CS 6741: Final Project
Re-Implementing Statistical Dependency Analysis With Support Vector Machines

Rohit Jain
Cornell Tech
[rj288@cornell.edu]

Abstract

In this project, we studied and re-implemented the technique for dependency parsing proposed by Yamada et al. We recreated the results for the dependency trees converted from the Penn Treebank data. We also experimented with the biomedical data from the GENIA biomedical corpus(Yuka et al., 2005) and the Spanish universal dependency dataset(McDonald et al.,2013) to understand out of domain implications. The code is available on github: https://github.com/rohit-jain/parzer

1 Introduction

Many statistical parsers exist and perform very well but most of them require Penn Treebank style phrase structure annotations(Fig 1.). To take these parsers to different domains, Penn-style annotated data is required. This is unrealistic because it requires annotators to have good knowledge of deep linguistic theories, phrase structure rules in addition to the domain.

```
(ROOT
 (S
  (NP (NNP Rolls-Royce) (NNP Motor) (NNPS Cars) (NNP Inc.))
  (VP (VBD said)
   (SBAR
    (S
     (NP (PRP it))
     (VP (VBZ expects)
      (NP (PRP$ its) (NNP U.S.) (NNS sales))
      (S
       (VP (TO to)
        (VP (VB remain)
         (ADJP (JJ steady))
         (PP (IN at)
          (NP
           (QP (RB about) (CD 1,200))
           (NNS cars))))))))))
  (. .)))
```

Figure 1. Penn Treebank style phrase structure annotation

Alternatively word–word dependency structure rely on the relations between the words. It doesn't require phrase structure annotation so it could potentially make task of annotators to generate training data much easier and will also reduce the ambiguity. As shown in Fig 2. One word is identified as the head and the other word as modifier, bidirectional arrow indicates that there is a dependency. The most recent dependency structure trees also have labels for the edges. For this project we use unlabeled edges and the arrow points towards the head word.

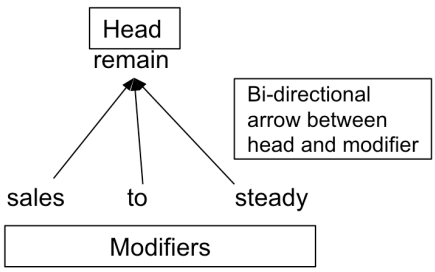

Figure 2. word-word dependency structure.

A parsing technique that can work robustly across different domains with less effort for generating training data is really exciting. Though it should be noted that with proper Penn Treebank style phrase structure annotations the predicition is very structured and it would be really difficult to do better than that without phrase structures.

In this project we experimented with the parsing technique proposed by Yamada et al not only on Penn Treebank data but on GENIA biomedical corpus(Yuka et al., 2005) and Spanish universal dependency dataset(McDonald et al.,2013) to better understand how the performance varies as the domain changes or the language changes. We also looked at how the choice of part of speech tagger can affect the performance of the parser if the tagger has not been trained for the domain.

In section 2 we go through the technique proposed in the original paper while listing the as-

sumptions or decisions we made at each step. In sections 3-5 we describe the experiments we recreated and the out of domain experiments with the results.

2 Approach

The paper by Yamada et al proposes a deterministic bottom up dependency parsing algorithm that uses a set of three parsing actions to construct the dependency tree. The dependency structure is learned using support vector machines(SVM)(Vladimir N. Vapnik, 1995). At each step output from the SVM is used to make the decision between parsing actions.

2.1 Parsing Actions

The parser proposed in the paper constructs the tree from left to right mainly using three parsing actions: Left, Right and Shift. These actions are applied to two neighboring words (referred to as target nodes). The actions details have been explained below.

Shift

This parsing action basically means there is no construction between the target nodes and we just shift the target nodes on the right. Figure 3 gives an example of the shift action.

