COMS E6998 Project: Unlabeled Non-projective Dependency Parsing

Mengqi Zong < *mz2326@columbia.edu* >

May 10, 2012

1 Outline

In this project, methods for dependency parsing have been reviewed.

For the problem of dependency parsing, we can formalize the weighted dependency parsing as searching for maximum spanning trees in directed graphs. The parsing algorithm of Eisner for projective dependency parsing has runs in $O(n^3)$. We will show that the non-projective dependency parsing using Chu-Liu-Edmonds algorithm will yield an $O(n^2)$ parsing algorithm. Experiments have been made using CONLL English data set. However, it turns out to be that for CONLL English data set, the results using Eisner algorithm are better than that using Chu-Liu-Edmonds algorithm.

2 Introduction

Dependency parsing has become a very interesting topic in the filed of machine learning recently. The primary advantage of using dependency structures other than lexicalized phrase structures is that dependency structures are more efficient to learn and prase while still encoding much of the predicate argument information needed in applications.

The dependency parsing we talk here is *dependency tree analyses*, in which each word depends on exactly one parent, either another word or a dummy root symbol. A dependency tree is *projective* if we put the words in their linear order, proceeded by the root, the edges can be drawn above the words without crossing. Otherwise, it is *non-projective*. Here is a projective dependency tree for sentence *John hit the ball with the bat*:

John	hit	the	ball	with	the	bat
2	0	4	2	2	7	5

The number below each word indicates the parent of this word in the dependency tree. For example, for word "the", the number 4 indicates the parent of word "the" is the 4^{th} word of the sentence, "ball". Also, number 0 represents the dummy root.

In English, projective trees are usually sufficient to analyze most sentence types. But there are certain types of English sentences that no projective-dependency tree can be drawn. For example, the sentence *John saw adog yesterday which was a Yorkshire Terrier*. In this case, non-projective dependency trees are preferred.

In this paper, we will talk about Ryan McDonald's approach (McDonald, 2005) of non-projective dependency parsing. The main idea of this approach is that dependency parsing can be formalized as the search for a maximum spanning tree in a directed graph. Using the spanning tree representation, we will use online-large margin discriminative training methods for non-projective dependencies.

Specifically, we will talk about an edge-based factorization of dependency trees and uses it to convert dependency parsing problem into finding maximum spanning trees in directed graphs. Then we will talk about the online-large margin learning framework used to train the dependency parsers.

Note that this entire paper will only focus on unlabeled dependency parsing, that is, the edges of the dependency trees are not partitioned into types of representing additional syntactic information such as grammatical function.

3 Dependency Parsing and Spanning Trees

3.1 Edge Based Factorization

Let $\mathbf{x} = x_1...x_n$ be a generic input sentence, $\mathbf{y}$ be a generic dependency tree for sentence $\mathbf{x}$. We will treat $\mathbf{y}$ as the set of tree edges, we write $(i, j) \in \mathbf{y}$ if there is a dependency in $\mathbf{y}$ from word x_i to word x_j.

We define the score of a dependency tree as the sum of the scores of all edges in the tree. And we define the score of an edge to be the dot product between a high dimensional feature representation of the edge and a weight vector,

$$s(i,j) = \mathbf{w} \cdot \mathbf{f}(i,j)$$

The score of a dependency tree $\mathbf{y}$ for sentence $\mathbf{x}$ is,

$$s(\mathbf{x},\mathbf{y}) = \sum_{(i,j)\in\mathbf{y}} s(i,j) = \sum_{(i,j)\in\mathbf{y}} \mathbf{w} \cdot \mathbf{f}(i,j)$$

$$s(i,j) = \mathbf{w} \cdot \mathbf{f}(i,j)$$

3.2 Maximum Spanning Trees

Let $G = (V, E)$ be a generic directed graph, and G consists of the vertex set $V = \{v_1, ..., v_n\}$ and set $E \subseteq [1:n] \times [1:n]$ of pairs (i,j) directed edges $v_i \rightarrow v_j$. Every edge has a socre $s(i,j)$. Note that $s(i,j)$ does not necessarily equal $s(j,i)$ since G is a directed graph.

