10.12871/clicit201422

Dependency Parsing Techniques for Information Extraction

Giuseppe Attardi
Dipartimento di Informatica
Università di Pisa
Largo B. Pontecorvo, 3
I-56127 Pisa, Italy
attardi@di.unipi.it

Maria Simi
Dipartimento di Informatica
Università di Pisa
Largo B. Pontecorvo, 3
I-56127 Pisa, Italy
simi@di.unipi.it

Abstract

English. Dependency parsing is an important component in information extraction, in particular when using suitable formalisms and accurate and efficient parsing techniques. We review recent advances in dependency parsing and describe our own contribution in the context of the Evalita 2014 DPIE task.

Italiano. *Il parsing a dipendenze è un componente importante nell'estrazione di informazione da testi, in particolare quando usato con una rappresentazione adeguata e tecniche di parsing accurate ed efficienti. Accenniamo agli sviluppi recenti nel parsing a dipendenze e descriviamo il nostro contributo nel contesto del task DPIE di Evalita 2014.*

1 Introduction

Information extraction is one of the primary goals of text analytics. Text analytics is often performed by means of advanced statistical tools, relying on patterns or matching with gazetteers for identifying relevant elements from texts. Dependency parsing in an attractive technique for use in information extraction because it can be performed efficiently, parsers can be trained on treebanks in different languages, without having to produce grammars for each of them and they provide a representation that is convenient to use in any further layers of analysis.

The effectiveness of the dependency representation was shown for example in the CoNLL 2008 Shared task on Joint Dependency Parsing and Role Labelling (Surdeanu et al. 2008): over 80% of the roles did indeed correspond to either direct or double indirect dependency links. Stanford Dependencies (SD) introduce a notation for dependencies that is closer to the representation of the roles so that they are easier to extract. Universal Dependencies in particular, generalized from SD, are helpful for dealing uniformly with multiple languages (De Marneffe et al., 2014).

Deep parsing (Ballesteros et al., 2014) can extract "deep-syntactic" dependency structures from dependency trees that capture the argumentative, attributive and coordinative relations between full words of a sentence.

Practical uses of text analysis based on dependency structure are reported in many applications and domains, including medical, financial or intelligence. Google for example applies dependency parsing to most texts it processes (Goldberg, 2013): parse trees are used in extracting relations to build the Knowledge Vault (Dong et al., 2014) and to guide translation (Katz-Brown et al., 2011).

There is still potential for improving dependency parsers in several directions:

- Integration with other layers of analysis, e.g. POS tagging and role labelling.
- Improving the accuracy.
- Exploiting distributed word representations (word embeddings).

Recent work on improving accuracy has explored two issues: the strategy adopted in the analysis and the use of features in the parsing decision process.

Transitions parsers are affected by the problem of having to decide sometimes too early which attachment to make, before having seen the remaining part of the sentence.

Goldberg and Elhadad (2010) proposed a so-called "easy first" approach, directing the parser to complete the simplest structures first and dealing with their combination later when more information from the constituents is available.

Sartorio, Satta and Nivre (2013) propose new parsing rules that allow delaying attachments: e.g. given the two top stack words *w* and *z*, *RA-k* allows adding a dependency link from the *k*-th rightmost descendant of *w* to *z*. These parsing rules only handle cases of non-projectivity.

A similar effect can be obtained by using in a creative way the rules for handling non-projectivity introduced by Attardi (2006). The effect of *RA-k* can be obtained by delaying attachments performing *Shift*'s and recovering later using a *Left-k* rule, in cases where the delay turns out to have been unnecessary. This approach allows retaining the parser ability to handle non-projectivity.

During training, a parser is typically shown only one sequence of decoding actions computed by a training oracle guide that knows the correct parse tree. However there can be more than one sequence for building the same parse tree. Hence during training, the oracle could present all of them to the parser. This would teach the parser actions that may be useful in situations where it must recover from earlier errors.

These experimental solutions have still to find their way into a production dependency parser.

