Improving an NP-Chunker by Exploting the Inner NP Dependency Structure

Mihaela Colhon
University of Craiova, Department of
Computer Science, 13 Alexandru Ioan
Cuza St.
mcolhon@inf.ucv.ro

Dan Cristea
“Alexandru Ioan Cuza” University of Iaşi, Faculty
of Computer Science, 16 Berthelot St. Iaşi
Institute of Computer Science, the Iaşi branch of
the Romanian Academy, 2 Codrescu St.
dcristea@info.uaic.ro

ABSTRACT

In this article we extend our previous work [3] dedicated to developing an automatic method for generating and applying syntactic patterns intended to recognize noun phrases. The intended scope of our studies is to automatically detect dependency relations within noun phrases exploiting some syntactic information of the phrases’ words. The patterns are extracted from a corpus that has been automatically annotated for token and sentence limits, part of speech and noun phrases and manually annotated for dependency relations between words. The patterns have been generalized in order to cover more instances that were present in the corpus and to reduce the size of the grammar. In this paper we refine the syntactic patterns by considering not only the two terms of the dependency relation but also the terms surrounding these two that will be treated as contexts: left, middle, right depending on their position from the dependency terms.

Author Keywords

Natural Language Processing; Dependency Parsing; HCI

ACM Classification Keywords

H.5.2 User Interfaces (e.g., HCI): Natural language

General Terms

Human Factors; Design; Measurement.

INTRODUCTION

A way to incorporate into user interfaces knowledge expressed in natural language or modalities for communicating in a human to human style can be achieved by implementing Natural Language Processing tasks. In the context of Human Computer Interaction (shortly, HCI) the Natural Language Processing (shortly, NLP) issues range from various speech recognition systems, to natural language interfaces for various applications, or a multitude of machine translation systems.

Natural language processing applications, such as information retrieval, automatic translation, sentiment analysis or applications that automatically have to answer questions, require both semantic analysis and morpho-syntactic analysis of texts at different levels. Particularly interesting from the HCI perspective are the research conducted in the domain of linguistics or the works put into constructing treebanks, both monolingual and parallel [2].

As an exemplification of the necessity of reliable natural language processing tools, let us consider the following question-answer example:

the information: "[..] Prices of sunflower oil rose by 27% due to low production [..]"

and the question: "For which food the price has increased by 27%?"

The answer "sunflower oil" is within the nominal group "Prices of sunflower oil ", so the correct identification of the nominal group as well as the disambiguation of its structure is absolutely necessary in order to provide the correct answer. If the parser could not recover the structure of the nominal group then the correct answer candidate could not be found even if the phrase segment containing one or more nominal groups was been correctly identified. This problem also affects anaphase resolution systems, automatic syntactic analysis, but also automatic translation using syntactic trees [4, 16].

Automatic syntactic parsing of a text is used either singularly or in some linguistic processing chains in order to ease their purpose and to enhance precision, such as: coreference chains, entity recognition, nominal group processing (study in which the present work is included), or automated translation of a text. Syntactic parsing is the next level following morphological labelling, the latter being at the base of any processing chain.

Two of the most used formalisms to describe syntactic structures are in terms of hierarchies of constituents and dependency relations. Contiguous sequences of words are grouped under non-terminal symbols (as constituents), part of context free grammars, while dependency relations [11] are asymmetrical functional relations between pairs of words, considered head and modifier.

While these two traditions have sometimes been presented as competing with each other, a straightforward correspondence between a projective dependency analysis (in which there are no crossing links) and a constituent structure analysis seems to exist [1].

Reliable dependency parsing is a notorious difficult problem in Natural Language Processing [5]. We describe in this paper a pattern-based approach in dependency parsing that addresses only Nominal Phrases (NPs).

To build a dependency treebank, the human experts must decide for each word which is the one it depends on. However, rather often there is no consensus among annotators on what the correct dependency structure for a particular sentence should be, because the decision regarding dependencies involve a deep interpretation process.