A *maximum spanning tree* (MST) of G is a tree $\mathbf{y} \subseteq E$ that maximizes the value $\sum_{(i,j)\in\mathbf{y}} s(i,j)$ such taht every vertex in V appears in $\mathbf{y}$.

For each sentence $\mathbf{x}$ we define the directed graph $G_x = (V_x, E_x)$ given by

$$\begin{aligned} V_x &= \{x_0 = \text{root}, x_1, ..., x_n\} \\ E_x &= \{(i,j) : i \neq j, (i,j) \in [0:n] \times [1:n]\} \end{aligned}$$

It is clear that dependency trees for $\mathbf{x}$ and spanning trees for G_x coincide, since both kinds of trees are required to be rooted at the dummy root and reach all the words in the sentence. Hence, finding a dependency tree with the highest score is equivalent to finding a maximum spanning tree in G_x.

3.3 The Chu-Liu-Edmonds Algorithm

To find the highest scoring non-projective tree we simply search the entire space of spanning trees with no restrictions. And we will use the Chu-Liu-Edmonds algorithm for this. The Chu-Liu-Edmonds algorithm is shown in Figure 1.

Chu-Liu-Edmonds(G,s)
Graph G = (V,E)
Edge weight function $s : E \rightarrow \Re$

1. Let $M = \{x^*, x\} : x \in V, x^* = arg\max_{x'} s(x', x)$
2. Let $G_M = (V, M)$
3. If G_M has no cycles, then it is an MST:return G_M.
4. Otherwise, find a cycle C in G_M
5. Let $G_C = contract(G, C, s)$
6. Let $y =$ Chu-Liu-Edmonds(G_C, s)
7. Find a vertex $x \in C$ s.t. $(x', x) \in \mathbf{y}, (x'', x) \in C$
8. return $\mathbf{y} \cup C - \{(x'', x)\}$

contract$(G = (V, E), C, s)$

1. Let G_C be the subgraph of G excluding nodes in C
2. Add a node c to G_C representing cycle C
3. For $x \in V - C : \exists_{x' \in C}(x', x) \in E$
   Add edge (c, x) to G_C with
   $$s(c, x) = \max_{x' \in C} s(x', x)$$
4. For $x \in V - C : \exists_{x' \in C}(x, x') \in E$
   Add edge (x, c) tp G_C with
   $(x, c) = \max_{x' \in C}[s(x, x') - s(a(x'), x') + s(C)]$
   where $a(v)$ is the predecessor of v in C
   and $s(C) = \sum_{v \in C} s(a(v), v)$
5. return G_C

Figure 1: Chu-Liu-Edmonds algorithm for finding maximum spanning trees in directed graphs.

Informally, this algorithm greedily selects the incoming edge with highest weight. If a tree results, it must be the maximum spanning tree. If not, there must be a cycle. The procedure identifies a cycle and contracts it into a single vertex and recalculates edge weights going into and out of the cycle. And it has been proven that a maximum spanning tree on the contracted graph is equivalent to a maximum spanning tree in the original graph. Hence the algorithm can recursively call itself on the new graph.

Naviely, this algorithm runs in $O(n^3)$ time since each recursive call takes $O(n^2)$ to find the highest incoming edge for each word and to contract the graph. There are at most $O(n)$ recursive calls since we cannot contract the graph more than n times. But there is an efficient implementation of the algorithm with $O(n^2)$ time complexity for dense graphs by Tarjan (1977). So we will use this implementation here.

4 Estimating the Parameters

4.1 Online Large Margin Learning

In this section, we will talk about how to learn the weight vector $\mathbf{w}$. The basic idea is to extend the Margin Infused Relaxed Algorithm (MIRA) to learn dependency trees. The algorithm is shown in Fig. 2.