Besides the mentioned approach by Attardi for handling non-projectivity, another approach has been proposed later, which consists in introducing a single *Swap* action to exchange the two top elements of the stack. Often though the action though must be applied multiple times during parsing to move a whole constituent, one word at a time, to a new place where it can be eventually reduced. For example, the sentence:

```
Martin Marietta Corp. said it
won a $ 38.2 million contract
from the U.S. Postal Service to
manufacture and install auto-
mated mail - sorting machines .
```

requires the following sequence of actions[1]:

```
S R S L S R S S R S S S L S L S
R R S S S S S R R R L S swap S
S swap S S swap S S swap L L S
S swap S S swap S S swap L S S
swap R S S swap R R L L L L S L
L S L L
```

Basically, after the parser has reduced the phrases "`a $ 38.2 million contract`" and "`from the U.S. Postal Service`", it has to move the prepositional phrase "`to manufacture and install automated mail - sorting machines`" in front of the latter, by means of a sequence of alternating *Shift*/*Swap*, before it can be attached to the noun "`contract`". Nivre, Kuhlmann and Hall (2009) propose to handle this problem with an oracle that delays swaps as long as possible.

With the rules by Attardi (2006) instead, a single non-projective action (*Left*-2) is required to parse the above sentence:

```
S R S L S R S S R S S S L S L S
R R S S S S S R R R L L-2 S S S
S S L L S S S L S R S R R L L L
L L S L L
```

Notice that action *Left*-2 is equivalent to the pair *Swap RA*.

Non-projectivity has been considered a rare phenomenon, occurring in at most 7% of words in free order languages like Czech: however, counting the number of sentences, it occurs e.g. in over 60% of sentences in German.

Other approaches to deal with wrong too early parsing decision are to use a stacking combination of a left-to-right and right-to-left parser or to use a larger size beam. In the latter approach many alternative parsing are carried along and only later the wrong ones are pruned. Bohnet and Kuhn (2012) propose this approach in combination with a way to score the partial parse trees exploiting graph-based features.

Among the approaches to provide semantic word knowledge to improve parsing accuracy we mention the use of word clusters by Koo, Carreras and (2008) and leveraging information from the Knowledge Graph (Gesmundo and Hall, 2014). Word embeddings are used in the parser by Chen and Manning (2014).

2 Tools

Our experiments were based on DeSR, the first transition based parser capable of dealing directly with non-projective parsing, by means of specific non-projective transition rules (Attardi, 2006).

The DeSR parser is highly configurable: one can choose which classifier (e.g. SVM or Multi-Layer Perceptron) and which feature templates to use, and the format of the input, just by editing a configuration file. For example, to implement stacking, one needs to specify that the format of the input used by the second parser contains additional columns with the hints from the first par-

[1] We use a shorthand notation where `R` is a right parse action (aka *LA*), `L` is a left parse action (aka *RA*) and `S` is a *Shift*.

ser and how to extract features from them with a suitable feature model.

Rich features of the type proposed by Zhang and Nivre (2011) can be specified with the following notation, where 0 identifies the next token and -1 the last token, expressions indicate a path on the tree and eventually which token attribute to extract as a feature:

```
POSTAG(0)LEMMA(leftChild(-1))
```

It is also possible to represent conditional features, which depend on the presence of other words. For example, the following rule creates a pair consisting of the lemma of the next token and the lemma of the last token which was a verb, but only if the current token is a preposition:

```
if(POSTAG(0) = "E", LEMMA(0))
LEMMA(last(POSTAG, "V"))
```

Features may consist of portions of attributes that are selected by matching a regular expression. For example, a feature can be extracted from the morphology of a word:

```
match(FEATS(-1), "gen=.")
```

Binned distance features can be expressed as follows:

```
dist(leftChild(-1), 0))
```

Data Set

The EVALITA 2014 evaluation campaign on Dependency Parsing for Information Extraction is based on version 2.0 of the Italian Stanford Dependency Treebank (ISDT) (Bosco et al., 2013). It was provided to the participants split into a training set consisting of 7,398 sentences (158,447 tokens) and a development set of 580 sentences (12,123 tokens).

ISDT adopts an Italian variant of the Stanford Dependencies annotation scheme.

Experiments

The flexibility of DeSR allowed us to perform a number of experiments.

As a baseline we used DeSR MLP, which obtained scores of 87.36 % LAS and 89.64 % UAS on the development set. We explored using a larger number of features. However, adding for example 16 word-pair features and 23 triple-word features, the score dropped to 85.46 % LAS and 87.99 % UAS.

An explanation of why rich features are not effective with the DeSR parser is that it employs a Multi-Layer Perceptron that already incorporates non linearity in the second layer by means of a *softsign* activation function. Other parsers instead, which use linear classifier like perceptron or MIRA, benefit from the use of features from pairs or triples of words, since this provides a form of non-linearity.

