# PARSING A DIPENDENZE BIDIREZIONALE ADDESTRATO SUL TURIN UNIVERSITY TREEBANK

# *BIDIRECTIONAL DEPENDENCY PARSING TRAINED ON THE TURIN UNIVERSITY TREEBANK*

LUCAS CHAMPOLLION · PRASHANTH MANNEM · LIVIO ROBALDO

## SOMMARIO/*ABSTRACT*

In questo articolo descriviamo l'applicazione di un parser a dipendenze bidirezionale [3] addestrato sul Turin University Treebank [1].
*In this paper, we describe the application of a bidirectional dependency parser [3] trained on the Turin University Treebank [1].*

**Keywords:** Dependency parsing, bidirectional parsing, LTAG-spinal

## 1 Introduction

Shen [3] proposed a bidirectional dependency parsing algorithm which does a greedy search over the sentence and picks the relation between two words with the best score each time and builds the partial tree instead of doing a left-to-right or right-to-left parsing. The search can start at any position and can expand the partial results in any direction. The order of search is learned automatically. The parser uses two operations *attach* and *adjoin* to establish a relation between two entities. The parser requires the data to be in LTAG-spinal format [3], a format derived from LTAG (Lexicalized Tree-adjoining Grammar [2]). Although neither LTAG nor LTAG-spinal are dependency grammars formalisms, we can use the parser for dependency parsing anyway as it operates directly on LTAG-spinal derivation trees, which are very close to dependency trees. The CoNLL format data released by the organizers is converted into LTAG-spinal format by using the *attach* operation to represent projective relations in the corpus and the *adjoin* operation to represent non-projective relations in the corpus. In the next section, we formally present the bidirectional parsing algorithm. In section 3 we present the results of our system in EVALITA [1].

## 2 Parsing algorithm

We first define the data structures and then formalize the bidirectional parsing algorithm. Each word is associated a set of hypothesis POS tags in the input. A POS tag with lexical item is called a **node** in dependency parsing. For initialization, each word comprises a **fragment**, a continuous part of a sentence. A **fragment** hypothesis represents a possible analysis for a fragment. We can combine the hypotheses for two nearby fragments with various operations like *attachment* and *adjunction*. We represent an operation $R_{type,main}$ with a 4-tuple

$$R_{type,main}(f_l, f_r, n_1, n_2)$$

where $type \,\epsilon$ {*adjunction, attachment*} is the type of operation. $main$ = $left$ or $right$, representing whether the left or the right fragment is the parent. $f_l$ and $f_r$ stand for the left and right fragment hypotheses involved in the operation. $n_1$ and $n_2$ stand for the nodes involved in the operation.

An operation $R$ on fragment hypotheses $R.f_l$ and $R.f_r$ generates a new hypotheses $f(R)$ for the new fragment which contains the fragments of both $R.f_l$ and $R.f_r$. A priority queue $Q$ is used to store all the candidate operations that could be applied to the current partial results. Operations in $Q$ are ordered with the score of an operation $s(R)$. We have

$$s(R) = W.\phi(R)$$

$$score(f(R)) = s(R) + score(R.f_l) + score(R.f_r)$$

where $s(R)$ is the score of the operation $R$, which is calculated as the dot product of a weight vector $W$ and $\phi(R)$, the feature vector of $R$. $s(R)$ is used to order the operations in $Q$.

The feature vector $\phi(R)$ is defined on $R.f_l$ and $R.f_r$, as well as the context hypotheses. If $\phi(R)$ only contains information in $R.f_1$ and $R.f_r$ we call this **level-0 feature dependency**. If features contain information of outside hypotheses of nearby fragments, we call this **level-1 feature**

**dependency**. We introduce a **chain**, which is used to represent a set of fragments, such that hypotheses of each fragment always have feature dependency relations with some other fragments within the same chain. Furthermore, each fragment can only belong to one chain. A set of related fragment hypotheses is called a **chain hypothesis**. For a given chain, each fragment contributes a fragment to build a chain hypothesis. We use beam search and set a predefined beam width, which means that we keep the top k chain hypotheses for each chain. The score of a chain hypothesis is the sum of the scores of the fragment hypotheses in this chain hypothesis. A **cut** $T$ of a given sentence, $T = \{c_1, c_2, \cdots, c_m\}$, is a set of chains satisfying

- exclusiveness: $\cup c_i \cap \cup c_j = \emptyset, \forall i, j$, and
- completeness: $\cup(\cap T) = V$ .

