It has the subcategorization frame

   object(NP)

   For the transcription, *ich* would be assigned to the subject slot and *einen Krimi* to the object slot while *morgen abend* would be interpreted as time supplement to the verb and *im fernsehen* as location adjunct to either the verb or the object.

   However, for each hypothesis one of these assignments can only be done at the cost of a conflict. Hence the A* search tests the other alternatives of step 1 and 2, but as none of them provides a conflict-free solution, the primary interpretation is rated best.

For the individual hypotheses, the parser detected the following conflicts:

H1: incongruency: subject *mich (me)* has accusative case instead of nominative

H2: incongruency: subject *ich (I)* differs from the verb in person and number

H3: Spare chunk: *ersehnt (desired)*
Vacant grammatical function: no noun for P *im (on)*

H4: Spare chunk: *sorgen (sorrow)*
Spare chunk: *abend (night)*

H5: Spare chunk: *haben (have)*

For H1 the error can be localized in the word *mich (me)*. An adequate repair strategy was either to guess that *mich (me)* was originally nominative or to ask the speaker for the subject: *Wer oder was? (who or what?)*
For H2 the system is unsure if the error is located in the subject or in the verb. Either both were originally first-person singular or both were second-person plural. If an error marker (compare Figure 8) was found at exactly one of both positions, it is possible to disambiguate this error diagnosis, with the same conclusions as for H1. If not, the risk is too high to choose the wrong alternative. Therefore it is better to ask for the subject.

[1]The left bracket is one of the five topological fields in German linguistics. For detailed information on German verb order and topological fields see Altmann and Hofmann (2004).

```
(0,2,DIE SPARTE,DP,I      (2,3,GELAUFEN) (3,6,AB ELF UHR,PP,I       (6,7,GEW"AHLT)
 [[DET:1 [[kas NOM]                     [[P:1 (1) [[pos prepos]
          [num SING]                                [kasrek DAT]]]
          [gen FEM]]]                     [NP:2 (3) [[kas DAT]
  [NP:2 (4) [[kas NOM]                                [num SING]
             [num SING]                               [gen FEM]]]
             [gen FEM]]]                  [PP:0 (1)]]])
  [DP:0 (4)]]])
```

Figure 9: Extract from the chart resulting from parsing an utterance of the data set.

For H3 it is likely that the spare chunk *ersehnt* originally fulfilled the vacant grammatical function (and thus described the location of the detective story). The adequate question to the speaker was either *wo? (where?)* or *in wem oder was? (on whom or what?)*

Diagnostics are more complicated for H4. As both chunks are adjacent, it is likely that both words form a single word or chunk in the original utterance. But as they serve as supplement (which is facultative and therefore not included in the subcategorization frame) their grammatical function is unknown. Hence a specific question cannot be formulated. Only if the system expected the user to tell something about time, it is reasonable to guess that the missing phrase is about time. In this case it could try to ask: *wann? (when?)*

However, the latter does not work for H5 because here a part of the time specification (*tomorrow*) is complete apart from the other one (*night*). Hence an adequate strategy was either to ignore the missing chunk (that is to suppose it to be irrelevant for the meaning of the utterance), to reject the whole sentence (*Sorry, could you repeat?*) or to formulate a position-oriented question (*Sorry, could you repeat the word you said after* morgen*?*).

7. Evaluation on a Small Test Corpus

For evaluating the parser on word chains, we used a subset of the EMBASSI corpus (see Görz and Ludwig (2005)) containing 5 best hypotheses (as shown in Figure 1) for each of 53 utterances. 87% of the transcriptions were recognized as error free while 80% of the wrong hypotheses were classified as erroneous. The latter is due to the fact that not all confusions by the speech recognizer result in syntactic errors. If we leave out hypotheses that are grammatically correct but contain semantic misunderstandings, 96,6% are recognized as syntactically erroneous. Taking only one best hypothesis per utterance (that is the favorite of the speech recognizer), both recognition rates rise to 94%.

Localizing the error on the basis of syntactic conflicts turns out to be more difficult. Even if we only take those hypotheses into account that cause a single conflict, for only 49% the error position can correctly and unambiguously be identified. In contrast, for 36% the conflict is not tangent to the misunderstood word.

Taking a look at the individual utterances we can clearly see that it is hardly possible to find the error by means of parsing when the stem of the finite verb is affected by the misunderstanding. This is due to the fact that the finite verb determines the most important semantic relations in a sentence and that its subcategorization frame differs for different verbs. Hence the syntactic interpretation of the utterance tends to be completely wrong if the finite verb cannot be identified correctly. The same holds for an infinite main verb if it rules other chunks in the utterance, e. g. if the finite verb is an auxiliary.

Besides, error diagnostics that are useful for repair strategies seem hardly practicable if an hypothesis contains more than one misunderstanding. Particularly in short utterances multiple errors often effectuate that a substantial part of the semantic structure is not accessible.

An analysis of time perfermance on a 2,2 GHz machine showed that 90% of all tested hypotheses were processed within considerably less than one second. The average time for these examples was 0,19 secs. All hypotheses out of the remaining 10%—some of them took up to 23 seconds—were erroneous. They can be skipped by a time limit that guarantees the applicability of the method in real-time dialog systems without lowering the quality.

8. Open Issues and Further Work

While the evaluation indicates that the approach is practicable in general, some questions remain open.

