HW #4

Goals

Through this assignment you will:

- Explore issues in probabilistic parser design for natural language processing.
- Learn how to extract rule probabilities and explore parser evaluation.
- Improve your understanding of the probabilistic CKY algorithm through implementation.
- Investigate the tradeoffs in probabilistic parser design in terms of speed and accuracy.

NOTE: *You may work in teams of two (2) on this assignment. If you do so:*

- Please include a brief discussion of each teammate's contribution in the readme.

Background

Please review the class slides and readings in the textbook on the probabilistic Cocke-Kasami-Younger algorithm, optimization, and evaluation.

1. Inducing a Probabilistic Context-free Grammar

Based on the material in the lectures and text, implement a procedure that takes a set of context-free grammar parses of sentences (a small treebank) and induces a probabilistic context-free grammar from them.

Your algorithm must create a grammar of the form:

A -> B C [0.38725]

All productions must have an associated probability.

Specifically, the program should:

- Read in a set of parsed sentences (a mini-treebank) from a file
- Identify productions and estimate their probabilities
- Print out the induced PCFG with production of the form above.

1.1 Programming

Create a program named hw4_topcfg.sh to perform PCFG induction invoked as:

hw4_topcfg.sh <treebank_filename> <output_PCFG_file>

where:

- <treebank_filename> is the name of the file holding the parsed sentences, one parse per line, in Chomsky Normal Form.
- <output_PCFG_file> is the name of the file where the induced grammar should be written.

2. Converting from CKY to Probabilistic CKY

Implement a probabilistic version of the CKY parsing algorithm. Given a probabilistic context-free grammar and an input string, the algorithm should return the highest probability parse tree for that input string.

You should follow the approach outlined in the textbook and course notes. You may adapt the CKY implementation that you created for HW#3. You may use any language that you like, in keeping with the course policies.

Specifically, your program should:

- Read in a PCFG in NLTK format as generated above
- Read in a set of sentences to parse
- For each sentence:
  - Parse the sentences using a PCKY algorithm that you implement
  - Print the highest scoring parse to a file, **on a single line**

2.1 Programming

Create a program named hw4_parser.sh to perform PCKY parsing, invoked as:

hw4_parser.sh <input_PCFG_file> <test_sentence_filename> <output_parse_filename>, where:

- <input_PCFG_file> is the name of the file holding the induced PCFG grammar to be read.
- <test_sentence_filename> is the name of the file holding the test sentences to be parsed.
- <output_parse_filename> is the name of the file to which the best parse for each sentence will be written. Each parse should appear on only one line, and there should be one line per sentence in the output file. In other words, output a blank line if the sentence does not parse.

Note: *The test sentences may include words not seen in training; this happens in real life. In a baseline system, these may fail to parse.*

3. Evaluating the PCKY Parser

Use the **evalb** program to evaluate your parser.

The executable may be found in /dropbox/20-21/571W/hw4/tools/ along with the required parameter file. It should be run as:

$dir/evalb -p $dir/COLLINS.prm <gold_standard_parse_file> <hypothesis_parse_file>

where

- $dir is the directory where the program resides
- <gold_standard_parse_file> is the name of the file containing the gold standard parses for the sentences to be evaluated against. The file has one parse per line.
- <hypothesis_parse_file> is the name of the file containing the parses output by your system to be evaluated against the gold standard parses. The file has one parse per line.

4: Improving the Parser

You will also need to improve your baseline parser. You can improve the parser either by:

- Improving the coverage of the parser in terms of sentences parsed,
- Improving the accuracy of the parser as measured by evalb, or
- Improving the efficiency of the parser as measured by running time, with little or no degradation in accuracy.

You will either:

- Modify your grammar induction process
- Modify your parsing process
- Or both, depending on your approach

Create a second script, either:

1. **hw4_improved_parser.sh** — if you are modifying the parsing algorithm.
2. **hw4_improved_induction.sh** — if you are modifying the induction algorithm.

hw4_improved_parser.sh <input_PCFG_file> <test_sentence_filename> <output_parse_filename>

hw4_improved_induction.sh <treebank_filename> <output_PCFG_file>

Re-run the evaluation script on your new parses to demonstrate your improvement (re-parsing using your new PCFG file if necessary).

