Mixed probabilistic and deterministic dependency parsing

Christophe Cerisara, Alejandra Lorenzo

LORIA/CNRS UMR7503, Vandoeuvre-les-Nancy, France

cerisara@loria.fr, alejandra.lorenzo@loria.fr

Abstract

This work describes a new multi-stage dependency parsing framework that relies on stochastic probabilistic models, such as the Maximum-Entropy Markov Model. It proposes an original compromise between locally optimal parsers with global features, and globally optimal models with local features. The main advantage of this framework is its ability to choose the desired compromise over the full range between both extreme models, by modifying the topology of the underlying automaton. Thanks to its probabilistic definition, it further gives access to several powerful classical probabilistic algorithms, and in particular to marginalization and Bayesian inference of, for instance, missing or corrupted observations. The rank-1 model has been evaluated on a French broadcast news parsing task, and has obtained comparable performance to state-of-the-art transition-based parsers.

1. Introduction

Dependency parsing aims at finding the best syntactic dependency tree over an observed sentence. However, because of the greater than exponential number of possible structures over sentences, finding the globally optimal tree is not possible in a reasonable amount of time. This challenge paved the way for two main broad classes of algorithms: those focusing on a globally optimal exploration with local features, such as in the maximum spanning tree algorithm; and those optimizing a local objective function but with features that depend on the complete history of actions, such as in transition-based parsers. The work presented here describes a flexible framework that allows to build intermediary systems where the developer can choose and tune the desired compromise between both alternative solutions. In addition, the resulting model is probabilistic, as opposed to state-of-the-art systems such as [1, 2], which gives access to a wide range of interesting algorithms and techniques, such as the capability to marginalize out or infer missing or partial observations.

2. Related work

Most state-of-the-art dependency parsers belong to one of the two following classes of parsers:

1. Maximum-spanning tree parsers, such as the MST-parser [3]. These parsers explore the whole space of every possible dependency tree for each sentence, which enables them to infer the best tree in this space. This class of parsers also includes recently proposed discriminative CRF-based parsers [4, 5], which, thanks to a richer set of features, improve on the performance of generative parsers. However, the complexity of the algorithms in this class is typically in the order of $O(n^3)$ or higher.

2. Transition-based deterministic parsers, such as Nivre’s shift-reduce Malt parser [1]. These parsers successively apply the best possible local action in a deterministic way, using rich features that depend on the full tree produced so far. The final dependency tree is thus not globally optimum, and it is not possible to compute scores for other candidate trees. Despite these limitations, these parsers perform remarkably well and reach accuracy scores very close to those reached by the former class of parsers with a much lower complexity in $O(n)$, thanks to the possibility to use very rich features when choosing an action.

There has been a number of proposals to enhance transition-based parsers with probabilistic explorations of the trees space, but because of the dependency of features to the full history of actions, efficient dynamic programming algorithms such as Viterbi are not applicable. Hence, most such proposals rely on beam-search explorations [6], which can be viewed as a compromise between both types of parsers, by exploring with rich features a limited portion of the search space with a complexity of $O(n \log n)$. We propose next another such compromise that decomposes the parsing process into several successive steps, where the full search space is explored within each step, but with deterministic decisions taken after each step.

The proposed multi-stage parser is in some sense related to the algorithm described in [7], because the first dependencies that are created are most often the shortest, which are also the easiest to find. This is however not always true in our case, as the length of a newly created dependency arc actually depends on the size of the neighbor subtrees. Our proposed approach thus creates dependency arcs in an order that is neither left-to-right, nor easier-first, nor closest-first, but that tends to form constituent subtrees first and closest arcs first.

A number of other multi-pass parsing approaches have already been proposed in the literature. Hence, several left-to-right and right-to-left passes are realized in [8], but as a re-parsing process that exploits full dependency trees produced by previous parsers. A closer work is [9], which makes several left-to-right passes, but with a deterministic parser in each pass.

Extending deterministic transition-based parsers to make them explore several alternative solutions is an old idea [10], which has also been adapted to create probabilistic left-to-right parsers [11]. But this is only recently in [12] that this principle has gained success into bridging the gap between purely deterministic and generative probabilistic parsers. The authors of [12] thus propose to extend the classical shift-reduce algorithm into a best-first shift-reduce approach, and further exploit maximum entropy model to derive the probability of the winning tree. In a related work, the authors of [13] propose a beam-search best-first parsing algorithm that exploits an SVM classifier in each state for Chinese dependency parsing. The structured language model [14] also exploits a beam-search, but
within a fully probabilistic joint model of parsing and speech recognition. This generative Markov model can be viewed as a left-right bottom-up parsing algorithm. It is applied on phrase structure parses with relatively "poor" features. Although very good results are obtained with such beam-search approaches, we rather explore next unpruned generic Markovian models in each stage that can be tuned to address the desired compromise between search optimality and global features.

A more detailed review of recent dependency parsing advances can be found for instance in [5], and an excellent discussion on the different types of parsers in [15].