8.1. Suggestions for Better Diagnostics

More detailed error diagnostics—particularly if more than one conflict occurs—would require an elaborate statistical error model trained by large amounts of data. In order to provide more significant indications for error positions, the syntactical information given by conflicts should be combined with the statistical information given by error localization labels as shown in Figure 8. An important question to answer is if this could raise the rate of correctly localized errors to a level that makes a clarification worthwhile compared to a complete rejection.

Additionally, an evaluation is to be done on word confusion graphs as their use could lead to automatic correction provided that they contain every word of the utterance.

Up to now, our work was confined to independent utterances. However, in dialog systems contextual information is essential for interpretation of utterances. Considering context means to integrate our approach with a semantic analysis. Thereby also considerable improvements could be achieved both in efficiency and error detection, if dependencies between chunks are only considered if there is an adequate semantic relation between the words according to a semantic model. E. g. the verb *lesen (to read)*, that is allowed to rule an accusative object according to our linguistic model, would be restricted by the semantic model as the object must be something readable like a book. Of course, every semantic model would restrict the coverage to a small thematic domain.

What is more, a syntacto-semantic framework of expectation could be defined depending on the domain in order to disambiguate error diagnostics. E. g. a dialog system for information on train schedules would expect the user to tell something about date, time, departure and destination. If one of the appropriate functions cannot be assigned in an erroneous sentence, it is likely to be affected by the error.

8.2. Hard Issues for Parsing Spoken Language

Two linguistic phenomena disregarded in this paper are ellipsis and coordination. While non-elliptic coordinations could easily be integrated into our language model by redefining the second element as an dependent of the first one and so forth, ellipses are very hard to detect and resolve: how can we decide whether an ellipsis results from a misunderstanding or is intended by the speaker? For the purpose of resolution we need contextual information that is not only propagated from previous utterances but also interchanged between the coordination parts.

Up to now, our language model focusses on literary language, although the input of a natural language dialog system is a matter of spoken language. However, the grammar of spoken language turns out hard to describe. When talking spontaneously, a human speaker is only able to plan the syntactical structure up to a limited horizon given by his attention window. Due to the fact that the speaker cannot retroactively correct what he said, some ungrammaticality arises. Accepting this kind of deviation from the grammatical and topological restrictions of the model would be the key for a better handling of spoken language.

Some issues like self corrections (cf. Spilker et al. (2000)), enclitics, colloquial language or domain-specific terms can be handled by a preprocessing step. E. g. titles of telecasts that may extend on several words can be transformed into a special category `TITLE` that is handled as an independent or appositional noun phrase.

8.3. Portability to Other Languages

The general principle of this approach is also applicable to other languages as for every natural language a dependency grammar model can be built. The difference when switching to another language lies in the chunk rules and the subcategorization data. Beyond these adaptions, a new topology model must be built because the phenomenon of verb phrase fragmentation—which motivated the theory of topological fields—is a special peculiarity of German.

If finding speech recognition errors becomes easier or harder depends on the language. In general, natural languages can be classified according to their variability in word order: Grammatical functions of words can either be marked by flection or by word order. Languages with rich morphology like Latin prefer the former and thus can offer variable word order to a great extent. Languages with low capability of flecting words like English need to use the latter and therefore to highly restrict word order to avoid vast ambiguity. On that axis between flection and variable word order, German is located somewhere in the middle.

Grammatical functions marked by topology are less vulnerable to speech recognition errors than those marked by morphology because a word's position cannot be misunderstood, while inflected forms of a word usually only differ in single phonems or syllables that can very easily be confused. Thus we can assume that localizing and analyzing speech recognition errors would be easier in languages with restricted word order like English because the parser would have to take far fewer possible confusions into account. On the other hand, error diagnosis seems less practical for languages like Russian or Latin. However, if we only want to find out if an hypothesis is correct or not, we may get even better results for these languages as the effect of a confused word form cannot be compensated by the ambiguity of other word forms.

In order to find out if these assumptions prove true in practice further studies need to be performed.

8.4. Using Diagnostics for Clarification Strategies

In this section, we sketch briefly how the error diagnostics help the dialog system in initiating a clarification dialog. The following table shows what information the dialog system can extract from the (parsed) word confusion graph (see Figures 7, 8, and 9):

chunk	**interpretation**
(0,2)	DP chunk is the only hypothesis. It is correct.
(2,3)	PPART chunk is the only hypothesis, but in this position it is grammatically ill-formed.
(3,6)	PP chunk *ab elf Uhr* is grammatically correct, but there are several alternatives; so the risk for choosing it is very high even if the acoustic score of this reading is the highest available (see Figure 1).
(6,7)	The readings for the PPART chunk are semantically synonymous (in this domain). So, there is little risk in choosing any of them.

The topological information computed in phase two of the parsing process delivers the following diagnostics:

chunk	**interpretation**
(0,2)	DP chunk may be the subject in the utterance. However, it is semantically incomplete as *genre* lacks an apposition (which genre?). This fact is another indication that chunk (2,3) is misrecognized.
(2,3)	PPART chunk may be the (elliptical) predicate. However, none of its case frames can be filled by any other chunk. So, the risk is high for this interpretation.
(6,7)	PPART chunk may be the (elliptical) predicate. The risk for this option is low as (0,2) fills the subject case frame.

