Parsing with Unlexicalised Probabilistic Context Free Grammar

Chirag Nagpal
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania
chiragn@cs.cmu.edu

Abstract

CKY is one of the most popular dynamic programming algorithms used to parse text, given a certain Context Free Grammar. CKY can also be extended easily to parse Probabilistic Context Free Grammar, and hence a popular choice to generate parse trees to determine phrase structure in Natural Language text. In this study, CKY is implemented in order to parse Natural Language text from the Wall Street Journal corpus.

1 Introduction

CKY is a dynamic programming approach to parse text from Natural Language data. Given enough training dataset, CKY is able to perform reasonably fast and accurate parsing. In this study, we implement a CKY to parse sentences from the Penn Tree Bank Corpus. We first generate the Probabilistic grammar rules from the Penn Tree Bank corpus files 200-2199. We then test our model on enetences from files 2200-2299 .

Inorder to report results, the parser is tested for two tasks.

- Training Data consists of all sentences, Test Data consists of sentences with length of at most 40 words.
- Training and Test Data consists of sentences at most length of 15 words.

2 Corpus Statistics

	Tr=15,Te=15	Tr=inf,Te=40
Training Trees	9753	39832
Test Trees	421	1578

Table 1: Corpus Statistics

Tr.	Te.	No of Trees	Total time	Avg. Time
1000	40	1578	888817	563.26
1000	20	725	107161	147.80
1000	15	421	33250	78.98
1000	10	153	4328	28.29

Table 2: Decoding Time Vs. Sentence Length

From the table it is evident that the decoding is varies polynomially with respect to the the sentence size. As CKY is a dynamic program, this is an expected behaviour.

3 No Markovisation

We first experiment with a probabilistic grammar without any markovisation, that is that our grammar stores only the parent and child of the rule.

	Prec	Rec	F1	Time
Tr=15, Te=15	73.08	63.88	68.18	904
Tr=inf, Te=40	64.63	52.58	57.99	25036

Table 3: No Markovisation h=0, v=1

4 Lossless Markovisation

In order to provide context to the grammar,Parsing is carried out with lossless markovisation. This refers to a markovisation in which the right sibling retains the preterminal of its left sibling indefinitely.

	Prec	Rec	F1	Time
Tr=15, Te=15	78.84	70.15	74.24	2245
Tr=inf, Te=40	69.6	59.73	64.29	112797

Table 4: Lossless Markovisation

As compared to no markovisation, lossless markovisation takes longer time, this is explained by the increase in the number of symbols and rules in the grammar. However, lossless binarisation significantly improves the F1 score as compared to no annotation.

5 Horizontal and Vertical Markovisation

In order to improve the statistical support for the symbols in the Grammar, we experiment with different values for the level of horizontal and vertical markovisation. This allows us to alter the granularity of our grammar, and hence improve the parsing.

	Prec	Rec	F1	Time
Tr=15, Te=15	83.73	79.99	81.81	7875
Tr=inf, Te=40	77.27	72.74	74.93	888817

Table 5: Markovisation h=2, v=2

The performance for h=2, v=2 is significantly improved as compared to Lossless markovisation, running time also increases significantly. One would expect the running time to not increase as the number of symbols are lesser as compared to Lossless Markovisation, however in case of lossless markovisation, most symbols and rules had nulls, so they could be pruned, reducing the running time.

6 Tag Splitting

Certain Preterminal symbols like ’IN’, ’CC’ give rise to only very distinct kind of words. (For example, the tag ’IN’ can give rise to words like ’of’, ’like’, ’as’ etc.) Depending on the word they generate, the grammar can be improved by annotating the preterminal symbols corresponding to its leaf. This is known as ’Tag-splitting’. We explicitly perform this splitting in our grammar for the three symbols, ’IN’, ’CC’ and ’AUX’.

	Prec	Rec	F1	Time
Tr=15, Te=15	83.52	80.43	81.95	7754
Tr=inf, Te=40	78.96	74.34	76.58	869646

Table 6: Markovisation h=2, v=2 + Tag Splitting

For both the cases, Tag Splitting has a considerable improvement in performance in terms of F1 score. It is interesting to note that for the Tr=15, Te=15 case, Tag-Splitting leads to a decrease in Precision, this maybe explained by the lack of statistical support for Tag-Splitting in this cases. However even with this reduction in Precision, the Recall increasing, inreasing the overall F1 Score.

7 Internal Rule Annotation

We further perform annotation by annotating all Nonterminal symbols which generate only one child. This is known as internal annotation.

	Prec	Rec	F1	Time
Tr=15, Te=15	83.7	80.61	82.13	8016
Tr=inf, Te=40	79	74.37	76.61	859406

Table 7: Markovisation h=2, v=2 + Tag Splitting

In both tests, Unary annotation increased the Precision and Recall leading to a significant increase in the overall F1 score. The running time for Tr=15, Te=15 increases, which is expected, as unary annotation would lead to an increase in the number of symbols in our grammar.

However, for Tr=inf, Te=40 the running time decreased, since the running time depends on the current system load of the testbed, we attribute this decrease in running time to chance.

8 Conclusion

We implemented an unlexicalised CKY parser with various granularities of markovisation. We further extended the grammar to add certain Tag Splitting annotations and Unary rule annotation. In order to improve decoding time the following strategies were used

- Iterate over only the parents, indexed by the child for the Binary Rules.
- Storing both the Unary and Binary pointers as arrays, as array lookups are faster than other specialised data structures.
- Prune iterations over certain rules, if score is less than a certain threshold. This greatly increases speed of decoding, however at a cost of some F1 score. Thus there is a trade off associated with this step. However this can be made up for by intelligent markovisation strategies and annotations like Tag-Splitting, as demonstrated.
- In the current implementation, the tree construction that is carried out is done using a unary rule in the beginning, the use of a binary rule, may improve or worsen the F1 score with respect to the goldset.

Acknowledgments

We would like to convey our gratitude to Professor Taylor Berg-Kirkpatrick, Wanli Ma and Kartik Goyal for all the help for this assignment.

References

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.