LING570 Hw10: Word clustering
Due: 12/09/2009

The example files are under dropbox/09-10/570/hw10/examples/. **The homework is explained in class on 12/02/09.**

Q1 (15 points): Write a script, **create_vector.sh**, that creates feature vectors for words in word_list using the features in feat_list and the occurrence information in train_file.

- The command line is: create_vector.sh train_file output_file word_list feat_list
- In the train_file (e.g., **sec0-19.word**), each line is a sentence without POS tags.
- The word_list and feat_file (e.g., **word.100**) has the format: word freq
  The frequency information is NOT used in the homework.
- The output_file has the format (e.g., **ex/vectors**):
  "w featname1 featval1 featname2 featval2 ..."
  - w is a word that appears in word_list
  - featname has the format: "featidx_(L | R)=x":
    - L and R indicates whether the feature x appears to the left of w or not.
    - featidx is the index of the feature x according to feat_file; that is, suppose x appears on the i-th line of feat_file, it will have index $i-1$ for L features and index $N+i-1$ for R features, where N is the number of lines in feat_file.
  - featval is the number of occurrences of "x w" for the L features or "w x" for the R features in the training data.
  - For instance, suppose in the train_file, you see "new york" 919 times, and "york" appears in word_list and the word "new" appears on the 38th line of feat_list, your output_file should include something like "york ... 37_L=new 919 ..."
  - The lines in the file should have the same order of the lines in word_file.
  - The (featname, featval) pairs on each line should be sorted by featidx.
  - Note: the order of lines in **ex/vectors** is not the same as required in Q5. This is because I created this file with a word list different from word.100. So please do NOT treat ex/vectors as the gold standard for your experiments.

Q2 (30 points): Write a script, **k-medoids.sh**, that implements the k-medoids algorithm.

- The command line is: k-medoids.sh vector_file cluster_size sys_cluster
- vector_file is the output_file created in Q1.
- cluster_size is an integer indicating the number of clusters that you want the script to create.

- sys_cluster is the clusters produced by the algorithm. It has the format (e.g., **ex/sys_cluster**):
  w word1 word2 ...
  word1, word2, and so on form a cluster; w is the medoid of the cluster and it serves as the name of the cluster.
- For clustering, please use the cosine similarity function to calculate the similarity of two words.
- Choose the initial medoids in the following way: Let N be the number of lines in vector_file and C be the cluster_size. The i-th initial medoid is the word on the x-th line of vector_file, where $x = (i-1) * \lfloor N/C \rfloor$. For instance, if N=100 and C=34, $\lfloor N/C \rfloor$ would be 2, and the initial medoids would be the words on the 0-th, 2rd, 4th, ..., 66-th lines in vector_file.

Q3 (15 points): Write a script, **calc_acc.sh**, that maps sys clusters to the clusters in the gold standard, and then calculates the accuracy.

- The command line is: calc_acc.sh gold_cluster sys_cluster flag > map_file 2>acc_file
- sys_cluster (e.g., **ex/sys_cluster**) is the cluster file produced by Q2.
- gold_cluster (e.g., **gold.100**) has the same format as sys_cluster except that the cluster name here is a POS tag that serves as the name of a cluster.
- flag is 0 if the mapping from sys clusters to gold clusters needs to be one-to-one, and it is 1 if the mapping can be many-to-one.
- map_file (e.g., **ex/res.1_to_1.map**) shows the mapping from sys to gold clusters. It has the format "sys_cluster_name ⇒ gold_cluster_name cnt".
  The cluster name is the first word of the line in a cluster file; cnt is the number of words in the sys_cluster that appear in the gold_cluster.
- acc_file should end with a line of the format "Acc=xx", where xx is the tagging accuracy after mapping. You can decide what other lines should look like. An example is **ex/res.1_to_1.acc**.

Q4 (5 points): Write a wrapper, **wrapper.sh**, that does everything by calling the codes in Q1-Q3.

- The command line is: wrapper.sh train_file word_list feat_file cluster_size gold_cluster output_dir
- The wrapper should call the commands in Q1-Q3 and produce the following files under output_dir:
  - vectors: the vector_file produced by Q1
  - sys_cluster: the system output produced by Q2
  - The mapping and accuracy files created in Q3.
  - Please see files under **ex/** for naming convention.

Q5 (35 points): Run wrapper.sh with **sec0-19.word** as train_file and other files specified in the table. Fill out the table.

Table 1: Tagging accuracy with the word clustering approach

word list	feat list	cluster size	gold cluster	output dir	1-to-1 Acc	many-to-1 Acc	num of iterations	running time
word.100	word.100	34	gold.100	100-100-34				
word.100	word.500	34	gold.100	100-500-34				
word.500	word.100	36	gold.500	500-100-36				
word.500	word.500	36	gold.500	500-500-36				
word.1000	word.100	39	gold.1000	1K-100-39				
word.1000	word.500	39	gold.1000	1K-500-39				
word.5000	word.100	41	gold.5000	5K-100-41				
word.5000	word.500	41	gold.5000	5K-500-41				

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

1. Your note file hw10.* should include Table 1.
2. The scripts and codes for Q1-Q4.
3. The output dirs created by Q5. Please name your dir according to the "output dir" column in Table 1.
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.