While deciding how to react on the utterance, the dialog system must be aware that (3,6) is very confusing. This indicates that a large part of the whole utterance may be misrecognized. The critical part even extends to (2,3) as discussed above. Therefore, a cautious strategy would be to reject the utterance as a whole and to answer:

Sorry, I didn't understand. Could you repeat, please?

A more risky strategy would assume the subject chunk (0,2) and the elliptical predicate (6,7) to be stable enough. So, the system could try to clarify the type of genre in chunk (2,3):

Which genre do you want to select?

This strategy also includes a clarification of (3,6) because it assumes the chunk to be a complement to the DP or the predicate. It further assumes the user to repeat this part of

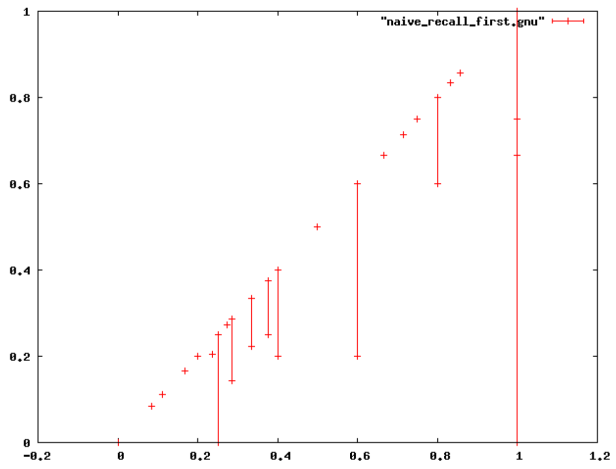

Figure 10: x-axis: percentage of errors in the transcription, y-axis: percentage of errors localized by error labels

the utterance as well. However, the risk for something different to happen is high. Therefore, this strategy potentially increases the risk for the whole clarification to fail.

9. Summary

The discussion of clarification strategies indicates further research directions: Do native speakers really interpret ill-formed sentences in a similar way to that presented here? Do they take similar decisions for how to continue a dialog in case they feel there is some misunderstanding?
While these are open issues for dialog research, the paper shows that progress can be achieved in the parsing phase: Our approach constructs a possible reading of user input and computes error diagnostics that are not related to artifacts of the parsing strategy or the grammar (formalism), but to a model of how native speakers analyze and understand utterances. The resulting analysis eases the tasks for modules in later steps of the natural understanding process. We showed that the analysis delivers valuable criteria for a dialog module. In case of misunderstandings it can generate more "natural" continuations than common slot-filling approaches do. The reason is again that the diagnostics are not related to the artifacts of the technical aspects of the dialog system, but to the human usage of spoken language. The evaluation of our approach to parsing proved to be reliable. We can show that almost all grammatical errors can be localized as indicated by Figure 10.

10. References

Hans Altmann and Ute Hofmann. 2004. *Topologie fürs Examen*. Linguistik fürs Examen. VS Verlag für Sozialwissenschaften, Wiesbaden, Germany, 1. edition.

Dan Bohus and Alexander I. Rudnicky. 2005. Sorry, I didn't catch that! - an investigation of non-understanding errors and recovery strategies. In Laila Dybkjaer and Wolfgang Minker, editors, *Proceedings of SIGDIAL 2005*, pages 128–143, Lisbon.

Daniele Falavigna, Roberto Gretter, and Giuseppe Riccardi. 2002. Acoustic and word lattice based algorithms for confidence scores. In *Proceedings of ICSLP 2002*.

Kilian Foth, Michael Daum, and Wolfgang Menzel. 2004. A broad-coverage parser for german based on defeasible constraints. In *Beiträge zur siebten Konferenz zur Verarbeitung natürlicher Sprache*, pages 45–52.

Malte Gabsdil and Oliver Lemon. 2004. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In *Proceedings of 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 344–351, Barcelona.

Kim Gerdes and Sylvain Kahane. 2001. Word order in German: A formal dependency grammar using a topological hierarchy. In *Meeting of the Association for Computational Linguistics*, pages 220–227.

Jonathan Ginzburg. 1998. Clarifying utterances. In J. Hulstijn and A. Nijholt, editors, *Proceedings of the 2nd Workshop on the Formal Semantics and Pragmatics of Dialogue*.

Günther Görz and Bernd Ludwig. 2005. Speech dialogue systems – a pragmatics-guided approach to rational interaction. *KI – Künstliche Intelligenz*, 10(3):5–10.

Dilek Hakkani-Tür and Giuseppe Riccardi. 2003. A general algorithm for word graph matrix decomposition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*.

Deirdre Hogan. 1998. Statistical methods for identifying umgrammaticality in texts. Master's thesis, Computer Science Department, Trinity College, University of Dublin.

Ulf Krum, Hartwig Holzapfel, and Alex Waibel. 2005. Clarification questions to improve dialogue flow and speech recognition in spoken dialogue systems. In *Proceedings of Interspeech 2005*.

Bernd Ludwig. 2006. Tracing actions helps in understanding interactions. In Jan Alexandersson and Alistair Knott, editors, *Proceedings of SIGDIAL 2006*, Sydney.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer Speech and Language*, 14(4):373–400.

David Schlangen. 2004. Causes and strategies for requesting clarification in dialogue. In Michael Strube and Candy Sidner, editors, *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*, pages 136–143, Cambridge, Massachusetts, USA, April 30 - May 1. Association for Computational Linguistics.