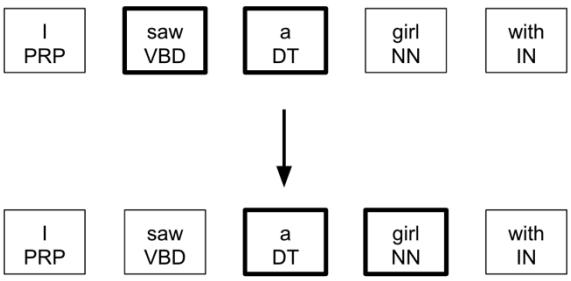

Figure 3: Shift action, the nodes before after the action

Left

The Left action constructs a dependency relation between target nodes where the right target node becomes a child of the left node. We shift the focal point for the next step to the left. The original paper does not specify what is the new focal point in case of left parsing action after execution and this is a decision we made. Eg. In Figure 4 'girl' becomes the child node of 'a' after the left parsing action is applied and the next target nodes are 'saw' and 'a'.

Right

A Right action constructs a dependency relation between target nodes where the left node of tar-

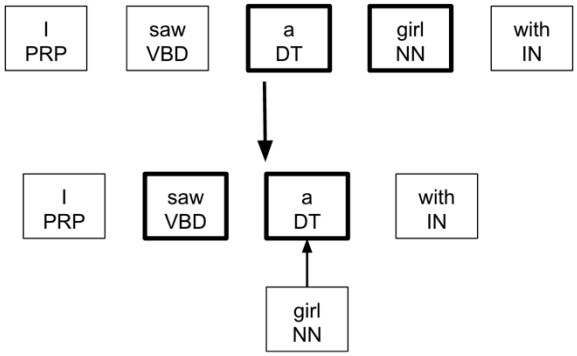

Figure 4: Left action: the nodes before and after the execution. 'girl' becomes the child and focus shifts to 'saw'.

-get nodes becomes a child of the right one. Figure 5 is an example of the action Right. After applying this action, "a" becomes a child of "girl" ("a" modifies "girl"). In this case the paper did propose moving the focal point to the left for the next step. Eg. 'saw' is the new left target node.

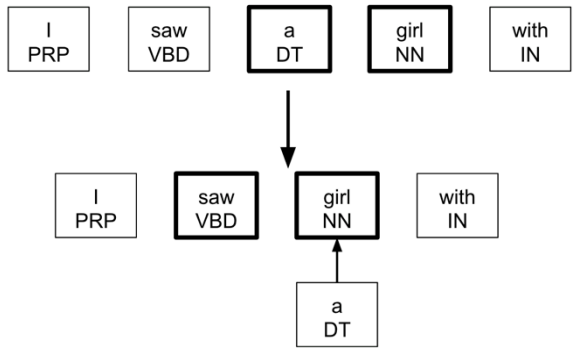

Figure 5: Right action: the nodes before after the action. 'a' becomes the child and focus shifts to 'saw'.

If the modifier node i.e. the node that is shifted in case of left/right parsing actions is not a complete subtree yet then we apply shift instead of applying the left/right parsing action. It is worth noting that the choice of new focal points in case of left and right parsing actions is an important decision to be made. It does not effect the accuracy but we should choose the policy that leads to the least number of parsing decisions because each decision is a sample input to the training algorithm and would affect the performance of the algorithm.

2.2 Parsing Algorithm

The algorithm uses contextual features from the target nodes and the nodes surrounding them to determine the appropriate parsing action. The action is executed to construct the dependency relation tree. Figure 6 gives a snapshot of the pseudo-code of the algorithm as given in the paper.

Input Sentence: $(w_1, p_1), (w_2, p_2), \cdots, (w_n, p_n)$
Initialize:
$i = 1$;
$\mathrm{T} = \{(w_1, p_1), (w_2, p_2), \cdots, (w_n, p_n)\}$
no_construction $= true$;
Start:
while $|\mathrm{T}| \geq 1$ **do begin**
if i == $|\mathrm{T}|$ **then**
if no_construction == $true$ **then break;**
no_construction $= true$
i = 1 ;
else
$\mathbf{x} = get\_contextual\_features(\mathrm{T}, i)$;
$y = estimate\_action(model, \mathbf{x})$;
$construction(\mathrm{T}, i, y)$;
if $y ==$ *Left* or *Right* **then**
no_construction $= false$;
end;
end;
end;

Figure 6: Parsing Algorithm

At initialization each node (word, POS tag) is considered as a tree in itself. The algorithm parses the set of trees from left to right considering two neighboring trees as the target nodes. The variable i in the algorithm is the index of the left target node in the tree set. At each step the algorithm uses get_contextual_features function to extract features from the target nodes and their neighboring nodes. The details of features as proposed in the paper have been described in section 2.3. The function estimate_action infers the parsing action based on the model that has been learned while training.

