Dependency Parsing for Weibo: 
An Efficient Probabilistic Logic Programming Approach

William Yang Wang, Lingpeng Kong, Kathryn Mazaitis, William W. Cohen
Language Technologies Institute & Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA.
{yww, lingpenk, krivard, wcohen}@cs.cmu.edu

Abstract
Dependency parsing is a core task in NLP, and it is widely used by many applications such as information extraction, question answering, and machine translation. In the era of social media, a big challenge is that parsers trained on traditional newswire corpora typically suffer from the domain mismatch issue, and thus perform poorly on social media data. We present a new GFL/FUDG-annotated Chinese treebank with more than 18K tokens from Sina Weibo (the Chinese equivalent of Twitter). We formulate the dependency parsing problem as many small and parallelizable arc prediction tasks: for each task, we use a programmable probabilistic first-order logic to infer the dependency arc of a token in the sentence. In experiments, we show that the proposed model outperforms an off-the-shelf Stanford Chinese parser, as well as a strong MaltParser baseline that is trained on the same in-domain data.

1 Introduction
Weibo, in particular Sina Weibo\(^1\), has attracted more than 30% of Internet users (Yang et al., 2012), making it one of the most popular social media services in the world. While Weibo posts are abundantly available, NLP techniques for analyzing Weibo posts have not been well-studied in the past.

Syntactic analysis of Weibo is made difficult for three reasons: first, in the last few decades, Computational Linguistics researchers have primarily focused on building resources and tools using standard English newswire corpora\(^2\), and thus, there are fewer resources in other languages in general. Second, microblog posts are typically short, noisy (Gimpel et al., 2011), and can be considered as a “dialect”, which is very different from news data. Due to the differences in genre, part-of-speech taggers and parsers trained on newswire corpora typically fail on social media texts. Third, most existing parsers use language-independent standard features (McDonald et al., 2005), and these features may not be optimal for Chinese (Martins, 2012). To most of the application developers, the parser is more like a blackbox, which is not directly programmable. Therefore, it is non-trivial to adapt these generic parsers to language-specific social media text.

In this paper, we present a new probabilistic dependency parsing approach for Weibo, with the following contributions:

- We present a freely available Chinese Weibo dependency treebank\(^3\), manually annotated with more than 18,000 tokens;
- We introduce a novel probabilistic logic programming approach for dependency arc prediction, making the parser directly programmable for theory engineering;
- We show that the proposed approach outperforms an off-the-shelf dependency parser, as well as a strong baseline trained on the same in-domain data.

In the next section, we describe existing work on dependency parsing for Chinese. In Section 3, we present the new Chinese Weibo Treebank to the research community. In Section 4, we introduce the proposed efficient probabilistic programming approach for parsing Weibo. We show the experimental results in Section 5, and conclude in Section 6.

---

\(^1\)http://en.wikipedia.org/wiki/Sina_Weibo
\(^2\)For example, Wall Street Journal articles are used for building the Penn Treebank (Marcus et al., 1993).
\(^3\)http://www.cs.cmu.edu/~yww/data/WeiboTreebank.zip
2 Related Work

Chinese dependency parsing has attracted many interests in the last fifteen years. Bikel and Chi-ang (2000; 2002) are among the first to use Penn Chinese Tree Bank for dependency parsing, where they adapted Xia’s head rules (Xia, 1999). An important milestone for Chinese dependency parsing is that, a few years later, the CoNLL shared task launched a track for multilingual dependency parsing, which also included Chinese (Buchholz and Marsi, 2006; Nilsson et al., 2007). These shared tasks soon popularized Chinese dependency parsing by making datasets available, and there has been growing amount of literature since then (Zhang and Clark, 2008; Nivre et al., 2007; Sagae and Tsujii, 2007; Che et al., 2010; Carreras, 2007; Duan et al., 2007).