Ingo Schröder. 2002. *Natural Language Parsing with Graded Constraints*. Ph.D. thesis, Dept. of Computer Science, University of Hamburg.

Gabriel Skantze. 2004. Exploring human error recovery strategies: Implications for spoken dialogue systems. *Speech Communication*, 45(3):325–341.

Jörg Spilker, Martin Klarner, and Günther Görz. 2000. Processing self corrections in a speech to speech system. In *Proceedings of the 18th conference on Computational linguistics*, pages 1116–1120, Morristown, NJ, USA. Association for Computational Linguistics.

Chunking and Dependency Parsing

Giuseppe Attardi, Felice Dell'Orletta

Dipartimento di Informatica, Università di Pisa
largo B. Pontecorvo 3, I-56127 Pisa, Italy
attardi@di.unipi.it, dellorle@di.unipi.it

Abstract

Since chunking can be performed efficiently and accurately, its use is attractive as a preprocessing step in full parsing stages. For instance chunk data might be provided to a statistical dependency parser to improve its accuracy. We present a set of experiments meant to select a set of features that provides the greatest improvement to a Shift/Reduce statistical dependency parser. We report on the accuracy gains that such parser can obtain using features from gold chunks, from chunks produced using a statistical chunker and from approximate chunks obtained by detecting noun phrases through regular expression patterns. A parser exploiting features from approximate chunks is applied to a chunking task and its accuracy in chunking is compared to that of a specialized statistical chunker.

1. Introduction

Chunking or *shallow parsing* segments a sentence into a sequence of syntactic constituents or *chunks*, i.e. sequences of adjacent words grouped on the basis of linguistic properties (Abney, 1996).
This process can be carried out efficiently and thus chunking can be useful in several tasks, for instance Terminology Discovery, Named Entity Recognition (Carreras & Màrquez, 2005) and Text Mining (Banko et al., 2007), and also as an intermediate step providing input to further full parsing stages (Shiuan & Ann, 1996).
Chunking can be considered an example of *Partial Parsing*. Partial parsing was introduced to overcome the limitations of current full deep parsing techniques, trying to recover syntactic information efficiently and reliably from unrestricted text, by sacrificing completeness and depth of analysis (Abney, 1996).
The major critiques of full parsing techniques are lack of robustness for many NLP application and inability to identify good parse trees in noisy surroundings. Recent progress in statistical dependency parsing (see for instance the CoNLL 2007 Shared Task on multilingual dependency parsing (Nivre at al., 2007)) shows that dependency parsing can be made robust enough and also capable of achieving high accuracy in the analysis of multiple languages.
In this paper we will discuss whether providing chunk data to a statistical dependency parser can benefit its accuracy by presenting some experiments showing which kind of output from the chunker appears to be more effective in improving the accuracy of a dependency parser for English.
We first present a set of experiments meant to evaluate several possible features extracted from gold chunks and to determine which ones improve most the parser accuracy. We then verify whether these improvements are still preserved when using the output of a statistical chunker instead of the gold chunks.
Chunking exploits POS tags produced by previous processing steps and hence using a chunker leads to a more complex layered parser architecture. But since the parser itself may have access to the same information that the chunker uses to infer the chunks, one may wonder whether the parser might itself subsume the task of the chunker.
We will show that indeed the addition of simple chunker-like features allows the parser to achieve an accuracy close to that from using gold chunk data.
In particular we define a simple feature extracted from noun phrase boundaries induced through regular expression patterns applied to the input of a dependency parser.
The above information pre-segments the text and benefits the dependency parser accuracy thus avoiding both the propagation of errors introduced by the use of further statistical chunkers and the cost of an additional pre-processing step.
The only drawback is that these rules are language-dependent and hence must be adapted for each language.
Finally we analyze the degree of accuracy that a state-of-the-art dependency parser can achieve in a chunking task and compare it to that of a specialized statistical chunker.

2. Goal of the paper

The goal of the paper is to investigate the relationship between chunking and dependency parsing, in particular:

- to investigate whether a chunker can provide useful information to improve the accuracy of a dependency parser
- to investigate whether a dependency parser can produce as accurate chunks as those produced by a specialized statistical chunker.

The first issue has been analyzed for constituency parsers by Hollingshead, Fisher and Roark (2005).
We study it for dependency parsers, and in particular we use a deterministic Shift/Reduce dependency parser, because its speed and efficiency are comparable to those of shallow parsers. The second issue indeed makes sense only if these conditions are met.

3. Related Work

Hollingshead, Fisher and Roark (2005) compare high-accuracy context-free constituent parsers with high-accuracy finite-state chunkers on several shallow parsing tasks. Specifically, they compare the Charniak parser

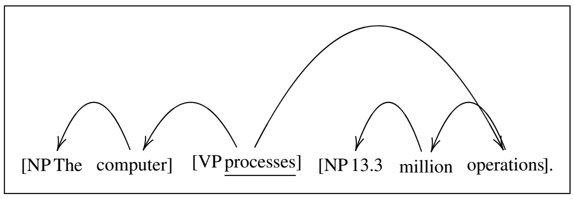
[NP The computer] [VP processes] [NP 13.3 million operations].

Figure 1: Chunked phrase and its dependency tree.

(Charniak & Johnson, 2005) with their own state-of-the-art chunker, concluding that there is no accuracy or robustness benefit to shallow parsing with finite-state models over using constituent parsers.