Basically, the online learning algorithm considers a single training instance at each update to $\mathbf{w}$. The final weight vector is the average of the weight vector after each iteration. This average effect could help overfitting (Collins, 2002).

For dependency trees, the loss of a tree is defined to be the number of words with incorrect parents relative to the correct tree. Note that for arbitrary inputs, there are typically exponentially many possible parses and thus exponentially many margin constraints in line 4 of Figure 2.

4.2 Single-best MIRA

For the exponential blow-up in number of trees, we can relax the optimization by using only the single margin constraint for the tree with the highest score, $s(\mathbf{x}, \mathbf{y})$. In this way, we can greatly decrease the time complexity of the MIRA. All we need to do is

to change the line 4 of Figure 2 into the following:

$$\begin{aligned}
&\min ||w^{(i+1)} - w^{(i)}|| \\
&\text{s.t. } s(\mathbf{x_t}, \mathbf{y_t}) - s(\mathbf{x_t}, \mathbf{y}') \geq L(\mathbf{y_t}, \mathbf{y}') \\
&\text{where } \mathbf{y}' = arg \max_{y'} s(\mathbf{x_t}, \mathbf{y}')
\end{aligned}$$

Also, McDonald et al. (2005) used an update with k constraints for the k highest-scoring trees. Though it has been shown that small values of k are sufficient to achieve the best accuracy for these methods, k best extensions to the Chu-Liu-Edmonds algorithm are too insufficient. We will show this by experiment in the later section of this paper.

Training data: $\mathcal{T} = \{(x_t, y_t)\}_{t=1}^{T}$

1. $\mathbf{w_0} = 0; \mathbf{v} = 0; i = 0$
2. for $n : 1...N$
3. for $t : 1...T$
4. $\min ||w^{(i+1)} - w^{(i)}||$
   s.t. $s(\mathbf{x_t}, \mathbf{y_t}) - s(\mathbf{x_t}, \mathbf{y}') \geq L(\mathbf{y_t}, \mathbf{y}'), \forall \mathbf{y}' \in \mathrm{dt}(\mathbf{x_t})$
5. $\mathbf{v} = \mathbf{v} + \mathbf{w}^{i+1}$
6. $i = i + 1$
7. $\mathbf{w} = \mathbf{v}/(N * T)$

Figure 2: MIRA learning algorithm.

4.3 Factored MIRA

Another solution for the exponential number of margin constraints is that we can factor the output by edges to obtain the following constraints:

$$\begin{aligned}&\min ||w^{(i+1)} - w^{(i)}|| \\ &\text{s.t.} s(l,j) - s(k,j) \geq 1 \\ &\forall (l,j) \in \mathbf{y_t}, (k,j) \notin \mathbf{y_t}\end{aligned}$$

In this way, the weight of the correct incoming edge to the word x_j and the weight of all other incoming edges must be separated by a margin of 1. And when all these constraints are satisfied, the correct spanning tree and all incorrect spanning trees are separated by a score at least as large as the number of incorrect incoming edges. This is because the scores for all the correct arcs cancel out, leaving only the scores for the errors causing the difference in over all score. And since each single error results in a score increase of at least 1, the entire difference must be at least the number of errors.

For a graph G with n nodes, there are $O(n^2)$ edges. That is, the number of constraints is $O(n^2)$. In this way, we factor the exponential number of margin constraints into a polynomial number of local constraints.

5 Experiments

We performed experiments on the CONLL English data set. We used the predefined training, development and testing split of this data set. This data sets consists of projective dependency trees. The goal of this experiment is to test how efficient the Chu-Liu-Edmonds non-projective MST algorithm could be for English.

In this experiment, we compared the following systems:

1. **McD Single-best:** The projective parser of McDonald et al. (2005) that uses Eisner algorithm for both training and testing. This system uses single-best MIRA.

2. **McD k-best:** The projective parser of McDonald et al. (2005) that uses Eisner algorithm for both training and testing. This system uses k-best MIRA with $k = 5$.