Besides the CoNLL shared tasks, there are also many interesting studies on Chinese dependency parsing. For example, researchers have studied case (Yu et al., 2008) and morphological (Li and Zhou, 2012) structures for learning a Chinese dependency parser. Another direction is to perform joint learning and inference for POS tagging and dependency parsing (Li et al., 2011; Hatori et al., 2011; Li et al., 2011; Ma et al., 2012). In recent years, there has been growing interests in dependency arc prediction in Chinese (Che et al., 2014), and researchers have also investigated character-level Chinese dependency parsing (Zhang et al., 2014). However, even though the above methods all have merits, the results are reported only on standard newswire based Chinese Treebank (e.g. from People’s Daily (Liu et al., 2006)), and it is unclear how they would perform on Weibo data.

To the best of our knowledge, together with the recent study on parsing tweets (Kong et al., 2014), we are among the first to study the problem of dependency parsing for social media text.

3 The Chinese Weibo Treebank

We use the publicly available µtopia dataset (Ling et al., 2013) for dependency annotation. An interesting aspect of this Weibo dataset is that, besides the Chinese posts, it also includes a copy of the English translations. This allows us to observe some interesting phenomena that mark the differences of the two languages. For example:

- Function words are more frequently used in English than in Chinese. When examin-
Fragmentary Unlabeled Dependency Grammar (FUDG) is a newly proposed flexible framework that offers a relative easy way to annotate the syntactic structure of text. Beyond the traditional tree view of dependency syntax in which the tokens of a sentence form nodes in a tree, FUDG also allows the annotation of additional lexical items such as multiword expressions. It provides special devices for coordination and coreference; and facilitates underspecified (partial) annotations where producing a complete parse would be difficult. Graph Fragment Language (GFL) is an implementation of unlabeled dependency annotations in the FUDG framework, which fully supports Chinese, English and other languages. The training set of our Chinese Weibo Treebank\(^4\) includes 14,774 tokens, while the development and test sets include 1,846 and 1,857 tokens respectively.

4 A Programmable Parser with Personalized PageRank Inference

A key problem in multilingual dependency parsing is that generic feature templates may not work well for every language. For example, Martins (2012) shows that for Chinese dependency parsing, when adding the generic grandparents and siblings features, the performance was worse than using the standard bilexical, unilexical, and part-of-speech features. Unfortunately, for many parsers such as Stanford Chinese Parser (Levy and Manning, 2003) and MaltParser (Nivre et al., 2007), it is very difficult for programmers to specify the feature templates and inference rules for dependency arc prediction.

In this work, we present a Chinese dependency parsing method for Weibo, based on efficient probabilistic first-order logic programming (Wang et al., 2013). The advantage of probabilistic programming for parsing is that, software engineers can simply conduct theory engineering, and optimize the performance of the parser for a specific genre of the target language. Recently, probabilistic programming approaches (Goodman et al., 2012; Wang et al., 2013; Lloyd et al., 2014) have demonstrated its efficiency and effectiveness in many areas such as information extraction (Wang et al., 2014), entity linking, and text classification (Wang et al., 2013).

Algorithm 1 A Dependency Arc Inference Algorithm for Parsing Weibo

Given:

(1) a sentence with tokens \(T_i\), where \(i\) is the index, and \(L\) is the length;
(2) a database \(D\) of token relations from the corpus;
(3) first-order logic inference rule set \(R\).

for \(i = 1 \rightarrow L\) tokens do
\[
S \leftarrow ConstructSearchSpace(T_i, R, D);
\]
\[
\overrightarrow{P}_i \leftarrow InferParentUsingProPPR(T_i, S);
\]
end for

Greedy Global Inference
for \(i = 1 \rightarrow L\) tokens do
\[
Y_i = \arg \max \overrightarrow{P}_i;
\]
end for

4.1 Problem Formulation

We formulate the dependency parsing problem as many small dependency arc prediction problems. For each token, we form the parent inference problem of a token \(T_i\) as solving a query \(edge(T_i, ?)\) using stochastic theorem proving on a search graph. Our approach relies on a database \(D\) of inter-token relations. To construct the database, we automatically extract the token relations from the text data. For example, to denote the adjacency of two tokens \(T_1\) and \(T_2\), we store the entry \(adjacent(T_1, T_2)\) in \(D\). One can also store the part-of-speech tag of a token in the form \(haspos(T_1, DT)\). There is no limitations on the arity and the types of the predicates in the database.

