Chunking and Dependency Parsing

Giuseppe Attardi, Felice Dell'Orletta

Affiliation1, Affiliation2, Affiliation3
Address1, Address2, Address3
author1@xxx.yy, author2@zzz.edu, author3@hhh.com

Abstract

Since chunking can be performed efficiently and accurately, it is attractive to use it as a preprocessing step in full parsing stages. We analyze whether providing chunk data to a statistical dependency parser can benefit its accuracy. We present a set of experiments meant to select first a set of features that provide the greates improvement to a Shift/Reduce dependency parser, then to determine an appropriate feature model. We report on accuracy gain obtained using features from chunks produced using a statistical chunker as well as from an approximate representation of noun phrases induced directly by the parser. Finally we analyze the degree of accuracy that such a parser can achieve in chunking compared to 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 see as the basic task in *Partial Parsing*. Partial Parsing was introduced as a response to the difficulties of the full traditional parsing and it is described as a techniques to recover syntactic information efficiently and reliably from unrestricted text, by sacrificing completeness and depth of analysis (Abney, 1996). Among the critiques of full parsing (and in favor of partial parsing) the most important are that full parsing are not sufficiently robust for many NLP application and that full parsing doesn't identify good parse tree in noisy surroundings. Recent progresses in full statistical parsing (see for instance the CoNLL 2007 Shared Task on multilingual dependency parsing (Nivre at al., 2007)) show that full parser is not only robust but also capable of giving good results in analysing many different 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 inducing noun phrase boundaries from simple rules applicable 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. 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 (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).

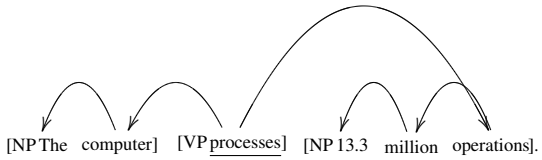

Figure 1: Chunked phrase and its dependency tree.

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.

3. 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.

3.1. 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 the dependency parser MaltParser (Nivre, Hall & Nilsson, 2006), which we could taylor by adding extra types of features.

MaltParser is a classifer-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 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 directy 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

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.

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."
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.

4. 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.

4.1. 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.

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.

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.

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

5. 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

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.

Chunks	precision	recall	F
gold tree	94.84%	94.62%	94.73
none	92.09%	90.27%	91.17
EOC/NP	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.

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.

5.1. 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 1.8% less than the upper bound, 1.9% less than the statistical chunker and is even better than the one obtained using the gold chunks. 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 they could be even better if chunks were obtained in a more consistent way than through a conversion from *chunklink*, for instance directly from dependency trees.

6. Simplified 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 dependency parsing task, as shown in the first part of this paper, though avoiding the use of both gold chunks and statistically extracted chunks.

6.1. 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?)+
```

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.

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.

6.2. 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 7.

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.

Model	LAS	UAS
baseline	85.06	86.23
NP-simple	85.78	87.03

Table 7: MaltParser accuracy with NP-simple features.

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

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 un test, 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.

Table 8 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.

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.

7. 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 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

8. 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*.

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*.