In contrast, in building phrase-structures, human annotators are confronted to much less ambiguity. This is because only a sequence of constituents should be indicated in the right hand side of a grammar rule, not also their relative roles/functions with respect to the parent constituent, as indicated by the left symbol. It is therefore normal that the effort of establishing correct dependency structures be paid back, and the difference stays in a much closer resemblance of a dependency structure to a semantic interpretation then in the case of a constituent structure. This aspect is even more important in the case of languages that have a free word order (as are for instance Czech, Romanian, etc.), where dependency treebanks are preferred to constituent structure representations (see, for instance the Prague Dependency Treebank [8]).

Using treebank data for training and evaluation of parsing systems is identified under the name of treebank parsing, a methodology that has been used to construct robust and efficient parsers for several languages over the last ten years [10]. For this kind of parsing, the treebank data is used to train the parser but also to evaluate the quality of the resulted parser with respect to accuracy as well as efficiency. Calacean and Nivre [3] report results on a MaltParser-based dependency [12] for Romanian, trained on a manually annotated Romanian Treebank[1] built in the RORIC-LING project [9]. Their precision for recognizing labelled relations are between 60.8% and 95.9%, depending on the length of the link, while the recall is in the range 71.3% - 96.3%. Their corpus includes only short sentences (with an average of 8.94 tokens per sentence) and a gold standard part-of-speech annotation.

In this paper we present an on-going research started in 2014 on a treebank dependency parsing mechanism that is restricted to NP chunks. The presented study refines the syntactic patterns defined in [5] by considering not only the two terms of the dependency relation but also the terms surrounding these two that will be treated as contexts: left, middle, right depending on their position from the dependency terms. We work on Romanian, but the method is general enough to be applicable to other languages as well.

The paper is organized as follows: in Section 2 we present the Treebank corpus we use for our study. Section 3 describes the manner in which the collection of patterns extracted from an automatic annotation to NP chunks over the Treebank are organized. Section 4 describes the method and the results while Section 5 formulates a number of conclusions.

THE TREEBANK

At <anonym-university>, a dependency Treebank for the Romanian language was built. We have used this resource in the training and evaluation stages of our proposed mechanism.

The corpus contains Romanian texts selected from a wide range of genres/registers of language[2]. Three levels of annotation have been added to the raw text and encoded in XML, by adopting a simplified form of the XCES standard [7]:

- Level-1: segmentation and lexical information. Sentences have their boundaries manually marked and each token has attached its part of speech, lemma, and morpho-syntactic information, by running an automatic processing chain that includes: tokenisation, POS-tagging and lemmatisation [14];
- Level-2: noun phrases. By exploiting Level-1 information, an NP-chunker [15] adds information regarding noun phrase boundaries and their head words;
- Level-3: syntactic dependency data. During a manual annotation phase [13] each token of all sentences has been complemented with its head-word and the dependency relation towards the head.

The Treebank thus acquired contains 2,630 sentences, in which 7674 NP structures were identified by the NP chunker.

THE DATABASE OF NOUN PHRASES PATTERNS

The primary data extracted from the corpus is used to associate dependency structures with each sequence of MSD[3] tags corresponding to NPs extracted from the corpus,

[1] http://www.phobos.ro/roric/texts/xml/

[2] The corpus mainly includes Romanian translations of the first chapter of George Orwell's novel "1984", Romanian parts from the JRC-Acquis corpus, Romanian Wikipedia texts, grammar texts used in high schools, etc.

[3] Morpho Syntactic Description – notation used in the Multext projects [6].

which we will call in the following morphological structures.

The corpus includes only contiguous NPs. As we have already said, the corpus used in this study puts in evidence three levels of annotations: POS tags, NP chunks and dependency relations. For each NP chunk, we extracted the morphological structure and the configuration of dependency relations manually marked among the words of the NP chunk.

To syntactic constituents of the sentence correspond dependency structures organized in (sub)trees. Each node of such a (sub)tree is a word, connected with a dependency relation to its head word. The roots of these (sub)trees are the only elements related to words outside the constituents themselves. The only exception is the very root of the sentence which is not related to another word. In the same way, the head word of an NP is the only word belonging to the NP related outside the NP.

