11-711 Project 2

Hengruo Zhang
Electrical & Computer Engineering
Carnegie Mellon University
5000 Forbes Avenue
hengruoz@andrew.cmu.edu

1 Introduction

In this project, we implement a CKY parser without coarse-to-fine pruning and evaluate it with different configurations on Penn TreeBank dataset. We got 79.53 F1 score using less than 2GB memory and 487 seconds. Actually, we got a higher score (81.54 F1) in a much longer time (1420s) and we didn't adopt that implementation.

The structure of this project is as follows:

- Section 2 introduces the implementation of submitted version.
- Section 3 introduces our experiments, which also explain why we adopted the implementation in Section 2.
- Section 4 provides error analysis.

2 Implementation

2.1 Pre-processing

Tag Splitting

We splitted "IN" tags into prepositions and subordinate conjunctions.

Binarization

We annotate our syntax trees by order 2 Vertical Markovization and order 2 Horizontal Markovization. In particular, we don't add parent information onto virtual nodes, i.e., the nodes in the form like "@$X \to Y$". For saving more memory, we implement our own binary tree class.

With the above implementation, the numbers of rules are listed in Table 1

B-rule	U-rule	Lexicon
7174	9297	44389

Table 1: Sizes of rules

Tally and Scoring

We almost totally use the classes in the given JAR file and modified the scoring strategy in Grammar and SimpleLexicon. Our scoring strategy is as follows:

Denote $X \to YZ$ a binary rule, $X \to Y$ a unary rule, $X \to a$ a lexicon rule, NT the set of all non-terminals, T the set of all terminals, $V = NT \bigcup T$, $S(\bullet)$ the score of $\bullet$,and $N(\bullet)$ the number of $\bullet$.

$$S(X \to YZ) = \log \frac{N(X \to YZ)}{N(X)}$$
$$S(X \to Y) = \log \frac{N(X \to Y)}{N(X)}$$
$$S(X \to a) = \log(\frac{N(X \to a)}{N(X)} \frac{|V|}{|V| + d_1} \frac{N(a) + d_2}{N(a)})$$

where $d_1 = 1.0$ and $d_2 = 0.75$.

2.2 CKY Parsing

We stored the intermediate scores in two 3d-arrays, one for unary rules and the other for binary rules. Our recursion formulae are listed as follows:

$$\mathbf{BS}[i][j][X] = \max_{X \to YZ} \max_{k \in [i,j]} S(X \to YZ) + \mathbf{US}[i][k][Y] + \mathbf{US}[k+1][j][Z]$$
$$\mathbf{US}[i][j][X] = \max_{X \to Y} S(X \to Y) + \mathbf{BS}[i][k][Y]$$

where **US** is the 3d-array for unary rules, **BS** is the 3d-array for binary rules. Besides, we also stored lexicon rules in $\mathbf{BS}[i][i][X]$ since all lexicon rules are in unary form and we also need to use US to store unary closures of lexicon rules.

To reconstruct the syntax trees, we store backtrack points in four 3d-arrays: **uBacks** for unary rules, **bSplitBacks** for the index to distinguish the left child and the right child (assume l is the list of words and $\mathbf{bSplitBacks}[i][j][X] = s$, $l[i : s]$

should be constructed as the left child and $l[s:j]$ should be the right child. Note that $l[i:j]$ means words from inclusive end i to inclusive end j), **bLeftBacks** and **bRightBacks** for the labels of the left child and the right child respectively.

3 Performance

3.1 Reducing Count

We tried different d_1 and d_2. By the results in Table 2, reducing count won't affect the performance much. Also, we applied reducing count to both unary and binary rules, too, and that would result in slightly negative effects.

d1	**d2**	**F1**
0.0	0.0	79.21
0.25	0.25	79.34
0.75	0.75	79.40
1.0	1.0	79.50
1.25	1.25	79.18
0.75	1.0	79.51
1.0	0.75	79.47
0.25	1.0	79.22
1.0	0.25	79.19

Table 2: Reducing count d1 and d2

3.2 Vertical & Horizontal Markovization

We tried different Markovization strategies and the results are listed in Table 3. Note that "p" means whether annotating virtual nodes (Y) or not (N). From this table, we know that vertical Markovization improves F1 scores much more and annotating virtual nodes can improve F1 scores but consume much more time. Besides, infinite horizontal Markovization can impair F1 scores because it generates too many labels and makes the average number of samples per label too small.

3.3 Tag Splitting

We split words in "IN" POS tags into two categories: prepositions and subordinate conjunctions. After splitting, we got 0.23 more F1 score.

3.4 Memory Saving

We stored all intermediate scores in one 3d-array at first because for both unary and binary rules, they use just half part of their score array. However, it saving limited memory and the index conversion slows down decoding time so we gave up

Setting	**F1**	**Time(s)**
v=1,h=0,p=N	70.23	230
v=1, h=1, p=N	71.59	273
v=1, h=2, p=Y	74.57	331
v=2, h=1, p=N	77.38	334
v=2, h=1, p=Y	77.94	621
v=2, h=2, p=N	79.51	487
v=2, h=2, p=Y	81.54	1420
v=2, h=∞, p=N	79.25	1537
v=2, h=∞, p=Y	79.34	1641

Table 3: Markovization strategies

this optimization. Also, we tried build our own counters and rule data structures but they are not significantly memory efficient so we abandoned them.

4 Error Analysis

4.1 Maximum Length of Test Sentences

Intuitively, the longer a sentence is, the harder it can be parsed. Our experiment verified this intuition. By Table 4, it's obvious that F1 scores increase first and decrease later with the maximum length of test sentences. There are three reasons why the scores are pretty low for much shorter sentences:

1. There are many polysemous words which are very hard to be parsed without a concrete context. We found many failed tests contain words like "right", "that", "left", and so on, which are all have many meanings.

2. There are fewer tests in such a short length so the results are easily affected.

3. There are many weak-grammatical phrases which can be parsed in different orders. We will illustrate it more detailedly later.

Length	**F1**	**Length**	**F1**
5	76.24	25	82.17
10	85.79	30	81.22
15	87.84	35	80.23
20	84.26	40	79.51

Table 4: F1 for different max lengths

4.2 Weak-grammatical Phrases

Weak-grammatical phrases are usually compound nouns like ”CMU bus stop” and parallel adjectives like ”black big spotty”. For example, ”Phyllis Kyle Stephenson Newport News” can be parsed into ”(NP (NNP Phyllis) (NNP Kyle) (NNP Stephenson) (NNP Newport) (NNP News))” or ”(NP (NNP Phyllis) (NNP Kyle) (NNP Stephenson)) (NP (NP (NNP Newport) (NNP News)))” but there is only one right answer.