Given the database of token relations, one then needs to construct the first-order logic inference theory \(R\) for predicting dependency arcs. For example, to construct simple bilexical and bi-POS inference rules to model the dependency of an adjacent head and a modifier, one can write first-order clauses such as:

\[
edge(V_1, V_2) :-
adjacent(V_1, V_2), hasword(V_1, W_1),
hasword(V_2, W_2), keyword(W_1, W_2) \ #adjWord.
\]

\[
edge(V_1, V_2) :-
adjacent(V_1, V_2), haspos(V_1, W_1),
haspos(V_2, W_2), keypos(W_1, W_2) \ #adjPos.
\]

\[
keyword(W_1, W_2) :- \ # kw(W_1, W_2).
\]

\[
keypos(W_1, W_2) :- \ # kp(W_1, W_2).
\]

\(^4\)The corpus is freely available for download at the URL specified in Section 1.
Figure 2: After mapping the database $D$ to theory $R$, here is an example of search space for dependency arc inference. The query is $\text{edge}(S_1T_5, X)$, and there exists one correct and multiple incorrect solutions (highlighted in **bold**).

Here, we associate a feature vector $\phi_c$ with each clause, which is annotated using the # symbol after each clause in the theory set. Note that the last two (keyword and keypos) clauses are feature templates that allow us to learn the specific bi-POS tags and bilexical words from the data. In order for one to solve the query $\text{edge}(T_i, ?)$, we first need to map the entities from $D$ to $R$ to construct the search space. The details for constructing and searching in the graph can be found in previous studies on probabilistic first-order logic (Wang et al., 2013) and stochastic logic programs (Cussens, 2001). An example search space is illustrated in Figure 2. Note that now the edges in the search graph correspond to the feature vector $\phi_c$ in $R$.

The overall dependency arc inference algorithm can be found in Algorithm 1. For each of the parent inference subtask, we use ProPPR (Wang et al., 2013) to perform efficient personalized PageRank inference. Note that to ensure the validity of the dependency tree, we break the loops in the final parse graph into a parse tree using the maximum personalized PageRank score criteria. When multiple roots are predicted, we also select the most likely root by comparing the personalized PageRank solution scores.

To learn the more plausible theories, one needs to upweight weights for relevant features, so that they have higher transition probabilities on the corresponding edges. To do this, we use stochastic gradient descent to learn from training queries, where the correct and incorrect solutions are known. The details of the learning algorithm are described in the last part of this section.

### 4.2 Personalized PageRank Inference

For the inference of the parent of each token, we utilize ProPPR (Wang et al., 2013). ProPPR allows a fast approximate proof procedure, in which only a small subset of the full proof graph is generated. In particular, if $\alpha$ upper-bounds the reset probability, and $d$ upperbounds the degree of nodes in the graph, then one can efficiently find a subgraph with $O(\frac{1}{\alpha\epsilon})$ nodes which approximates the weight for every node within an error of $d\epsilon$ (Wang et al., 2013), using a variant of the PageRank-Nibble algorithm of Andersen et al. (2008).