Vice-versa, but less surprisingly, they show large improvements in combining the output of high-accuracy context free parsers with the output of shallow parsers on shallow parsing tasks, but of course at significant higher performance costs.

Ciaramita and Attardi (2007) report on experiments by adding semantic features to a dependency parser. Two of these features are similar to chunk features: the EOS (End Of Segment) feature, similar to the distance to the end of a chunk, and the IOB tag, which however are only provided for Named Entities recognized in the text. Using these features, alone or in combinations, they report an improvement in error reduction in the LAS up to 5.8% with a parser trained on the WSJ Penn Treebank sections 2-21 (Marcus et al., 1993).

4. Chunk Information in Context Free Parsing

With respect to the dependency tree of a sentence, chunks have some properties which may provide useful hints to a parser.

For example all tokens in a chunk are linked through dependency chains to a single token which can be thus identified unambiguously as the *head of the chunk* (Federici et al., 1996). In the case of English, this is often the rightmost word of the chunk, in particular for noun phrases.

External links addressing a target in a chunk, point to the the head of the chunk. Figure 1 shows an example of a chunked phrase and its dependency tree.

The first set of experiments aims at investigating how best to exploit chunk information in a statistical dependency parser, selecting a set of features that provides the largest accuracy gain.

To this end we use "gold chunks" both in training and in testing. The chunks are those provided in the training set of the CoNLL 2005 Shared Task (Carreras & Màrquez, 2005). They were produced with a state-of-the-art statistical chunker (Carreras & Màrquez, 2003).

For our parsing experiments we used the WSJ sections 02-11 of the Penn Treebank II (Marcus et al., 1993), which were also used as the English training corpus in the CoNLL 2007 Shared Task on multilingual dependency parsing (Nivre at al., 2007), so that our scores can be compared with those of current state-of-the-art parsers.

Feature	Tokens
FORM	-1 0 1 $head$(-1)
POS	-2 -1 0 1 2 3 $leftChild$(-1) $leftChild$(0)
CPOS	-1 0 $prev$(-1)
DEPREL	-1 $leftChild$(-1) $rightChild$(-1) $leftChild$(0)

Table 1: Feature model for the baseline MaltParser.

4.1. Feature Model Selection

The first set of experiments was designed to evaluate the possible benefits of chunk data in parsing. We considered representing such information by means of the following four features types:

IOB: Inside, Outside, Beginning of chunk, in the standard IOB notation;

EOC: Distance to the end of the chunk;

TYPE: Type of the chunk;

NUMB: Number of the chunk within the sentence.

The IOB tag indicates whether a token is at the beginning, inside or outside a chunk (tokens outside of any chunk are mostly punctuation signs and conjunctions in ordinary coordinated phrases). TYPE denotes one of the 11 types of chunks defined in the CoNLL-2000 task (Sang & Buchholz, 2000).

In order to test which combination of these features was most effective, we used MaltParser (Nivre, Hall & Nilsson, 2006), a dependency parser which can be tailored selecting a specific set of features.

MaltParser is a classifier-based Shift/Reduce parser, which processes input tokens advancing on the input with Shift actions and accumulates processed tokens on a stack with Reduce actions.

The features of a number of tokens are considered at each step in the parsing algorithm as input to a classifier in order to decide the next action to perform. This set is called the *feature model*.

The features extracted for learning from the annotated corpus are: Form (the lexical form of the token), PosTag (the part of speech), CPosTag (the coarse part of speech) and DepRel (the dependency label).

As a baseline for English we used the same feature model chosen for English in (Hall et al., 2007), for their submission to the CoNLL 2007 Shared Task.

Table 1 describes concisely the feature model, listing which features are extracted from which tokens: positive numbers refer to input tokens while negative numbers refer to tokens on the stack, $leftChild(x)$ refers to the leftmost child of token x, $rightChild(x)$ to the rightmost child of token x, $head(x)$ to the head of x and $prev(x)$ to the token preceding x in the sentence.

We tested seven feature models, created by adding to the baseline model the four features individually as well as three combinations: IOB/NUMB, i.e. a pair of IOB tag

Model	LAS	UAS
baseline	85.06	86,23
NUMB	85.66	87,11
IOB	86.75	88.01
EOC	87.03	88.24
TYPE	87.10	88.36
IOB/NUMB	86.29	87.75
EOC/TYPE	**88.01**	**89.10**
IOB/NUMB + EOC/TYPE	86.50	87.89

Table 2: Parser accuracy with the addition of various chunk features.