There are 2 terminating conditions for the algorithm. First, when the algorithm goes from i=0 to |T| with only shift parsing actions i.e. no construction happened. The variable no_construction helps to track that. Second, when there is only one tree left in T i.e. the sentence has been completely parsed. For implementation details of the data structures please refer to section 3 in the readme file.

2.3 Features

The features for learning are extracted from the target nodes as well as the nodes surrounding them. If i and i + 1 are the indexes of the left and right target nodes then the left context is defined as the nodes on the left side of the target nodes: t_l,(l < i), and the right context is defined as those on the right: t_r (i + 1 < r). Context lengths (l, r) represent the numbers of nodes within the left and right contexts and context window mean all nodes from i-l to i+r+1. Figure 7 shows an example of context lengths (2, 2). In this case, the target nodes are 'of' and 'resort', left context are "–" and "sellers", and the right context are "who" and "–". The paper uses POS tags and words themselves as the features for the target nodes and the nodes within their left and right context.

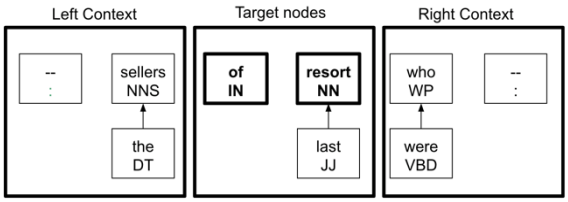

Figure 7: Example of context window with left and right context lengths of 2.

We also consider child nodes up to one level of hierarchy as part of the context window. If a node has multiple left or right children we consider all of them as part of the context and extract POS tags and word string features from them. Eg. If node 'were' had a child it wouldn't be considered part of the context window for the above target nodes. All the features are boolean one hot vectors since we are looking at the presence/absence of a word/POS tag string.

Type	Value
pos	Part of speech(POS) tag
lex	word string
ch-L-pos	POS tag of all left children nodes
ch-L-lex	Word string of all left children nodes
ch-R-pos	POS tag of all right children nodes
ch-R-lex	Word string of all left children nodes

Table 2: Summary of features used for learning

The dictionary for each position of the context is learned separately based on the training data. So, if a certain word string never appeared in right target node then it would treated as unknown in the feature vector for that position. This is not clear from the paper and an assump-

tion we made because otherwise the number of features is significantly different and much higher from the number in the paper. One other thing we tried was assuming all words that appear only once as unknown but the number of features is still very different. The code for this is in the extract_test_features method in parser/dependency_parser.py.

2.4 Learning from features

This is the estimate_action part of the algorithm presented in figure 4. Yamada et al used SVM as the learning algorithm. Mainly for 2 reasons: (1) High generalization performance in high dimensional feature spaces. (2) Learning with combination of multiple features is possible by virtue of polynomial kernel functions. SVMs can deal with non-linear classification using kernel functions. Especially use of the polynomial function $(x' \cdot x'' + 1)^d$ as the kernel, such an optimal hyperplane has an effect of taking account of combination of d features without causing a large amount of computational cost.

Since the classifier needs to make decision between shift, left and right parsing action this is a multi-class classification. When there are a large number of training sentences like in case of Penn Treebank standard training set (section 02 - 21), the number of training examples becomes about 1.6 million. Even though SVM's have the advantage of polynomial kernel functions, it is difficult to scale this and learn all training examples at once. Sparsity of the feature vectors is also an added disadvantage. To handle this the paper proposes grouping the training examples based on the POS tag of the left target node. For example, if the POS tag of the left target node is "VB" in analysis of test sentences, the parser would use SVM^{VB} to estimate the parsing action.

