LING570 Hw10: MaxEnt POS tagger
Due: 12/6

The example files are under /dropbox/18-19/570/hw10/examples/.

Q1 (55 points): Create a MaxEnt POS tagger, **maxent_tagger.sh**.

- The command line is: maxent_tagger.sh train_file test_file rare_thres feat_thres output_dir
- The train_file and test_file have the format (e.g., **test.word_pos**):
  w_1/t_1 w_2/t_2 ... w_n/t_n
- rare_thres is an integer: any words (in the train_file and test_file) that appear LESS THAN raw_thres times in the **train_file** are treated as *rare words*, and features such as pref=xx and suf=xx should be used for rare words (see Table 1 in (Ratnaparkhi, 1996)).
- feat_thres is an integer: All the w_i features (i.e., CurrentWord=xx features), regardless of their frequency, should be kept. For all OTHER types of features, if a feature appears LESS THAN feat_thres in the **train_file**, that feature should be removed from the feature vectors.
- output_dir is a directory that stores the output files from the tagger. Your script should create the following files and store them under output_dir:
  - train_voc (e.g., **ex_train_voc**): the vocabulary that includes all the words appearing in train_file. The file has the format "word freq" where freq is the frequency of the word in the training data. The lines should be sorted by freq in descending order. For words with the same frequency, sort the lines alphabetically.
  - init_feats (e.g., **ex_init_feats**): features that occur in the train_file. It has the format "featName freq" and the lines are sorted by the frequency of the feature in the train_file in descending order. For features with the same frequency, sort the lines alphabetically.
  - kept_feats (e.g., **ex_kept_feats**): This is a subset of init_feats, and it includes the features that are kept after applying feat_thres.
  - final_train.vectors.txt (e.g., **ex_final_train.vectors.txt**): the feat vectors for the train_file in the Mallet text format. Only features in kept_feats should be kept in this file.
  - final_test.vectors.txt: the feat vectors for the test_file in the Mallet text format. The format is the same as final_train.vectors.txt.
  - final_train.vectors: the binary format of the vectors in final_train.vectors.txt.
  - me_model: the MaxEnt model (in binary format) which is produced by the MaxEnt trainer.
  - me_model.stdout and me_model.stderr: the stdout (standard out) and stderr (standard error) produced by the MaxEnt trainer are redirected and saved to those files by running command such as "mallet train-classifier --trainer MaxEnt --input final_train.vectors --output-classifier me_model > me_model.stdout 2 > me_model.stderr". The training accuracy is displayed at the end of me_model.stdout.
  - sys_out: the system output file when running the MaxEnt classifier with command such as "mallet classify-file --input final_test.vectors.txt --classifier me_model --output sys_out".

Your script **maxent_tagger.sh** should do the following:

1. Create feature vectors for the training data and the test data. The vector files should be called **final_train.vectors.txt** and **final_test.vectors.txt**.
2. Run **mallet import-file** to convert the training vectors into binary format, and the binary file is called **final_train.vectors**.
3. Run **mallet train-classifier** to create a MaxEnt model **me_model** using **final_train.vectors**
4. Run **mallet classify-file** to get the result on the test data **final_test.vectors.txt**.
5. Calculate the test accuracy

For step 2-4, you should use Mallet commands. For Step 5, if you don't want to write code for it, you can use the vectors2classify command, which covers step 3-5. In that case, you need to convert **final_test.vectors.txt** to the binary format first.

For the first step, you need to write some code. Features are defined in Table 1 in (Ratnaparkhi, 1996). The following is one way for implementing this step:

1. create train_voc from the train_file, and use the word frequency in train_voc and rare_thres to determine whether a word should be treated as a *rare word*. The feature vectors for rare words and non-rare words are different.
2. Form feature vectors for the words in train_file, and store the features and frequencies in the **training data** in init_feats.
3. Create kept_feats by using feat_thres to filter out low frequency features in init_feats. Note that w_i features are NOT subject to filtering with feat_thres and every w_i feature in init_feats should be kept in kept_feats.
4. Go through the feature vector file for train_file and remove all the features that are not in kept_feats.
5. Create feature vectors for test_file, and use only the features in kept_feats. If a word in the test_file appears LESS THAN rare_thres times (or does not appear at all) in the training_file, the word should be treated as a *rare* word even if it appears many times in the test_file.
6. For the feature vector files, replace all the occurrences of "**,**" with "**comma**" as Mallet treats "**,**" as a separator.

Q2 (20 points): Run maxent_tagger.sh with wsj_sec0.word_pos as train_file, test.word_pos as test_file, and the thresholds as specified in Table 1:

- *training accuracy* is the accuracy of the tagger on the train_file
- *test accuracy* is the accuracy of the tagger on the test_file
- *# of feats* is the number of features in the train_file before applying feat_thres
- *# of kept feats* is the number of features in the train_file after applying feat_thres
- *running time* is the CPU time (in minutes) of running maxent_tagger.sh.

Please do the following:

Table 1: Tagging accuracy with different thresholds

Expt id	rare thres	feat thres	training accuracy	test accuracy	# of feats	# of kept feats	running time
1_1	1	1					
1_3	1	3					
2_3	2	3					
3_5	3	5					
5_10	5	10					

- Fill out Table 1.
- What conclusion can you draw from Table 1?
- Save the output files of maxent_tagger.sh to res_*id*/, where *id* is the experiment id in the first column (e.g., the files for the first experiment will be stored under res_1_1). Submit only the subdirs for the first row and the last row (i.e., res_1_1 and res_5_10).

Submission: Your submission should include the following:

1. readme.[txt|pdf] includes Table 1 and your answer to Q2.
2. hw.tar.gz that includes maxent_tagger.sh and res_1_1/ and res_5_10/ created in Q2 (see the complete file list in submit-file-list).