For example, if we take the following bracket representation for a Romanian noun phrase „*lamele de ras tocite*" (En. "*the blunt razor blades*"):

```
[NP [Ncfpry lamele] [Spsa de] [Ncfsry
ras] [Afpfp-n tocite]]
```

its morphological structure is:

```
        Ncfpry Spsa Ncfsry Afpfp-n[4]
```

and its internal dependencies are:

```
a.subst.(lamele-1,de-2)
prep.(de-2,ras-3)
a.adj.(lamele-1,tocite-4)
```

In the notations above the name of relation is placed in front of a pair of words, the first one being the head and the second - the modifier word. The number attached to a word denotes its position inside the chunk.

A database table is built out of the 3 layers of notations in the corpus: each record in this table includes a triplet

```
      <seq-msd> <seq-rels> <seq-heads>
```

in which <seq-msd> represents a morphological structure (a pattern of MSD tags), <seq-rels> represents the sequence of dependency relations (transferred from the words to the MSD representations on the respective positions - the first position is counted as 1) and <seq-heads> is the sequence of head positions in the pattern[5]. The 0-rel always marks in <seq-rels> the relation of the head word of the NP going out of the NP, and with 0-head is noted in <seq-heads> the exterior head of the head word of the NP. The head of an NP chunk is usually attached, in the syntactic tree of the sentence, to another word which, of course, is not part of that NP sequence.

For example, to the NP "*lamele de ras tocite*", the following entry corresponds in the database:

```
<Ncfpry Spsa Ncfsry Afpfp-n>

<0-rel a.subst. prep. a.adj.>

<0-head 1 2 1>
```

Based on the dependency data coded in the corpus the entries of the database are grouped in three categories:

- with no external heads: no 0-rel in the <seq-rels> field;
- with one external head: exactly one 0-rel in the <seq-rels> field;
- with more than one external head: minim two 0-rel in the <seq-rels> fields.

Using these representations one can easily detect the incorrect NP sequences marked in the corpus but also the ambiguous <seq-msd> sequences with respect to their dependency structures.

Indeed, exploiting this grouping, the NP chunks incorrectly marked in the corpus during the automatic NP-chunking results immediately as they correspond to <seq-msd> entries which include more than one 0-rel in the <seqrels> sequence.

Opposite to this case, <seq-msd> sequences with one single external head represent correct NP chunks that are connected with the other words of the sentences they belong to by means of their heads.

Correct MSD sequences of NP chunks are also the <seq-msd> entries in the database with no 0-rel. These cases correspond to verb elliptical sentences where the NP is the very root of the sentence - the main verb is missing.

The ambiguous dependency constructions popup immediately as being the sets of entries in the database that have the same <seq-msd> sequence.

APPLYING PATTERNS

Our method for generating and applying corpus based patterns in dependency parsing works as follows (see Figure 1): starting from flat NP sequences and using a set of syntactic patterns extracted from the training corpus, we identify dependency links between the words of the NPs, based on their MSD finger-print. By this, the flat NP sequences become dependency sub-trees. In order to do that, the dependency relations, considered independent one of the others, are decoupled from the particular morphological structures they occur in. The set of contexts of each relation is then tried to be generalized. The aim of

[4] See the Appendix for the meaning of the MSD tags in this paper.

[5] Of course, the property of unique occurrence of a record in the database is observed here as well.

the generalization is to reduce the number of dependency relations patterns extracted from the corpus, but also to infer deducible sequences not instantiated in the corpus. During the pattern-generalization process, the following steps are repeated for all records of the database in which a relation *R* occurs between identical tags:

for each `<seqrels>` sequences in the database, three types of contexts are marked:

- **left context**: is represented by the sequence of MSD tags in `<seq-msd>` appearing to the left of the position of the first element involved in the dependency *R*;
- **middle context**: is represented by the sequence of MSD tags appearing in between the two tags involved in the targeted dependency relation
- **right context**: is represented by the sequence of MSD tags appearing to the right of the second element involed in the targeted dependency relation *R*.

For each dependency relation, the dependents are represented by two MSD tags:

- `1:` the MSD tag for the head
- `2:` the modifier MSD tag

The contexts (if they are present in the pattern structure) may be optional or mandatory: the optional ones are marked with `?` and the mandatory ones with `{1}`.

