LING570 Hw9: MaxEnt POS tagger
Due: 12/02/2009

The example files are under dropbox/09-10/570/hw9/examples/.

Q1 (70 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 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 those words.
- feat_thres is an integer: All the w_i features (CurrentWord=xx features) should be kept. For all OTHER 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. It should include the following:
  - train_voc (e.g., **ex_train_voc**): the vocabulary collected from train_file. The file has the format "word freq" and the lines are sorted by the frequency of the word in the train_file.
  - train.vectors.feats (e.g., **ex_train.vectors.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.
  - kept_feats (e.g., **ex_kept_feats**): This is a subset of train.vectors.feats, and it includes only the features that occur at least feat_thres times in the train_file.
  - final_train.vectors.txt (e.g., **ex_final_train.vectors.txt**): the feat vectors for the train_file in the Mallet text format.
  - 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
  - me_model: the MaxEnt model (in binary format) which is produced by the MaxEnt trainer.

Your tagger **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 **info2vectors** to convert the vectors into binary format. The binary files are called **final_train.vectors** and **final_test.vectors**
3. Run **vectors2train** to create a MaxEnt model **me_model** using **final_train.vectors**
4. Run **classify** to get the result on the test data **final_test.vectors**.

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_1	2	1					
2_3	2	3					
3_3	3	3					
3_5	3	5					
5_10	5	10					

Among the four steps, only Step 1 requires much coding whereas you should use Mallet for the other three steps. For the first step, you should use the features defined in Table 1 in (Ratnaparkhi, 1996). Remember to do the following:

- Use rare_thres to identify *rare words* in the training data and in the test data, and represent them differently from non-rare words in the feature vectors.
- Remove low-frequency features from the feature vectors.
- Replace "**,**" with "**comma**" as Mallet treats "**,**" as a separator.

Q2 (30 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. If you cannot get CPU time, you can use clock time.

Please do the following:

- Fill out the table.
- 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.
- What conclusion can you draw from the experiments?

Submission: Submit a tar file via CollectIt. The tar file should include the following.

1. In your note file hw9.*, include Table 1 and your answers to Q2.
2. The code for Q1.

3. The res_*/ created by Q2.

4. You should gzip your tar file. If the compressed file is still too big for CollectIt, you can just tar the files for (1) and (2) and in your note file specify the location of (3) on patas. Make sure that directory can be accessed by David.