CS 187 - Dependency Parsing
Phase 4: Final Paper Draft

Lauren Urke, Nathaniel Herman, Henrik Sigstad, Ariel Camperi

Abstract

The abstract ...

1 Introduction

- Motivation
- What we did and why we think it is important
- Statement of key result

2 Statement of the problem

- Further discussion about why we care...

3 Description of the method

3.1 Description of SVMs

3.2 Description of dependency parsing

3.3 Description of code

4 Description of data

To implement Dependency Parsing, we used the Penn Treebank Project Data. The treebank data contains sentences that have been parsed into a linguistic trees that also contain part-of-speech tagging.

Parsed in this way, "Pieere Vinken, 61 years old, will join the board as a non executive director Nov. 29.", will look like this:

NEED TO FIGURE OUT HOW TO TAB THIS

```
( (S
        (NP-SBJ
             (NP (NNP Pierre) (NNP Vinken) )
             (, ,)
             (ADJP
                    (NP (CD 61) (NNS years) )
                    (JJ old) )
              (, ,) )
        (VP (MD will)
             (VP (VB join)
                    (NP (DT the) (NN board) )
                    (PP-CLR (IN as)
                         (NP (DT a) (JJ nonexecutive) (NN director) ))
                    (NP-TMP (NNP Nov.) (CD 29) )))
        (. .) ))
```

This information is helpful, but could be improved. To conduct analysis on the dependencies, we wanted to be able to easily infer the relationship between two words. Therefore, we used a program from the Lund University Computer Science Department that converted the Penn Treebank parses into dependency parses. For each word in a sentence, a dependency parse indicates the parent word and the part of speech. For example, the same sentence now comes out as this:

ID	Token	Part of Speech	Parent ID	Class
1	Pierre	NNP	2	NAME
2	Vinken	NNP	8	SBJ
3	,	,	2	P
4	61	CD	5	NMOD
5	years	NNS	6	AMOD
6	old	JJ	2	APPO
7	,	,	2	P
8	will	MD	0	ROOT
9	join	VB	8	VC
10	the	DT	11	NMOD
11	board	NN	9	OBJ
12	as	IN	9	ADV
13	a	DT	15	NMOD
14	nonexecutive	JJ	15	NMOD
15	director	NN	12	PMOD
16	Nov.	NNP	9	TMP
17	29	CD	16	NMOD
18	.	.	8	P

This data is now much more helpful for determining whether there is a direct parental relationship between two words in a sentence. The Penn Treebank contains 24 sections.

We used sections 02-22 for training and section 23 for testing. Section 23 contains 2,416 sentences, which includes 56,684 words (49,892 without punctuation).

5 Results

5.1 Results

We used three measurements to evaluate the effectiveness of this method.

$$\text{Dependency Accuracy} = \frac{\text{number of correct parents}}{\text{total number of parents}}$$
$$\text{Root Accuracy} = \frac{\text{number of correct root nodes}}{\text{total number of sentences}}$$
$$\text{Complete Rate} = \frac{\text{number of complete parsed sentences}}{\text{total number of sentences}}$$

Using a default context of (2,2), we evaluated two different linear SVC methods. With a One vs One Linear SVC method, we obtained the following results:

Dependency Accuracy	Root Accuracy	Complete Rate
0.89135	0.93675	0.32711

Using a One vs Many Linear SVC, we obtained the following, slightly worse results:

Dependency Accuracy	Root Accuracy	Complete Rate
0.88680	0.93894	0.32285

The Sci-kit Learn offered several other SVM options, but after testing each of them with a set context length, we determined that none of them improved our results. We then empirically determined the optimal context length for the One vs One classifier. The Context determines how many words to the left and right of the target nodes are considered.

	(2,2)	(2,3)	(2,4)	(2,5)	(3,2)	(3,3)	(3,4)	(3,5)
Dep. Acc.	0.891	0.890	0.890	0.889	0.887	0.887	0.887	0.887
Root Acc.	0.937	0.928	0.934	0.930	0.932	0.929	0.928	0.927
Comp. Rate	0.327	0.330	0.325	0.326	0.312	0.318	0.316	0.317

We determined that the One vs One Linear SVC gave the best results when used with a context of 2 on either side of the target nodes.

6 Conclusion

Yamada and Matsumoto's implementation of dependency parsing used a One vs One classifier to parse the sentences. They tested different kernel degrees, and ended up with a quadratic kernel. They also determined that a context of 2 on either side of the target nodes was best. Their quadratic implementation gave only slightly better results than our linear implementation, while their linear implementation did worse than ours:

	Dependency Accuracy	Root Accuracy	Complete Rate
Y and M Quadratic	0.900	0.896	0.379
Y and M Linear	0.854	0.811	0.261
Our Linear Results	0.89135	0.93675	0.32711

Our implementation successfully modeled Yamada and Matsumoto's dependency parsing algorithm with fairly similar results on the linear kernel level. In theory, a One vs Many classifier should work better than a three-part One vs One classifier. Possible next steps to improve results include investigating other One vs Many classifiers and attempting to pinpoint why this classifier did not improve parsing accuracy. Another possible next step includes testing out various other kernel functions.