3 Experiments

We did experiments on three datasets.

Penn Treebank dataset: Sections 02-21 as training set, section 22 as dev and section 23 as the test data. The phrase structure trees were converted to dependency style trees using tool provided by Yamada et al that has rules similar to the Collins(1997) head rules. The original paper used Nakagawa's(2002) tagger for POS tagging the test sentences, but we used the Stanford POS tagger(Kristina et al., 2003) and GENIA tagger(Yoshimasa et al.,2005).

GENIA Corpus Treebank: The argument for dependency parsing is that it would work well across domains. Therefore, in addition to the penn Treebank we also experimented on GENIA Treebank that is a completely different domain from Penn Treebank. It contains 1999 medical abstracts and is completely different from newswire text. David Mc'Closky(2010) has converted the GENIA Treebank to Penn Treebank style trees available at http://nlp.stanford.edu/~mcclosky/biomedical.html, and that is what we used. We then used Stanford parser(Marie-Catherine et al.,2006) to convert Penn Treebank phrase structure trees to universal dependency trees. For tagging test sentences from the GENIA Treebank we try the stanford POS tagger and the GENIA tagger.

Spanish universal dependency dataset: With this dataset the goal is to see how the parser does on a different language all together. We used standard train, development and test sets for Spanish from the universal dependency project. For tagging test sentences here we only used Stanford POS tagger.

The Stanford tagger by default does not output the universal POS tag set(Petrov et al.,2005)(http://universaldependencies.github.io/docs/u/pos/index.html). It uses tags from AnCora 3.0 (http://clic.ub.edu/corpus/en) corpus and simplifies them further by nulling out fields (using a label 0) that are not relevant for them (http://nlp.stanford.edu/software/help/spanish-faq.shtml). Eg. AQ0MS0 and AQ0MSP are both simplified to AQ0000 as they are both adjectives. The tags though are still compliant with the EAGLES(http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html) standard. Petrov et al.(2011) provide mappings between EAGLES POS tags and the universal POS tags but they cannot be used directly (https://github.com/slavpetrov/universal-pos-tags/). To resolve this issue we did the mapping on our own and it's available in the code(converter/spanish_mapping).

4 Evaluation metric

The paper has proposed 3 evaluation metrics and ignores punctuation marks. It is unclear which punctuation marks are ignored. We ignore any word that has the POS tags in table 3 for the Penn Treebank and GENIA corpus dataset. This decision was again made based on the number of training features reported in the paper. It is a bit

confusing because symbol like '-' that also appear outside of mathematical contexts and in similar contexts to ':' symbol has not been ignored.
In case of Spanish since we have universal POS tags, the set changed to 'PUNCT' and 'SYM'.

Tag	Description
``	Opening quotation mark
,	Comma
:	Colon or ellipsis
.	Sentence terminator
''	Closing quotation mark

Table 3: Ignored punctuation tags. Descriptions: https://www.comp.leeds.ac.uk/ccalas/tagsets/upenn.html

The metrics proposed are as follows:
Dependency Accuracy: This is the ratio of number of nodes that were assigned correct parents to the total number of parents. Here, we don't ignore punctuation mark if it is the head node.

Root Accuracy: number of trees that had the correct roots. This is calculated as ratio of correct root nodes to the total number of sentences. Here, we ignore the root if its POS tag is a punctuation tag.

Complete Rate: This is the number of trees that were parsed completely correctly. This is computed as the ratio of completely correctly parsed sentences to the total number of sentences.

5 Results

Yamada et al did a lot of experimentation with different sizes of the context window and combination of features. For the purpose of reimplementation we used left context length of 2 and right context length of 4 and all the features that have been are there in table 2 for every node in context window. This setting according to Yamada et al gave them the best results.

First, we tried to recreate the results with the linear and polynomial kernels for the SVM with Stanford tagger. The number of features is 420540 and the hyper parameters we used for the SVM are gamma (kernel coefficient)=1 and constant (independent term in kernel function)=1.