and chunk number, EOC/TYPE, a pair of distance to end of chunk and chunk type, and finally both the last two pairs.
Differently from (Hollingshead, Fisher & Roark, 2005), we do not use a combination of parser and chunker results, but the chunker outputs are used directly as features.
Table 2 list the results obtained, measured in terms of labeled attachment score (LAS) and unlabeled attachment score (UAS).
All features provide improvements with respect to the baseline model. Knowledge about chunks can be helpful to a Shift/Reduce dependency parser, since it typically operates with a limited lookahead and hence has only a narrower vision of the phrase being analyzed than for instance a Maximum Spanning Tree parser (McDonald et al., 2005).
In English the last word in a chunk often coincides with the head of the chunk. Most words inside the chunk will depend on it and this will be the only word in the chunk to depend on words external to the chunk.
Hence the ability to identify the head of a chunk may help the parser for instance in: a) directing links from internal tokens to the head, b) avoiding creating links from outside the chunk.
The EOC feature is quite informative in this respect since it represents the distance to the head and indeed the EOC/TYPE combination achieves the highest score.
From these observations and from the observation that only noun phrases occurred frequently as chunks of length greater than two, we tested a variant of EOC computed only for noun phrases (EOC/NP), which achieved these scores: 87.65% LAS and 88.85% UAS. Hence, the EOC/NP alone gets a LAS and UAS scores that are not statistically significantly different from the best.
In a similar vein, Ciaramita and Attardi (2007) exploit the information derived from a Named Entity tagger to improve the accuracy of a dependency parser.
The semantic features extracted from the Named Entity tagger in their models exploit the named entity tag, the IOB tag and the distance from the end of the segment (EOS). IOB and EOS provide similar improvements, slightly better than named entity tags. The authors remark in fact that IOB tags break the text into segments and “with respect to the rest of the tree, segments tend to form units, with their own internal structure. Intuitively, this information seems relevant for parsing. These locally structured patterns could help particularly simple algorithms like ours, which have limited knowledge of the global structure being built.”

Feature	Tokens
FORM	-1 0 1
POS	-2 -1 0 1
FORM bigram	<-2,-1> <-1,0> <0,1>
POS bigram	<-2,0> <-1,0> <0,1>
POS trigram	<-2,-1,0>

Table 3: Chunker features.

Chunks	Precision	Recall	F-measure
all	95.10%	96.05%	95.57
NP	95.07%	94.62%	94.84

Table 4: Accuracy of Maximum Entropy chunker.

In our experiments the single feature producing the greatest improvements turned out to be EOC/NP, which is similar to EOS, but includes also determiners, adjective, etc.
So we decided to use the EOC for noun phrases in all further experiments.

5. Using a Statistical Chunker in Dependency Parsing

A more realistic experiment is to replace “gold chunks” with the output of a statistical chunker in order to check whether the benefits of chunking are preserved even when chunks are computed statistically.

5.1. Chunker

We developed a Maximum Entropy chunker that assigns an IOB tag to each word in a sentence, based on generic features like the lexical form of the tokens, the POS tags, bigrams and trigrams of POS tags and words.
Table 3 describes the feature used by the chunker.
For training the chunker the same corpus annotated with “gold chunks” was used as in the previous section.
Table 4 shows the accuracy of the chunker in terms of precision, recall and F-measure on the test set used in our parsing experiments. The accuracy is typical of state-of-the-art chunkers like the one by Carreras and Màrquez (2003) which achieved an F-measure of 93.74 at the CoNLL-2000 shared task.
The first row shows the accuracy on all types of chunks, while the second provides the values on chunking only noun phrases.
Since the chunker has linear complexity in the length of the sentence, its use will not increase the overall parser complexity.

5.2. Parsing Accuracy with Statistical Chunker

Table 5 shows the results of the experiments using MaltParser. The first row is a baseline with no chunk data, the second using EOC/TYPE features and gold chunks. The last two lines report experiments using the EOC feature for just NP-chunks, obtained either from gold annotations or from a statistical chunker.
With the use of statistically computed chunks, we observe only a slight improvement with respect to the baseline, but a

Model	Data	LAS	UAS
baseline		85.06	86.23
EOC/TYPE	gold	**88.01**	**89.10**
EOC/NP	gold	87.65	88.85
EOC	stat	85.18	86.50
EOC/NP	stat	**85.41**	**86.70**

Table 5: MaltParser accuracy with gold or statistical chunk features.

significant drop with respect to the accuracy achieved using the EOC feature extracted from "gold chunks".

6. Comparing a Statistical Chunker and a Dependency Parser

We now turn to comparing the accuracy of a specialized chunker to that of a dependency parser in the NP-chunking task.
We will try to assess whether a dependency parser can be as accurate as a special purpose finite-state shallow parser, as Hollingshead, Fisher and Roark (2005) showed in the case of a constituent parser.
We needed a tool for extracting chunks from a dependency tree similar to those produced by the the script *chunklink* (Buchholz, 2000) used in the CoNLL-2000 Shared Task (Sang & Buchholz, 2000).
This was not only difficult because constituent trees and dependency trees are quite different, but also because some of the choices in *chunklink* were hard to interpret. Some of them were questionable, for instance the words tagged NN, but considered out-chunk (O-) or VP chunks having as heads words tagged NN.
Such anomalies occur frequently in the data set provided by the CoNLL-2000 Shared Task and are somehow acknowledged in (Sang & Buchholz, 2000), when they cite Ramshaw and Marcus (1995):

> While this automatic derivation process introduced a small percentage of errors on its own, it was the only practical way both to provide the amount of training data required and to allow for fully-automatic testing.

Despite the incompatibilities between the chunks extracted from dependency trees and those extracted by *chunklink*, in the next section we will attempt a comparison between the accuracy of a specialized chunker and of a dependency parser in the chunking task.