To confirm this hypothesis, we built a version of DeSR that uses a passive aggressive perceptron and exploits graph completion, i.e. it also computes a graph score that is added to the cumulative transition score, and training uses an objective function on the whole sentence, as described in (Bohnet and Kuhn, 2012). This version of DeSR, called DeSR GCP, can still be configured providing suitable feature templates and benefits from reach features. In our experiments on the development set, it reached a LAS of 89.35%, compared to 86.48% of DeSR MLP.

2.1 Word Embeddings and Word Clusters

We explored adding some kind of semantic knowledge to the parser in a few ways: exploiting word embeddings or providing extra dictionary knowledge.

Word embeddings are potential conveyors of semantic knowledge about words. We produced word embeddings for Italian (IWE, 2014) by training a deep learning architecture (NLPNE, 2014) on the text of the Italian Wikipedia.

We developed a version of DeSR MLP using embeddings: a dense feature representation is obtained by concatenating the embedding for words and other features like POS, lemma and deprel, also mapped to a vector space. However, experiments on the development set did not show improvements over the baseline.

Alternatively to the direct use of embeddings, we used clusters of terms calculated using either the DBSCAN algorithm (Ester et al., 1996) applied to the word embeddings or directly through the word2vec library (WORD2VEC. 2014).

We added cluster features to our feature model, extracted from various tokens, but in no configuration we obtained an improvement over our baseline.

2.2 Adding transitivity feature to verbs

Sometimes the parser makes mistakes by exchanging subjects and passive subjects. This might have been due to its lack of knowledge about transitive verbs. We run an experiment by adding an extra attribute TRANS to verb tokens, denoting whether the verb is transitive, intransi-

tive or both. We added to the feature model the following rules:

```
if(POSTAG(0) = "V", TRANS(0))
LEMMA(-1)
if(POSTAG(-1) = "V", TRANS(-1))
LEMMA(0)
```

but the LAS on the development set dropped from 87.36 to 85.54.

2.3 Restructuring Parse Trees

Simi, Bosco and Montemagni (2014) argued for using a simpler annotation scheme than the ISDT schema. The proposed schema, called MIDT++, is attractive not just because of a smaller number of dependency types but also because it provides "easier to learn" dependency structures, which can be readily converted to ISDT.

The results from that paper suggested the idea of a transformational approach for the present DPIE task. We experimented performing several reversible transformations on the corpus, before training and after parsing.

The transformation process consists of the following steps:

1. apply conversion rules to transform the training corpus;
2. train a parser on the transformed training set;
3. parse the test sentences with the parser;
4. transform back the result.

Each conversion rule *Conv* must be paired with a $Conv^{-1}$ rule, for use in step 4, such that:

$$Conv^{-1}\ (Conv\ \mathrm{T}) = \mathrm{T}$$

for any dependency tree T. We tested the following transformations:

- *Conv-conj*: transform conjunctions from grouped (all conjuncts connected to the first one) to a chain of conjuncts (each conjunct connected to the previous one);
- *Conv-iobj*: for indirect objects, make the preposition the head, as it is the case for other prepositional complements;
- *Conv-prep-clauses*: for prepositional clauses, labeled either *vmod* or *xcomp*, make the preposition the head;
- *Conv-dep-clauses*: for subordinate clauses, *advcl* and *ccomp*, make the complementizer the head;
- *Conv-NNP*: turn proper nouns into a chain with the first token as head.

Arranging conjunctions in a chain is possibly helpful, since it reduces long-distance dependencies. The *Conv-conj* conversion however may entail a loss of information when a conjunct is in turn a conjunction, as for instance in the sentence:

Children applaud, women watch and smile …

In order to preserve the separation between the conjuncts, this transformation, and other similarly, introduce extra tags that allow converting back to the original form after parsing.

The transformations were quite effective on the development set, improving the LAS from 89.56% to 90.37%, but not as much on the official test set.

2.4 Parser configurations

In our final experiments we used the following parsers: transition-based DeSR MLP parser (Attardi et al., 2009), transition-based with graph completion DeSR GCP, graph-based Mate parser (Bohnet, 2010), graph-based TurboParser (Martin et al., 2012).