	(a)	(b)
Root Acc.	0.909	**0.916**
Dep. Acc.	0.898	**0.903**
Comp. Acc.	0.372	**0.384**

Table 4: Dependency accuracies where Kernel function is $(x' \cdot x'' + 1)^2$ i.e. d = 2 and the context lengths are (2,4). Columns show the following results (a) our parser and (b) Yamada et al.

	(a)	(b)
Root Acc.	**0.833**	0.811
Dep. Acc.	**0.856**	0.854
Comp. Acc.	**0.264**	0.261

Table 5: Dependency accuracies where Kernel function is $(x' \cdot x'' + 1)$ and the context lengths are (2,4). Column (a) shows our results and column (b) are the results from Yamada et al.

Table 4 and 5 show that the results reported by Yamada et al are slightly better where d=2 and other way round for d=1. The difference though not very high, could have been a result of using different POS tagger or maybe the hyper-parameters of the SVM kernel used by the paper are different.

We also tried adding the last action as a feature to improve accuracy but didn't help. We also tried without child features of left and right contexts for the sake of training speed but the drop in accuracy was pretty significant.

Next, we use the same settings (d=2, l=2, r =4, gamma =1, constant = 1) but with GENIA tagger for tagging the test sentences instead of the Stanford tagger. We can see in table 6 that the root accuracy drops significantly. One of the reasons for this is that some of the tags produced by GENIA tagger eg. 'HYPH' are not their in the training set and we are not really giving classifier unknown examples while training to learn from.

	(a)	(b)
Root Acc.	**0.909**	0.885
Dep. Acc.	**0.898**	0.867
Comp. Acc.	**0.372**	0.366

Table 6: Dependency accuracies where Kernel function is $(x' \cdot x'' + 1)^2$ and the context lengths are (2,4) showing comparison between (a)PTB + GENIA tagger and (b)PTB + Stanford tagger.

For the different domain we try the parser on GENIA biomedical corpus. We try both taggers with it and the results have been summarized in

table 7. There are 196455 features using all the features in table 2. There are 2 main observations to be made. (1) The dependency accuracy and complete accuracy both drop as compared to the results on the Penn Treebank for both taggers. This can be mainly attributed to the domain transfer. Though for some reason the root accuracy increased. (2) The performance on all metrics drops very sharply in case we use Stanford POS tagger for GENIA corpus. The results are complete opposite of what we saw in case of Penn Treebank and GENIA tagger. This is mainly because the Stanford tagger has not been trained on biomedical data while GENIA tagger has been. This gives an interesting example of how tagger performance and domain that it has been trained on affects the results of parsing algorithm.

	(a)	(b)	(c)	(d)
Root Acc.	0.885	**0.916**	**0.909**	0.835
Dep. Acc.	**0.867**	0.801	**0.898**	0.774
Comp. Acc.	**0.366**	0.247	**0.372**	0.14

Table 7: Dependency accuracies where Kernel function is $(x' \cdot x'' + 1)^2$ and the context lengths are (2,4). Columns show the following results: (a) PTB + GENIA tagger, (b) GENIA Corpus + GENIA tagger, (c) PTB + Stanford tagger and (c) GENIA Corpus + Stanford tagger.

Another thing to note is that we are ignoring punctuation tags for evaluation metrics but in case of biomedical text the tags there significance could be completely different. Or there might be other tags that should be ignored. Eg. In the sentence "Rather , IFN-gamma antagonized the effect of IL-4 and suppressed the DC and MGC formation induced by GM-CSF + IL-4 and M-CSF + IL-4 , respectively ." The symbol "+" hold a very different significance from general English sentences and would also appear more often.

Next set of results(Table 8) for the Spanish language universal dependency dataset. We used only Stanford tagger for this because GENIA tagger doesn't have a pre-trained model for Spanish. The pre-trained model that Stanford POS tagger uses outputs the POS tags from AnCora 3.0 corpus, but they further simplified it to remove redundancy as mentioned earlier. For the purpose of this experiment we manually created this mapping between Stanford tag set and the universal POS tag for Spanish. It is possible that this mapping is not complete. There were some tags eg. 'PART' that were present in the training set but not there in the mapping. But the instances where these tags came into decision making while parsing were very few and probably wouldn't affect accuracy that much. But any errors in mapping any major tag eg. 'PRON'- proper noun would be very high.