### 4.3 Parameter Estimation

Our parameter learning algorithm is implemented using a parallel stochastic gradient descent variant to optimize the log loss using the supervised personalized PageRank algorithm (Backstrom and
<table>
<thead>
<tr>
<th>Method</th>
<th>Dev.</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford Parser (Xinhua)</td>
<td>0.507</td>
<td>0.489</td>
</tr>
<tr>
<td>Stanford Parser (Chinese)</td>
<td>0.597</td>
<td>0.581</td>
</tr>
<tr>
<td>MaltParser (Full)</td>
<td>0.669</td>
<td>0.654</td>
</tr>
<tr>
<td>Our methods — ProPPR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReLU (Bi-POS)</td>
<td>0.506</td>
<td>0.517</td>
</tr>
<tr>
<td>ReLU (Bilexical)</td>
<td>0.635</td>
<td>0.616</td>
</tr>
<tr>
<td>ReLU (Full)</td>
<td>0.668</td>
<td>0.666</td>
</tr>
<tr>
<td>Truncated tanh (Bi-POS)</td>
<td>0.601</td>
<td>0.594</td>
</tr>
<tr>
<td>Truncated tanh (Bilexical)</td>
<td>0.650</td>
<td>0.634</td>
</tr>
<tr>
<td>Truncated tanh (Full)</td>
<td>0.667</td>
<td>0.675*</td>
</tr>
</tbody>
</table>

Table 1: Comparing our Weibo parser to other baselines (UAS). The off-the-shelf Stanford parser uses its attached Xinhua and Chinese factored models, which are trained on external Chinese treebank of newswire data. MaltParser was trained on the same in-domain data as our proposed approach. * indicates $p < .001$ comparing to the MaltParser.

Leskovec, 2011). The idea is that, given the training queries, we perform a random walk with restart process, and upweight the edges that are more likely to end up with a known correct parent. We learn the transition probability from two nodes $(u, v)$ in the search graph using: $\Pr_w(v|u) = \frac{1}{Z} f(w; \Phi_{\text{restart}})$, where we use two popular non-linear parameter learning functions from the deep learning community:

- Rectified Linear Unit (ReLU) (Nair and Hinton, 2010): $\max(0, x)$;
- The Hyperbolic Function (Glorot and Bengio, 2010): $\tanh(x)$.

as the $f$ in this study. ReLU is a desirable non-linear function, because it does not have the vanishing gradient problem, and produces sparse weights. For the weights learned from $\tanh(x)$, we truncate the negative weights on the edges, since the default weight on the feature edges is $w = 1.0$ (existence), and $w = 0.0$ means that the edge does not exist in the inference stage.

5 Experiments

In this experiment, we compare the proposed parser with two well-known baselines. First, we compare with an off-the-shelf Stanford Chinese Parser (Levy and Manning, 2003). Second, we compare with the MaltParser (Nivre et al., 2007) that is trained on the same in-domain Weibo dataset. The train, development, and test splits are described in Section 3. We tune the regularization hyperparameters of the models on the dev. set, and report Unlabeled Attachment Score (UAS) results for both the dev. set and the hold-out test set. We experiment with the bilexical and bi-POS first-order logic theory separately, as well as a combined full model with directional and distance features.

The results are shown in Table 1. We see that both of the two attached pre-trained models from the Stanford parser do not perform very well on this Weibo dataset, probably because of the mismatched training and test data. MaltParser is widely considered as one of the most popular dependency parsers, not only because of its speed, but also the acclaimed accuracy. We see that when using the full model, the UAS results between our methods and MaltParser are very similar on the development set, but both of our approaches outperform the MaltParser in the holdout test set. The truncated $\tanh$ variant of ProPPR obtains the best UAS score of 0.675.

6 Conclusion

In this paper, we present a novel Chinese dependency treebank, annotated using Weibo data. We introduce a probabilistic programming dependency arc prediction approach, where theory engineering is made easy. In experiments, we show that our methods outperform an off-the-shelf Stanford Chinese Parser, as well a strong MaltParser that is trained on the same in-domain data. The Chinese Weibo Treebank is made freely available to the research community. In the future, we plan to apply the proposed approaches to dependency and semantic parsing of other languages.

Acknowledgements

We are grateful to anonymous reviewers for useful comments. This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program, and a Google Research Award. The authors are solely responsible for the contents of the paper, and the opinions expressed in this publication do not reflect those of the funding agencies.
References