5. Combining it All

Finally, write a script hw4_run.sh that will call all the components of the system:

1. Grammar Induction
2. PCKY Parsing
3. Evaluation of Baseline System
4. Improved PCKY Parsing
5. Evaluation of Improved System

Calling specification:

```
hw4_run.sh <treebank_filename> <output_PCFG_file> \
           <test_sentence_filename> <baseline_parse_output_filename> \
           <input_PCFG_file> \
           <improved_parse_output_filename> \
                   <baseline_eval> <improved_eval>
```

Where

- **treebank_filename**: Input parsed sentences.
- **output_PCFG_file**: Output PCFG for the ***unmodified*** grammar induction
- **test_sentence_filename**: input test sentences
- **baseline_parse_output_filename**: output parses from your baseline PCFG parser
- **input_PCFG_file**:

1. If you have not modified the induction process:
   - You should take this argument in your script, but may ignore it and re-use the original induced PCFG output.
2. If you have modified the induction process:
   - This argument should specify the output PCFG of the modified induction process.

- **improved_parse_output_filename**: output parses from your improved PCFG parser
- **baseline_eval**: evalb output for your baseline parses
- **improved_eval**: evalb output for your improved parses.

6. Files

Training, Test, Evaluation, Example Data

You will use the following files, derived from the Air Travel Information System (ATIS) subset of the Penn Treebank as described in class. All files can be found on patas in /dropbox/20-21/571W/hw4/data/, unless otherwise mentioned:

- **parses.train**: parses of 514 sentences from the Air Travel Information System domain. These parses will form your (relatively small) training treebank for this assignment. They will form the basis for inducing your PCFG rules.
- **sentences.txt**: over 50 sentences from the Air Travel Information System domain. These are the test sentences your system must parse and will be evaluated on, both for the baseline and for your improved system.
- **parses.gold**: parses of the 50+ test sentences from the Air Travel Information System domain. These parses will provide the gold standard and will be used to be evaluate the output of your system on the test sentences above.
- **toy.pcfg**: This file contains a simple Probabilistic Context-Free Grammar (in Chomsky Normal Form) that parses a simple toy set of sentences.
- **toy_sentences.txt**: This file contains a small set of toy example sentences.
- **toy_output.txt**: Example output from parsing with the pcfg in the one parse per line format required by evalb.

Submission Files

- **hw.tar.gz**: Tarball containing the following:
  - Output Files
    - You should generate output files, named as specified below, corresponding to each of the main components of this assignment:
      - **hw4_trained.pcfg**: This file should contain the probabilistic context-free grammar trained on the **parses.train** treebank.
      - [**hw4_improved.pcfg**]: If you are improving your induction algorithm, include your improved PCFG.
      - **parses_base.out**: This file should contain the results of parsing the sentences in **sentences.txt** using the PCFG induced in part 1 and your PCKY implementation. Your output file **MUST** have the same number of lines as the input strings to parse. If your algorithm fails to parse a sentence you should output a blank line in the output file corresponding to that input.
      - **parses_base.eval**: This file should contain the results of running evalb on your baseline parser output.
      - **parses_improved.out**: This file should contain the output of your improved parser, again run on the **sentences.txt** test data. It should have the same format as **parses_base.out**.

        - **parses_improved.eval**: This file should contain the results of running evalb on your improved parser output.
    - Program Files:
        - **hw4_topcfg.sh**: Code to induce the PCFG.
        - **hw4_parser.sh**: Code implementing PCKY.
        - **{ hw4_improved_parser.sh | hw4_improved_induction.sh}**: Code invoking your parsing improvements.
        - **hw4_run.sh**: Run all steps end-to-end:
- **readme.{txt|pdf}**: Write-up file.
    - This file should describe and discuss your work on this assignment. Include problems you came across and how (or if) you were able to solve them, any insights, special features, and what you learned. Give examples if possible. If you were not able to complete parts of the project, discuss what you tried and/or what did not work. Make sure to discuss the improvements you implemented and compare your 'improved' results to your baseline results. This will allow you to receive maximum credit for partial work.

6. Handing In your Work

- If working in groups:
    - one group member should submit the tarball and readme
    - the other group member should submit a .txt file naming the other groupmate.