We see a very sharp drop in all the accuracies and performance is even worse compared to GENIA corpus. One possible reason for this could be the bad POS tagging or issues with the manual mapping of tags. Another reason could be the difference in structures of Spanish and English. We have also ignored the 'PUNCT' and 'SYM' POS tags as punctuation during evaluation, but similar to the case of GENIA corpus we don't know the significance of these tags in Spanish.

	(a)	(b)	(c)
Root Acc.	0.708	**0.909**	0.835
Dep. Acc.	0.758	**0.898**	0.774
Comp. Acc.	0.116	**0.372**	0.14

Table 8: Dependency accuracies where Kernel function is $(x' \cdot x'' + 1)^2$ and the context lengths are (2,4). Column shows results for (a) Spanish + Stnford(Spanish) tagger, (b) PTB+Stanford tagger and (c) GENIA+Stanford tagger.

6 Conclusion

We observe that it is possible to achieve reasonable accuracy without phrase structure annotations. Accuracy is low compared to MEIP(Charniak, 2000) - 92.1% and Collins(1997) - 91.5%, but 90% is great considering the algorithm didn't use phrase structures.

The out of domain experiments are somewhat encouraging for GENIA corpus but still far from ideal. We still need to compare these results to the parsing accuracies achieved by other parser on GENIA. The parser does fail badly when we switch languages, but that's probably because of the structure and needs more experimentation. We also saw some interesting performance changes as we tried part of speech taggers trained on different domain data.

7 Future Work

The future work would include experimenting with other languages from the universal depend-

ency datasets and see if there is some language where the parser is able to maintain the performance. This would also require some work on handling projective dependencies with the algorithm or maybe come up with a different evaluation metric in those cases.

It is possible that we might be able to achieve better results with different combination of features, context length and classifier hyper-parameters. Trying different classifiers is also an option. It would also be interesting to explore is the parsing actions used. Right now if the target node for left/right parsing action is not a complete sub-tree shift is applied to it.

References

Collins, Michael. Three Generative, Lexicalised Models for Statistical Parsing. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (jointly with the 8th Conference of the EACL), pages 16–23, 1997.

Charniak, Eugene. A Maximum-Entropy-Inspired Parser. In Proceedings of the Sec- ond Meeting of North American Chapter of Association for Computational Linguistics (NAACL2000), pages 132–139, 2000.

David McClosky. 2010. Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing. Ph.D. thesis, Department of Computer Science, Brown University.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL 2003*, pp. 252-259.

Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *LREC 2006*

McDonald, Ryan T., et al. "Universal Dependency Annotation for Multilingual Parsing." *ACL (2)*. 2013.

Nakagawa Tetsuji, Taku Kudoh, and Yuji Matsumoto. Revision learning and its application to part-of-speech tagging. In Proceedings of Association for Computational Linguistics, pages 497–504, 2002.

Petrov, Slav, Dipanjan Das, and Ryan McDonald. "A universal part-of-speech tagset." *arXiv preprint arXiv:1104.2086* (2011)

Vladimir N. Vapnik. The Nature of Statistical Learning Theory. New York, 1995.

Yamada, Hiroyasu, and Yuji Matsumoto. "Statistical dependency analysis with support vector machines." Proceedings of IWPT. Vol. 3. 2003.

Yoshimasa Tsuruoka and Jun'ichi Tsujii, Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data, Proceedings of HLT/EMNLP 2005, pp. 467-474.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii, Developing a Robust Part-of-Speech Tagger for Biomedical Text, Advances in Informatics -10th Panhellenic Conference on Informatics, LNCS 3746, pp. 382-392, 2005

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta and Jun'ichi Tsujii, Syntax Annotation for the GENIA corpus, In the Proceedings of the IJCNLP 2005, Companion volume, pp. 222-227.