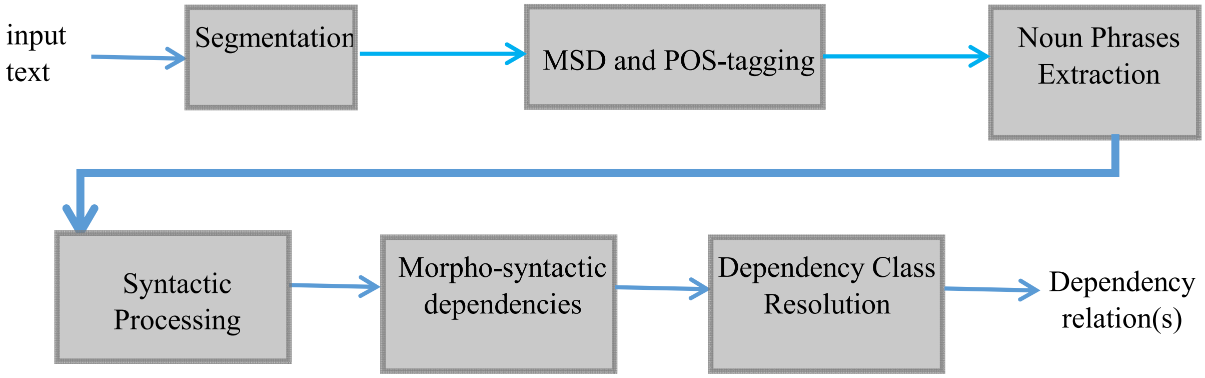

Figure 1. Dependency Parsing Mechanism within Noun Phrases

Let us now take three examples of NPs:

NP#1: "*Fetele acestea frumoase*" (En: "*These beautiful girls*")

NP#2: "*Mâinile lui curate*" (En: "*His clean hands*")

NP#3: "*Lumânarile aprinse*" (En: "*candles lit*")

In all these NPs the same relation (adjectival attribute, noted with "*a.adj.*") is found between words displaying identical MSD tags (that is `Ncfpry` as head and `Afpfp-n` as modifier). The following patterns for this dependency relation are found:

From NP#1:

```
<1:Ncfpry Dd3fpr- 2:Afpfp-n>
```

From NP#2:

```
<1:Ncfpry Pp3mso- 2:Afpfp-n>
```

From NP#3:

```
<1:Ncfpry 2:Afpfp-n>
```

After the generalization process, all these representations will be merged into a single one with an optional middle context:

```
<1:Ncfpry (Dd3fpr-|Pp3mso)? 2:Afpfpn>
```

The middle context in the generalized pattern is optional because it can either contain one of the tags separated by the "or" operator ("|") or can be an empty one like in the last example.

After generalization, the patterns extracted from the corpus were grouped into 24 sets, each corresponding to one relation and covering the span of an entire NP.

The evaluation scores obtained are given in the next section.

EVALUATION

The total Treebank sentences were split into a training set and a testing set. The parser was trained on approximately 90% of the Treebank and evaluated on the remaining 10% using a 10-fold cross-validations policy, which guaranteed no intersection between training and evaluation sentences.

For the testing set, the most frequent relations identified by the parser were:

- the adjectival attribute (*a.adj.*),
- nominal attribute (*a.subst.*),
- determiner (*det.*) and
- preposition (*prep.*).

No restrictions of projectivity of the generated dependency structures have been included at this moment. The obtained scores are given in Table 1. As one can observe, the best accuracy was obtained for determiner relation while the lower was for the adjectival attribute relation. Nevertheless, the evaluation scores give hope for continuing our pattern-based approach in Romanian dependency parsing.

Accuracy **Dependency relations**	**Precission**	**Recall**	**F-measure**
ALL	0.63	0.74	0.68
a.adj.	0.54	0.86	0.66
a.subst.	0.67	0.72	0.69
det.	0.90	0.82	0.86
prep.	0.74	0.66	0.70
Unlabelled relations	0.72	0.78	0.75

Table 1: Evaluation scores

CONCLUSION

It is well-known that there is a difficult trade in designing proper generalization patterns, because making them too lax could trigger false instances. On the other hand, making them too straight will imply low recall scores. There is perhaps more to be done in this direction.