### 3. Proposed model

#### 3.1. Multi-stage parsing

In the proposed model, parsing is realized by iterating through multiple stages, where each stage takes as input a sequence of partial dependency trees, links together some of these trees and outputs a smaller sequence of larger trees. Initially, there are as many trees as words in the sentence. The algorithm ends when there is a single tree per sentence. Each stage is based on a Maximum Entropy Markov Model (MEMM) [16] that processes input trees sequentially and links together adjacent subtrees. Each state of the MEMM can perform three basic actions: link with dependency X the root of the current tree to a word of the next tree (RA_X), or to a word of the previous tree (L_A_X), or do not create any link (N.A). The topology of this MEMM is shown in Figure 1.

For training, oracle actions for all stages of all sentences are computed and pooled together, and the MEMM model is trained using the Generalized Iterative Scaling algorithm on this set of observations. For parsing, the Viterbi algorithm is used within each stage to compute the globally optimal sequence of actions, where the score of each path is the product of the posterior probabilities of the Maximum Entropy model, detailed in Section 3.2. The trees produced by the Viterbi algorithm are considered as fixed observations for the subsequent stages. When the best sequence only contains N.A actions, then the second best path is considered.

The Viterbi algorithm can be used within each stage if and only if every feature of the Maximum Entropy Model depends only on the arcs produced at the previous stages and not at the current stage, hence respecting the first-order Markov constraint. Finally, it is easy to check that the MEMM model can only create projective trees, thanks to the forbidden transition from RA to LA states.

This process is illustrated in Figure 2, where three stages are needed to parse the sentence. In practice, we have observed that the number of stages follows a logarithmic curve and never exceeds 6 on the training corpus. Furthermore, although the worst-case complexity of the proposed parser is in \( O(N^2) \) in function of sentence size, we have observed that its empirical average complexity is actually closer to \( O(N \log(N)) \).

#### 3.2. Maximum Entropy model

The joint transition and observation probability is computed in each state of the MEMM by a Maximum Entropy model. It takes as input the features shown in Table 1, which encode relevant information from the observed words and trees produced at the previous stages. This feature set is inspired by the features that are classically computed in transition-based parsers, such as in [1].

Most features require a target head word \( w_t \) to be identified, but the MEMM states only identify a target head tree \( n_t \). The selection of the head word \( w_t \in n_t \) is realized by an independent process that can be implemented for instance by testing every possible head word and choosing the one that maximizes the posterior of the Maximum Entropy model. We do not study this head selection process in greater details, because all the different solutions we have tested so far have about the same impact in terms of performances.

The output classes of the maximum entropy model are the labeled states of the MEMM.

<table>
<thead>
<tr>
<th>Feature class</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>From node ( n_t )</td>
<td>( \text{Form}(w_{t+1}) )</td>
</tr>
<tr>
<td>From node ( n_{t-1} )</td>
<td>( \text{POS}(w_t) )</td>
</tr>
<tr>
<td>From node ( n_{t+1} )</td>
<td>( \text{POS}(w_{t+1}) )</td>
</tr>
<tr>
<td>Node-independent</td>
<td>( \text{Form}(w_{t-1}) )</td>
</tr>
</tbody>
</table>

**Notations:** \( w_t \) is the root word of \( n_t \), \( w_{t-1} \) represents any word in \( n_{t-1} \). \( w_{t+1} \) that does not violate any projectivity constraint when linked to \( w_t \). Joint features are represented with a dash '-' between joint elements; \( \text{dist}(w, w') \in \{1, 2, 3+\} \) represents the number of words between \( w \) and \( w' \). stage represents the stage of the decoding process; it can take 6 values: 1 to 5, and 6+.

Table 1: Features of the MEMM

### 4. Experimental validation

#### 4.1. Experimental setup

All experiments are realized on a French broadcast news Treebank, the “Ester Treebank” (ETB), which has been manually annotated with dependencies [17]. The training and test corpus are respectively composed of 50,000 and 4,807 words. Every sentence is first processed automatically to compute the part-of-speech tags with the Tree Tagger [18]. Parsing evaluation is performed with the CoNLL-2006 scoring script. It reports the Labeled Attachment Score (LAS), which corresponds to the...
ratio of the number of words that have been assigned both a
correct head word and a correct dependency label.

The finite state machine code is re-implemented, while the
maximum entropy models are trained with 200 iterations of the
GIS algorithm with the OpenNLP library.

The proposed parser is compared with two state-of-the-art
parsers:

- The Malt parser [1] v.1.6.1 with default libSVM config-
  uration, which is one of the best transition-based parser.
- The MATE parser [2], which is one of the fastest and
  most efficient maximum spanning-tree parser.

4.2. Evaluation

The performances obtained with the proposed parser and the
state-of-the-art Malt and MATE parsers are shown in Figure 3.
The empirical complexity of the three systems, measured in
terms of computation time and memory consumption, is also
shown in Table 2.