3. **Single-best MIRA:** We use Chu-Liu-Edmonds algorithm to find the best dependency tree for Single-best MIRA training and testing.

4. **k-best MIRA:** We use Chu-Liu-Edmonds algorithm to find the best dependency tree for k-best MIRA training and testing with $k = 5$.

5.1 Results

All results are shown in Table 1. There are two main metrics in the table: *Accuracy* and *Complete*. *Accuracy* measures the number of words that correctly identified their parent in the tree. *Complete* measures the number of sentences in which the resulting tree was completely correct. Also, *iteration* in Table 1 means the iteration time of updating $\mathbf{w}$. That is, the number of *iteration* is the T in Line 3, Figure 2.

Basically, the results show that for CONLL English data sets, training and testing with Chu-Liu-Edmonds algorithm is worse than using the Eisner algorithm. This should due to the fact that the fact that all dependency trees in the data set are projective dependency trees. And Eisner algorithm uses the a prior knowledge that trees are projective.

The results also indicate that a iteration time of 10 is enough for most cases, which indicates that the training converges very quickly. Also, with the increment of iteration time from 10, the *Accuracy* and *Complete* all tends to decrease (The only exception is McD k-best with iteration = 50). The decreases of performance should due to the overfitting effect.

Though the Chu-Liu-Edmonds non-projective MST algorithm has a parsing complexity of $O(n^2)$ and the projective Eisner algorithm has a complexity of $O(n^3)$, experiments showed that their running time are almost the same – not only for McD Single-best and Single-best MIRA, but also for McD k-best and k-best MIRA. This should due to the large constant factor of Chu-Liu-Edmonds non-projective MST algorithm.

At last, compared with its Single-best MIRA counterparts, the k-best MIRA does not give a better result. This is opposite to that of McD k-best and McD Single-best.

6 Summary

The Chu-Liu-Edmonds non-projective MST algorithm is a very promising algorithm for parsing non-projective dependency trees. This algorithm has a advantage of unifor-

Dependency Parsing Systems	Accuracy	Complete
McD Single-best with iteration = 10	**74.1**	**9.0**
McD Single-best with iteration = 20	72.9	7.5
McD Single-best with iteration = 30	72.8	8.5
McD Single-best with iteration = 40	72.5	8.0
McD Single-best with iteration = 50	72.3	8.5
McD k-best with iteration = 10	74.2	11.0
McD k-best with iteration = 20	74.1	11.0
McD k-best with iteration = 30	74.0	11.0
McD k-best with iteration = 40	74.2	11.5
McD k-best with iteration = 50	**74.3**	**12.0**
Single-best MIRA with iteration = 10	**72.4**	6.5
Single-best MIRA with iteration = 20	71.9	7.0
Single-best MIRA with iteration = 30	72.3	7.0
Single-best MIRA with iteration = 40	72.2	7.0
Single-best MIRA with iteration = 50	72.1	7.0
k-best MIRA with iteration = 10	**72.1**	**7.0**
k-best MIRA with iteration = 20	72.0	7.0
k-best MIRA with iteration = 30	71.9	7.0
k-best MIRA with iteration = 40	71.6	7.0
k-best MIRA with iteration = 50	71.5	6.5

Table 1: Dependency parsing results for English using spanning tree algorithms

mity and simplicity. It runs in $O(n^2)$ against the running time $O(n^3)$ of Eisner dynamic programming algorithm, which by construction enforces the non-crossing constraint.

However, results of experiments on CONLL English data set show that for projective-dependency tress, Eisner projective MST algorithm produces a better result than Chu-Liu-Edmonds non-projective MST algorithm. That is, for projective dependency structures, like English, the Eisner projective MST algorithm is still a better choice.

7 Reference

- Ryan McDonald, Fernando Pereira (2005). *Non-projective dependency parsing using spanning tree algorithms.*

- Ryan McDonald, Fernando Pereira (2005). *Online large-margin training of training of dependency parsers.*