We are aware that the model would gain in precision if lexical information would be included, by enriching the MSD tags of the generated patterns with lemmas. In our experiments till now we excluded lexical information, because lexical information presupposes a much larger training corpus, which was not at our fingers during this phase of research.

REFERENCES

1. Michael Brody. 1994. Phrase structure and dependence. Working papers in the theory of grammar. 1(1), Theoretical Linguistics Programme, Budapest Univ.
2. Marian Cristian Mihăescu, Mihaela Gabriela Ţacu, Dumitru Dan Burdescu. 2014. Use Case of Cognitive and HCI Analysis for an E-Learning Tool, Informatica 38: 273-279
3. Mihaela Călăcean, Joakim Nivre. 2009. A Data-Driven Dependency Parser for Romanian, Proceedings of TLT-7.
4. Alexandra Cristina Cristea. 2014. Syntactic Study on Nominal Groups in Romanian (in Romanian), Dissertation Thesis, "Alexandru Ioan Cuza" University of Iaşi
5. Mihaela Colhon, Dan Cristea. 2014. Automatic Extraction of Syntactic Patterns for Dependency Parsing in Noun Phrase Chunks, Bucharest Working Papers in Linguistics, vol. XVI, nr. 1, ISSN 1454-9328
6. Tomaž Erjavec. 2010. Multext-east version 4: Multilingual morphosyntactic specifications, lexicons and corpora. In LREC.
7. Nancy Ide, P. Bonhomme, L. Romary. 2000. Xces: An xml-based encoding standard for linguistic corpora. In Proceedings of the Second International Language Resources and Evaluation Conference. Paris: European Language Resources Association.
8. Jan Hajič, Alena Böhmová, Eva Hajičová, Barbora Vidová-Hladká. 2000. The prague dependency treebank: A three-level annotation scenario, A. Abeillé, editor, Treebanks: Building and Using Parsed Corpora, Amsterdam:Kluwer: 103-127.
9. Florentina Hristea, Marius Popescu (eds.). 2003. Building Awareness in Language Technology, University of Bucharest Publishing House.
10. Svetoslav Marinov, Joakim Nivre. 2005. A Data-Driven Dependency Parser for Bulgarian. In Proceedings of TLT 2005: 89-100.
11. Igor Mel'čuk. 1987. Dependency Theory: Syntax and Practice, Albany, NY:SUNY Press
12. Joakim Nivre, Johan Hall, Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. Proceedings of LREC-2006: 2216-2219
13. Cenel A. Perez. 2012. Casuistry of Romanian functional dependency grammar, M. A. Moruz, D. Cristea, D. Tufis, A. Iftene, H. N. Teodorescu (eds.) Proceedings of the 8th International Conference "Linguistic Resources And Tools For Processing Of The Romanian Language": 19-28
14. Radu Simionescu. 2012. Graphical grammar studio as a constraint grammar solution for part of speech tagging, M. A. Moruz, D. Cristea, D. Tufis, A. Iftene, H. N. Teodorescu (eds.) Proceedings of the 8th International Conference "Linguistic Resources And Tools For Processing Of The Romanian Language": 109-118.

15. Radu Simionescu. 2012. Romanian deep noun phrase chunking using graphical grammar studio, M. A. Moruz, D. Cristea, D. Tufis, A. Iftene, H. N. Teodorescu (eds.) Proceedings of the 8th International Conference ‘Linguistic Resources And Tools For Processing Of The Romanian Language”: 135-143.

16. David Vadas, James R. Curran. 2011. Parsing Noun Phrases in the Penn Treebank, School of Information Technologies, University of Sydney, Australia.

APPENDIX

The following table gives the Morpho-syntactic descriptions (shortly, MSD) used in this paper.

MSD tag	**The meaning of the notation (according to MULTEXT-East lexical specifications)**
Afpfp-n	Adjective qualifier positive feminine plural -definiteness
Dd3fpr-	Determiner demonstrative third feminine plural direct
Ncfpry	Noun common feminine plural direct +definiteness
Ncfsry	Noun common feminine singular direct +definiteness
Pp3mso-	Pronoun personal third masculine singular oblique
Spsa	Adposition preposition simple accusative