6.1. Accuracy of a Dependency Parser at Chunking

There is an upper bound to the accuracy of chunks we can obtain from dependency parse trees due to inaccuracies of our chunks extractor: this limit is given in the first line of Table 6 where chunks are extracted from the correct parse trees.
The table also lists the percentages of precision, recall and F-measure obtained on NP-chunks extracted using three parsers augmented with chunk information: the first with no chunk data, the second with gold chunk data and the third with simulated chunk data, using the model EOC-102 presented in the next section.
The parser using EOC-102 chunks achieves an F-measure which is 1.7% less than the upper bound, 1.8% less than the statistical chunker and is even better than the one obtained using the statistical chunk informations (EOC/NP stat). We expect that such small difference might disappear by making our chunk extractor more similar to *chunklink*.
The results by the parser are quite acceptable, but one should consider that our chunk extractor from dependency trees is not fully consistent with the behavior of *chunklink*. In fact, results are much better if we compute the accuracy with respect to chunks obtained with our extractor from gold dependency parse trees, as shown in table 7.

Chunks	precision	recall	F
gold tree	94.84%	94.62%	94.73
none	92.97%	91.27%	92.11
EOC/NP stat	93.86%	91.85%	92.84
EOC-102	93.46%	92.64%	93.05
ME chunker	95.07%	94.62%	94.84

Table 6: Accuracy of dependency parser at chunking.

Chunks	precision	recall	F
none	95.58%	95.04%	95.31
EOC/NP stat	96.25%	95.28%	95.76
EOC-102	96.49%	95.88%	96.18

Table 7: Accuracy of dependency parser at chunking with respect to chunks extracted from gold dependency trees.

7. Approximate Chunking

In the previous sections we showed that pre-segmenting the text into chunks and determining the head of the chunk, greatly benefited the accuracy of a dependency parser. However, such benefits were not preserved when gold chunks were replaced by statistically extracted chunks.
In this section we present a method to obtain similar benefits as those provided by chunks in a dependency parsing task, as shown earlier, avoiding though the use of either gold chunks or statistically extracted chunks.

7.1. Simple Noun Phrases

We consider approximate noun phrases which can be recognized deterministically with a simple finite-state parser. Ambiguous cases, for instance due to the presence of conjunctions, are discarded.
A *simple noun phrase* (NP-simple) is a sequence of words ending with a noun (i.e. a token having one of the following POS: NN, NNS, NNP, NNPS) possibly followed by a possessive ending. Adjectives, adverbs, pronouns, nouns and determiners may precede the noun according to the following pattern:

```
RB+DT?(JJ|JJR|JJS|CD|VBD|PRP\$)*
((NN|NNS|NNP|NNPS)POS?)+
```

Model	LAS	UAS
baseline	85.06	86.23
NP-simple	85.78	87.03

Table 8: MaltParser accuracy with NP-simple features.

Detection of simple noun phrases can be easily incorporated within the feature extraction processing steps of the parser, avoiding the addition of a separate preprocessor and the potential introduction of extra errors.

When extracting deterministically NP-simple chunks, utilizing the regular expression seen above, the only cases of extra errors that can be introduced are the sequences *Noun Noun* not belonging to the same chunk but assembled in our NP-simple chunks. In these cases we miss pointing some *pseudo* EOC cases. Actually these cases are not very frequent, in fact in our test set they are present only 6 times on 4880 tokens, while in the training set 673 times on 436.916 tokens.

We will show how it is possible to obtain significant improvements in the dependency parser accuracy exploiting the pre-segmentation and the chunk head indication in the NP-like chunk.

7.2. Experimental Results

In the experiments with NP-simple chunks we used the same training and test as in the previous experiments.

The feature used to represent chunks is EOC, which had proved most effective with gold chunks. For the tokens contained in an NP-simple chunk, the feature represents the distance from the end of the chunk; for the other tokens the feature is just the POS of the token. Since this EOC is only an estimate and only for NP chunks, we call it *pseudo EOC* (EOC-pseudo).

Similarly to the experiments based on the NP chunks, the EOC-pseudo feature was extracted from two tokens on the stack and from three tokens on the parser input. The results of this experiment are shown for MaltParser in Table 8.

These results show a great improvement with respect to both the baseline and to those obtained using the statistical chunker. These demonstrate how the NP-simple chunk pre-segmentation and the determination of the final token crucially contribute to the improvement of the Shift/Reduce parser accuracy.

As stated before, such information provides a more comprehensive view of the remaining part to be analyzed rather than that of a dependency Parser based on the Shift/Reduce model. The latter is intrinsically unable to produce such results. Having used a non-ambiguous segmentation allow us to obtain really homogeneous train and test sets, differently from the experiments performed with the statistical parser. It was therefore avoided the necessity to introduce a greater quantity of information in the system through the introduction of a further training-set for the statistical chunker.

Having realized that using NP-simple chunks improves a dependency parser accuracy, we decided to verify whether we could reduce the number of features extracted from NP-simple chunks without much loss in accuracy.

Model	Tokens	LAS	UAS
EOC-10123	-1 0 +1 +2 +3	85.78	87.03
EOC-101	-1 0 +1	85.29	86.50
EOC-102	-1 0 +2	**85.96**	**87.13**
EOC-103	-1 0 +3	85.86	86.99
EOC034	0 +3 +4	85.20	86.25
EOC-12	-1 +2	85.59	86.80
EOC-13	-1 +3	85.61	86.78
EOC-10	-1 0	85.82	86.89
EOC03	0 +3	85.49	86.62
EOC0	0	85.78	86.82

Table 9: Models using EOC-pseudo features and corresponding accuracy scores.