DESR MLP is a transition-based parser that uses a Multi-Layer Perceptron. We trained it on 320 hidden variables, with 40 iterations and a learning rate of 0.001, employing the following feature model:

Single word features
$s_2.l\ s_1.l\ b_0.l\ b_1.l\ b_2.l\ b_3.l\ b_0^{-1}.l\ lc(s_1).l\ lc(b_0).l\ rc(s_1).l\ rc(b_0).l$
$s_2.p\ s_1.p\ b_0.p\ b_1.p\ b_2.p\ b_3.p\ s_1^{+1}.p\ lc(s_1).p\ lc(b_0)$.p
$rc(s_1).p\ rc(b_0).p$
$s_1.c\ b_0.c\ b_1.c$
$s_1.m\ b_0.m\ b_1.m$
$lc(s_1).d\ lc(b_0).d\ rc(s_1).d$
$match(s_1.m$, "gen=.")
$match(b_0.m$, "gen=.")
Word pair features
$s_1.c\ b_0.c$
$b_0.c\ b_1.c$
$s_1.c\ b_1.c$
$s_1.c\ _2.c$
$s_1.c\ _3.c$
$rc(s_1).c\ b_0.c$
Conditional features
$if(b_0.p$ = "E", $b_0.l)\ last$(POSTAG, "V")).l

Table 1. Feature templates: s_i represents tokens on the stack, b_i tokens on the input buffer. $lc(s_i)$ and $rc(s_i)$ denote the leftmost and rightmost child of s_i, *l* denotes the lemma, *p* and *c* the POS and coarse POS tag, *m* the morphology, *d* the dependency label. An exponent indicates a relative position in the input sentence.

For the DeSR GCP parser we used the features described in (Bohnet and Nivre, 2012).

The Mate parser is a graph-based parser that uses passive aggressive perceptron and exploits reach features. The only configurable parameter is the number of iterations (set to 25).

TurboParser is a graph-based parser that uses third-order feature models and a specialized accelerated dual decomposition algorithm for making non-projective parsing computationally feasible. TurboParser was used in configuration "full", enabling all third-order features.

2.5 Parser combination

Further accuracy improvements are often achieved by ensemble combination of multiple parsers. We used the parser combination algorithm by Attardi and Dell'Orletta (2009), which is a fast linear algorithm and preserves a consistent tree structure in the resulting tree. This is relevant for the present task, since the evaluation is based on relations extracted from the tree. An algorithm that only chooses each link independently, based on independent voting, risks of destroying the overall tree structure.

3 Results

We submitted three runs, all with the same combination of the four parsers above. They differ only in the type of conversion applied to the corpus:

1. Run1: *Conv-iobj*, *Conv-prep-clauses*
2. Run2: no conversion
3. Run3: *Conv-iobj*, *Conv-prep-clauses, Conv-dep-clauses*

The first run achieved the best accuracy scores among all submissions, according to the LAS (Labeled Accuracy Score) and UAS (Unlabeled Accuracy Scores), as reported in Table 2. Punctuations are excluded from the evaluation metrics.

Run	LAS	UAS
Unipi_Run1	**87.89**	**90.16**
Unipi_Run2	87.83	90.06
Unipi_Run3	87.84	90.15

Table 2. Evaluation of accuracy on dependencies.

Unipi_Run1 also obtained the best scores in the evaluation of accuracy on extracted relations, as reported in Table 3.

The results show an apparent correlation between the two types of evaluations, which we observed consistently also during our experiments on the development set. Our tree-based combination algorithm preserves this property also on the combined output.

Run	Precision	Recall	F1
Unipi_Run1	**81.89**	**90.45**	**85.95**
Unipi_Run2	81.57	89.51	85.36
Unipi_Run3	81.54	90.37	85.73

Table 3. Evaluation on accuracy of relations.

The scores obtained on the test set are significantly lower than those we had obtained on the development set, where the same parser combination achieved 90.37% LAS and 92.54% UAS. Further analysis is required to explain such difference.

4 Conclusions

The Evalita 2014 task on Dependency Parsing for Information Extraction provided an opportunity to exploit a larger training resource for Italian, annotated according to an international standard, and to test the accuracy of systems in identifying core relations, relevant from the perspective of information extraction.

There have been significant advances recently in dependency parsing techniques, but we believe there are still margins for advances in the core techniques along two directions: new transition rules and strategies for applying them, and exploiting semantic information acquired from distributed word representations.

We have started exploring these ideas but for the moment, we achieved top accuracy in this task using just consolidated techniques.

These remain nevertheless promising research directions that are worth pursuing in order to achieve the performance and accuracy needed for large-scale information extraction applications.

Acknowledgments

Luca Atzori and Daniele Sartiano helped performing the experiments using embeddings and clusters.