<table>
<thead>
<tr>
<th>Parser</th>
<th>Training time</th>
<th>Test time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malt (libSVM)</td>
<td>1000 sec</td>
<td>107 sec</td>
</tr>
<tr>
<td>Malt (liblinear)</td>
<td>16 sec</td>
<td>3 sec</td>
</tr>
<tr>
<td>MATE</td>
<td>213 sec</td>
<td>41 sec</td>
</tr>
<tr>
<td>MEMM</td>
<td>283 sec</td>
<td>5 sec</td>
</tr>
</tbody>
</table>

Table 2: Comparison of practical running times and memory
consumptions of different systems for training and testing on
the Ester Treebank, on a computer with 4 cores and 8 Gb of
RAM. Time is expressed in seconds, and memory in Mb.

There is no statistically significant difference between the
proposed system and the Malt parser, and we can thus conclude
that the proposed approach gives comparable results to the state-
of-the-art supervised parsers. The MATE parser outperforms
both systems, which might be due to its more exhaustive ex-
ploration of the search space, but at the cost of a much higher
complexity.

5. Extension to higher ranks

One of the most interesting feature of the proposed framework
is the fact that it can address the full range of possible com-
promise between both extreme parsing paradigms presented in
Section 1, in short, local features with global search, and global
features with local search. Hence, with a minor modification
of the automaton in Figure 1, it is possible to force the parser to
produce a single dependency link per stage. This corresponds to
a purely deterministic easiest-first and shortest-first transition-
based parser, which approximates the Nivre-Eager algorithm
used in the Malt parser that follows a modified shortest-first
strategy but chunk-wise instead of sentence-wise.

On the other extreme, it is possible to extend the automaton
to generate longer dependencies, ultimately tending towards a
single-stage model. This however requires more complex modi-
fications of the automaton in order to still guarantee acyclic and
non-projective trees. This automaton $G(n)$ for rank $n$ is de-
defined recursively as in Figure 4. There are 2 non-emitting entry
states, respectively for paths that may create "backward" links
or not. Similarly, there are 2 non-emitting exit states for paths
that may create "forward" links. STR(n-1) and STL(n-1) both
represent the very same special sub-automaton that generates
all possible complete trees over exactly $n − 1$ observed words
$(w_0, \ldots, w_{n-1})$, and further links the root of the tree to either
the previous or next word, $w_{-1}$ or $w_n$. STR(n-1) and STL(n-1)
are named differently though because this sub-automaton has to
be duplicated at two different places in the network. We have
not found any straightforward recursive description of $ST^*(n-1)$
and we have thus written a program that automatically gen-
ernates and factorizes $ST(n-1)^2$. $G(1)$ is the automaton shown

---

1http://incubator.apache.org/opennlp

2The source code of the proposed parser is freely distributed under
an open-source licence and is accessible as part of the JSafran software
at http://rapsodis.loria.fr/jsafran/index.html
Figure 4: Recursive definition of the rank-\(n\) automaton \(G_n\). The output of \(\text{STR}_{n-1}\) is \(\text{outFwd}\), because \(\text{STR}_{n-1}\) follows a \(\text{RA}_n\) that points to the following node. The input of \(\text{STL}_{n-1}\) is inBck, because the following \(\text{LA}_n\) points back to the preceding node.

With this recursive definition, it is possible to create rank-\(n\) automata that generate dependency arcs of any length up to \(n\). The bigger \(n\) is, the smaller the required number of stages to parse any sentence is, tending ultimately towards a single-pass parser for sentences shorter than \(n\). In this case, thanks to the globally optimum probabilistic model, the model becomes equivalent to a maximum spanning-tree parser. The downside is that the number of states in the automaton increases exponentially with \(n\). But the objective is clearly not to replace state-of-the-art maximum spanning tree parsers, but rather to open a new exploration area that fills the gap between local features/global search and global features/local search.

6. Conclusion

In this work, an original stochastic dependency parsing framework is proposed that explores a new compromise between suboptimal models with global features and optimal models with local features. The parsing process is split up into several stages, with each stage being parsed into a globally optimal solution, thanks to very efficient dynamic programming algorithms. Deterministic decisions are taken after every stage, which on the one hand make the parser overall suboptimal, but on the other hand give the possibility to use and exploit global features that are derived from the dependency arcs created at the previous stages. By changing the rank of the automaton, the developer may tune and change the compromise between both extreme approaches at will.

In terms of performance, the rank-1 model tested here obtains similar accuracies than the Malt parser, and moderately lower scores than the MATE parser on the Estex Treebank, which is a French broadcast news corpus. Its mixed deterministic and Markovian processes makes both training and parsing very fast and greatly limits memory footprints. There are many ways to improve this parser. First, although we have chosen MEMM because of practical reasons, and in particular its low complexity and the possibility to easily modify the code, other finite state transducers, and in particular Conditional Random Fields, might be used. Second and higher-order Markovian models might also be considered in order to somehow replace the missing rich features that would exploit the dependencies created within a stage. Another interesting option may be to extend the deterministic process across stages with a graph of alternative solutions or with some kind of backtracking to limit the impact of errors from the previous stages. Finally, the underlying probabilistic model gives access to a wide range of powerful probabilistic algorithms, and in particular to marginalization and Bayesian inference for handling partial and missing observations.

7. Acknowledgments

This work has been partly funded by the CNRS PEPS SYFRAP project and the CRPP MISN for the Région Lorraine.

8. References