Table 9 lists the feature models that we tested and the corresponding accuracy scores. For each model we report the tokens for which the EOC-pseudo features are extracted.

The simplest model EOC0, which exploits the EOC-pseudo information only from the next input token, achieves a score which is not statistically different from the best result. More surprisingly, the score is higher than most of those obtained using features extracted from “gold chunks” as reported in an earlier section.

The best model, EOC-102, exploits information from the token on top of the stack, the next input token and the second input token.

This is the model we used in the chunk extraction experiments in the previous section. However, this choice is somewhat arbitrary, since there is a statistically significant difference ($p - value < 0.05$) only between the best model and the last two ones. Statistical significance was computed using the *compare* script (Bikel, 2006), with 10.000 iterations.

Apparently, the EOC feature on token +1 is less relevant once the value for token 0 is known: indeed whenever the value of EOC for token 0 is positive, the value for token +1 is determined.

8. Conclusions

While chunking can be a useful tool in itself, we have shown that it is of marginal utility as a preprocessing step to full dependency parsing.

Vice versa, dependency parsing can provide quite good accuracy at chunking and can provide also richer syntactic information on sentences for many language processing applications.

With the current availability of efficient and accurate dependency parsing technologies, dependency parsing should be considered as a valid and more sophisticated alternative to chunking in many applications requiring language processing.

Sagae, Miyao and Tsujiii (2007) for example have shown that constituent parsing can benefit from exploiting dependency constraints.

Chunking is an example of a task which has been sometimes delegated to a preprocessing stage in order to simplify the task of the parser by reducing the complexity of the data to analyze (Shiuan & Ann, 1996) or the number of features to deal with.

Parsers still rely on preprocessors for simple syntactic tasks like POS tagging, or they might rely on semantic analyzers for Named Entity Recognition. While this sounds appropriate from a software engineering perspective, it also breaks the sources of information in an artificial way.

Since most of these preprocessors are now based on statistical machine learning methods, an alternative approach would be to create a combined system which, instead of combining the outputs of the individual processors, collects the features that each one would extracts and learns directly from those.

This might have been impractical up to recently because of the explosion of the feature space, but it can be feasible by using methods that are capable of performing feature induction, like those using latent variables (Titov & Henderson, 2007).

Acknowledgments

This work has been supported in part by a grant from Fondazione Cassa di Risparmio di Pisa and by a donation from Yahoo! Research.

9. References

S. Abney. 1996. Tagging and Partial Parsing. In K. Church, S. Young, and G. Bloothooft (eds.), *Corpus-Based Methods in Language and Speech*.

G. Attardi. 2006. Experiments with a Multilanguage Non-Projective Dependency Parser. In *Proceedings of CoNNL-X 2006*. http://desr.sourceforge.net.

M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead and O. Etzioni. 2007. Open Information Extraction from the Web. In *Proceedings of the 20th IJCAI*.

D. Bikel. 2006. Randomized Parsing Evaluation Comparator. http://www.cis.upenn.edu/˜dbikel/software.html #comparator.

S. Buchholz. 2000. http://ilk.uvt.nl/˜sabine/chunklink/ README.html.

S. Buchholz and E. Marsi. 2006. Introduction to CoNNL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of CoNNL-X 2006*.

X. Carreras and L. Màrquez. 2003. Phrase Recognition by Filtering and Ranking with Perceptrons. In *Proceedings of RANLP-2003*.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task. In *Proceedings of CoNLL-2005*.

E. Charniak and M. Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of ACL 2005*.

M. Ciaramita and G. Attardi. 2007. Dependency Parsing with Second-Order Feature Maps and Annotated Semantic Information. In *Proceedings of IWPT 2007*.

S. Federici, S. Montemagni and V. Pirrelli. 1996. Shallow Parsing and Text Chunking: a View on Underspecification in Syntax. In *Proceedings of the Workshop On Robust Parsing. (ESSLLI-1996)*.

J. Hall, et al. 2007. Single Malt or Blended? A Study in Multilingual Parser Optimization. In Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007

K. Hollingshead, S. Fisher and B. Roark. 2005. Comparing and Combining Finite-State and Context-Free Parsers. In *Proceedings of HLT/EMNLP 2005*.

R. McDonald, F. Pereira, K. Ribarov and J. Hajič. 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP 2005*.

M. Marcus, B. Santorini and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2): 313–330.

J. Nivre, J. Hall and J. Nilsson. 2006. MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proceedings of the fifth Int. Conf. on Language Resources and Evaluation (LREC2006)*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel and D. Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of EMNLP/CoNLL-2007*.

L.A. Ramshaw and M.P. Marcus. 1995. Text Chunking Using Transformation-Based Learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*.

E. Tjong Kim Sang, S. Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL-2000*.

E. Tjong Kim Sang, F. De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language Independent Named Entity Recognition. In *Proceedings of CoNLL-2003*. 142–147.

K. Sagae, Y. Miyao and J. Tsujii. 2007. HPSG Parsing with Shallow Dependency Constraints. In *Proceedings of the 45th Annual Meeting of the ACL*.

P.Li Shiuan and C. Ting Hian Ann. 1996. A Divide-and-Conquer Strategy for Parsing. In *Proceedings of IWPT 1996*. 57–66.

I. Titov and J. Henderson. 2007. Constituent Parsing with Incremental Sigmoid Belief Networks. In *Proceedings of ACL 2007*.