References

Giuseppe Attardi. 2006. Experiments with a Multilanguage non-projective dependency parser. In: *Proc. of the Tenth Conference on Computational Natural Language Learning* (CoNLL-X '06), 166-170. ACL, Stroudsburg, PA, USA.

Giuseppe Attardi, Felice Dell'Orletta. 2009. Reverse Revision and Linear Tree Combination for Dependency Parsing. In: *Proc. of Human Language*

Technologies: The 2009 Annual Conference of the NAACL, Companion Volume: Short Papers, 261–264. ACL, Stroudsburg, PA, USA.

Giuseppe Attardi, Felice Dell'Orletta, Maria Simi, Joseph Turian. 2009. Accurate Dependency Parsing with a Stacked Multilayer Perceptron. In: *Proc. of Workshop Evalita 2009*, ISBN 978-88-903581-1-1.

Miguel Ballesteros, Bernd Bohnet, Simon Mille and Leo Wanner. 2014. Deep-Syntactic Parsing. In: *ProceedingsProc. of COLING 2014*.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proc. of Coling 2010*, pp. 89–97, Beijing, China. Coling 2010 Organizing Committee.

Bernd Bohnet and Jonas Kuhn. 2012. The Best of Both Worlds - A Graph-based Completion Model for Transition-based Parsers. In: *Proc. of EACL*. 2012, 77-87.

Bernd Bohnet and Joakim Nivre. 2012. Feature Description for the Transition-Based Parser for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing. Retrieved from http://stp.lingfil.uu.se/~nivre/exp/features_emnlp12.pdf

Cristina Bosco, Simonetta Montemagni, Maria Simi. 2013. Converting Italian Treebanks: Towards an Italian Stanford Dependency Treebank. In: *ACL Linguistic Annotation Workshop & Interoperability with Discourse*, Sofia, Bulgaria.

Danqi Chen and Christopher D. Manning. 2014. Fast and Accurate Dependency Parser using Neural Networks. In: *Proc. of EMNLP 2014*.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, Christopher D. Manning. 2014. Universal Stanford Dependencies: a Cross-Linguistic Typology. In: *Proc. LREC 2014*, Reykjavik, Iceland, ELRA.

Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy. 2014. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion.

Martin Ester et al 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the 2nd Int. Conference on Knowledge Discovery and Data Mining*. AAAI Press. pp. 226–231.

Andrea Gesmundo, Keith B. Hall. 2014. Projecting the Knowledge Graph to Syntactic Parsing. *Proc. of the 15th Conference of the EACL*.

Yoav Goldberg and Michael Elhadad. 2010. An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing. *Proc. of NAACL-2010*.

Yoav Goldberg. 2013. Personal communication, http://googleresearch.blogspot.it/2013/05/syntactic-ngrams-over-time.html

IWE. 2014. Italian Word Embeddings. Retrieved from http://tanl.di.unipi.it/embeddings/.

Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno. 2011. Training a Parser for Machine Translation Reordering. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. of ACL 2008*, Columbus, Ohio, USA.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order nonprojective turbo parsers. In: *Proc. of the 51st Annual Meeting of the ACL* (Volume 2: Short Papers), 617–622, Sofia, Bulgaria. ACL.

Joakim Nivre, Marco Kuhlmann and Johan Hall. 2009. An Improved Oracle for Dependency Parsing with Online Reordering. *Proc. of the 11th International Conference on Parsing Technologies (IWPT)*, 73–76, Paris, October.

Ryan McDonald et al. 2013. Universal dependency annotation for multilingual parsing. In: *Proceedings of ACL 2013*.

NLPNET. 2014. Retrieved from https://github.com/attardi /nlpnet/

Maria Simi, Cristina Bosco, Simonetta Montemagni. 2008. Less is More? Towards a Reduced Inventory of Categories for Training a Parser for the Italian Stanford Dependencies. In: *Proc. LREC 2014*, 26–31, May, Reykjavik, Iceland, ELRA.

Mihai Surdeanu, Richard Johansson, Adam Meyers. Lluís Màrquez and Joakim Nivre, 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies, *Proc. of the 12th Conference on Computational Natural Language Learning*, 159–177, Manchester, August 20.

Francesco Sartorio, Giorgio Satta and Joakim Nivre. 2013. A Transition-Based Dependency Parser Using a Dynamic Parsing Strategy. In: *Proc. of ACL 2013.*

WORD2VEC. 2014. Retrieved from http://code.google.com/p/word2vec/

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In: *Proc. of the 49th ACL: Human Language Technologies*: Short papers, Volume 2, 188-193. ACL.