Writeup

Ethan Hartzell

Problem Set 4

ehartz01

Preprocessing:

A built a static vocabulary using the function BuildVocab which picks out words that don't occur more than once and returns a set.

Then, we replace tokens not in the vocab with the unknown token.

The first trees of the training and test sets are printed below as output by the program:

first training set tree:

(S (NP (NP (NNP <UNK>) (NNP Vinken)) (NP|<,-ADJP-,> (, ,) (NP|<ADJP-,> (ADJP (NP (CD 61) (NNS years)) (JJ old)) (, ,)))) (S|<VP-.> (VP (MD will) (VP (VB join) (VP|<NP-PP-NP> (NP (DT the) (NN board)) (VP|<PP-NP> (PP (IN as) (NP (DT a) (NP|<JJ-NN> (JJ nonexecutive) (NN director)))) (NP (NNP Nov.) (CD 29)))))) (. .)))

first test set tree:

(S (NP (NP (JJ <UNK>) (NN trading)) (PP (IN during) (NP (DT the) (NN session)))) (S|<VP-.> (VP (VBD was) (VP (VBN <UNK>) (PP (ADVP+RB largely) (PP|<TO-NP> (TO to) (NP (NP (DT a) (NN round)) (NP|<PP-PP-,-SBAR> (PP (IN of) (NP (NN buy) (NNS programs))) (NP|<PP-,-SBAR> (PP (IN near) (NP (DT the) (NN close))) (NP|<,-SBAR> (, ,) (SBAR (WHNP+WDT which) (S+VP (VBD helped) (S+VP (VB offset) (NP (NP (DT the) (NN impact)) (NP|<PP-PP> (PP (IN of) (NP+NN profit-taking)) (PP (IN among) (NP (JJ blue) (NNS chips))))))))))))))) (. .)))

Training:

To train, we loop through each tree and adds the result of calling .productions() to an array. Then we call induce_grammar() on the array.

There are 1539 productions for the NP nonterminal.

The ten most probable are:

[NP -> DT NN, NP -> NP PP, NP -> NNP NNP, NP -> DT NP|<JJ-NN>, NP -> JJ NNS, NP -> DT NNS, NP -> JJ NN, NP -> NN NNS, NP -> DT NP|<NN-NN>, NP -> NP+NNS PP]

Testing:

3.1

To build the index we loop through the PCFG's productions and sort them into dictionaries for lexical and nonlexical items that map right hand sides to full productions.

Printing the index results in the file called index.txt

3.2

To implement the parse method we build two tables: the dynamic CKY table and the table of backpointers.

We use a series of nested loops to fill up these tables implemented as flat dictionaries with keys (i,j,lefthand side)

The log probability of the nonterminal S for the 5-token sentence “Terms were n’t disclosed .” is: -23.5605111717

3.3

BuildTree is returned from the Parse method.

We recursively build the tree by following the spans of the backpointers and calling BuildTree from within itself, and merging the left and right sides as children of the initial node. When we reach the end, we use the token at j of the current span as a child.

The result of calling this on the test sentence is as follows:

(S

(NP+NNS Terms)

(S|<VP-.>

(VP (VBD were) (VP|<RB-VP+VBN> (RB n't) (VP+VBN disclosed)))

(. .)))

3.4

When we sort the sentences by length we get the following results:

bucket1 has 23 sentences.

bucket2 has 134 sentences.

bucket3 has 187 sentences.

bucket4 has 86 sentences.

bucket5 has 28 sentences.

3.5 Below are the results for each bucket by using evalb on the files generated for each one.

Bucket1:

Bracketing FMeasure = 80.95

Average crossing = 0.27

Bucket2:

Bracketing FMeasure = 80.61

Average crossing = 1.05

Bucket1 and Bucket2 combined:

Bracketing FMeasure = 80.64

Average crossing = 0.94

The results for the other buckets could not be obtained within the time constraints.