Furthermore, we use $H^T = \{H^c | c \epsilon T\}$ to represent of sets of chain hypotheses for all the chains in cut $T$. With the above formal notations, we now list the training algorithm in Algorithm 1. A sentence is a linear graph with an edge between the adjacent words.

**Algorithm 1** Training Algorithm

```
W ← 0;
for round = 1..T, i = 1..n do
  LOAD graph $G_i(V, E)$, hidden structure $Y_i$;
  INITIATE cut $T$, hypotheses $H^T$, queue $Q$;
  while $Q$ is not empty do
    operation $y \leftarrow \arg_{op \epsilon Q}$ max score (op, W);
    if compatible($Y_i$,$y$) then
      UPDATE $T$, $H^T$, $Q$ with $y$ ;
    else
      $y^*$ ←searchCompatible($Q$,$y$);
      W ← W + $\phi(y^*) - \phi(y)$;
      UPDATE $Q$ with W;
    end if
  end while
end for
```

## 3 Results

Our system achieved 85.46% UAS (Unlabeled Attachment Score) in the EVALITA, 2007 dependency parsing task. The test corpus had sentences from two sources (Civil law and Newspaper). The UAS on test sentences (2607 words) from civil law is 88.30%, whereas for sentences (2357 words) from newspaper text, it is 82.61%. The performance on the newspaper test corpus is lower than that of the civil law corpus since the latter one is restricted to a single domain and the former one is not. The analysis of the performance of the parser across POS tags reveals that the parser performs poorly on punctuation, prepositions and coordination. In principle, the parser handles non-projective arcs effectively using the adjunction operation. However, since TUT has very few non-projective arcs, the true power of the parser could not be realized.

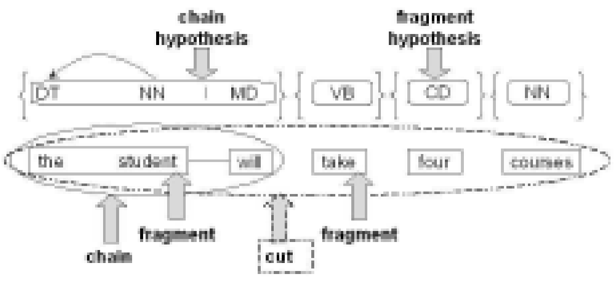

Figure 1: A state of the parser showing the **fragment**, **fragment hypothesis**, **chain**, **chain hypothesis** and **cut**

## 4 Future work

Since the work described here we have extended the parser to perform labeled dependency parsing. In the future, we plan to make use of this label information to improve the unlabeled attachment score.

## REFERENCES

[1] C. Bosco. A grammatical relation system for treebank annotation. Ph.D. Thesis. Turin University, 2004.

[2] A. K. Joshi and Y. Schabes. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, eds., Handbook of Formal Languages, vol. 3, pp. 69-124. Springer, 1997.

[3] L. Shen. Statistical LTAG Parsing. Ph.D. thesis. University of Pennsylvania, 2006.

CONTACT

LUCAS CHAMPOLLION, PRASHANTH MANNEM
*University of Pennsylvania*
*Email: champoll@ling.upenn.edu,*
*pmannem@seas.upenn.edu*

LIVIO ROBALDO
*Department of Computer Science, University of Turin*
*Email: robaldo@di.unito.it*

**LUCAS CHAMPOLLION** is a Ph.D. student of Linguistics at the University of Pennsylvania. His interests include formal and computational properties of LTAG, dependency parsing and formal semantics.