Author: Ryan Timbrook
UW Net ID: timbrr
Project: Ling 570 HW 10
Date: Dec 6, 2018

##---- Q1 ----##

Q1: Write maxent_tagger.sh
- Create a MaxEnt POS tagger

Format: command line: maxent_tagger.sh train_file test_file rare_thres feat_thres output_dir

Input: (e.g., test.word_pos)
-train_file: Format: w1/t1 w2/t2 ... wn/tn
-test_file: Format: w1/t1 w2/t2 ... wn/tn
-rare_thres: type=Integer
-> any words in training and test that appear less than this value are treated as rear words
-> features such as pref=xx and suf=xx should be used for rare_words
-feat_thres: type=Integer
-> all CurrentWord=xx features, regardless of their frequency, should be kept. For all other types of features, if a feature appears less than this value in the train_file, that feature should be removed from the feature vectors

Output File: (output_dir is a directory that stores the output files from the tagger)
Create and store the following files under this directory
-train_voc
-init_feats
-kept_feats
-final_train.vectors.txt
-final_test.vectors.txt

From Command line, Run as: ./maxent_tagger.sh wsj_sec0.word_pos test.word_pos 1 1 res_1_1

##---- Q2 ----##

*Run below 5 command lines on maxent_tagger to complete table 1:

#Creates feature vectors for train_file and test_file
Command Format: shell_script train_file test_file rare_thres feat_thres output_dir
CMD 1: $ ./maxent_tagger.sh wsj_sec0.word_pos test.word_pos 1 1 res_1_1
CMD 2: $ ./maxent_tagger.sh wsj_sec0.word_pos test.word_pos 1 3 res_1_3
CMD 3: $ ./maxent_tagger.sh wsj_sec0.word_pos test.word_pos 2 3 res_2_3
CMD 4: $ ./maxent_tagger.sh wsj_sec0.word_pos test.word_pos 3 5 res_3_5
CMD 5: $ ./maxent_tagger.sh wsj_sec0.word_pos test.word_pos 5 10 res_5_10

maxent_tagger.sh performs the following steps:

1. Executes maxent_tagger.py which creates final_train.vectors.txt and final_test.vectors.txt along with init_feats, kept_feats and train_voc output files
2. Executes mallet import-file commands to convert final_train.vectors.txt and final_test.vectors.txt into binary vector files, final_train.vectors and final_test.vectors
3. Executes vector2classify on final_train.vectors and final_test.vectors outputs are the me_model, me_model.stderr and me_model.stdout files

Mallet Commands used in maxent_tagger.sh

##convert the training and test vectors from the text format to the binary format.##

$ mallet import-file --token-regex "[^\s]+" --preserve-case --input final_train.vectors.txt --output final_train.vectors

$ mallet import-file --token-regex "[^\s]+" --preserve-case --input final_test.vectors.txt --output final_test.vectors --use-pipe-from final_train.vectors

MaxEnt Classifier Command used in maxent_tagger.sh

##training (with MaxEnt trainer) and for testing##

$ vectors2classify --training-file res_1_1/final_train.vectors --testing-file res_1_1/final_test.vectors --trainer MaxEnt --output-classifier me_model --report train:accuracy train:confusion test:raw test:accuracy test:confusion > res_1_1/me_model.stdout 2> res_1_1/me_model.stderr

#Get sys_out from classifier

$ mallet classify-file --input res_1_1/final_test.vectors.txt **--classifier res_1_1/"me_model"** --output res_1_1/"**sys_out**"

Table 1: Tagging accuracy with different thresholds

Expt_Id	rare_thres	feat_thres	training_accuracy	test_accuracy	#_of_feats	#_of_kept_feats	running_time(min)
1_1	1	1	0.9575036059503560	0.8280930992241730	325157	325157	3.9
1_3	1	3	0.9700759940582550	0.8366680277664350	325157	298784	5.98
2_3	2	3	0.9586661212891000	0.8293180890159240	356312	321519	2.78
3_5	3	5	0.9420464575574260	0.8248264597795010	373390	319095	3.48
5_10	5	10	0.9737572926309440	0.8558595345038790	398230	313552	4.28

Table 1 Conclusions:

- It's observed that the **test_accuracy** is positively correlated with **rare_thres** and **feat_thres** values. This is shown in the Correlation Table 1 below. Values of approximately plus or minus .7 represent statistical correlations.
- It's observed that the **#_of_feats** shows a positive correlation with **rare_thres** and **feat_thres** values. This is shown in the Correlation Table 1 below. As well, in Feats Plot 1, this correlation is visualized by the blue positive linear slope where the y-axis is the number of initial features and the x-axis is the rare_thres, feat_thres pairing.
- Overall, the best scores were observed where rare_thre=5 and feat_thres=10, yielding a ~3% increase in test_accuracy

Correlation Table 1

	rare_thres	*feat_thres*	*training_accuracy*	*test_accuracy*	*#_of_feats*	*#_of_kept_feats*	*running_time(min)*
rare_thres	1						
feat_thres	0.965545988	1					
training_accuracy	0.178991294	0.365370347	1				
test_accuracy	0.692105586	0.81638443	0.823642419	1			
#_of_feats	0.976952799	0.90937179	0.022433946	0.548678857	1		

#_of_kept_feats	0.066155021	-0.187755385	-0.568645678	-0.400524736	0.153575786	1	
running_time(min)	-0.215659957	0.032342637	0.584325952	0.369891982	-0.366658382	-0.888275844	1

Regression Analysis Table 1

Regression analysis on: Input x=#_of_feats Output y=test_accuracy

Regression Statistics	
Multiple R	0.548678857
R Square	0.301048488
Adjusted R Square	0.068064651
Standard Error	30481.21075
Observations	5

	Coefficients	*Standard Error*	*t Stat*	*P-value*
Intercept	-804918.1138	1021065.76	-0.788311728	0.488065207
test_accuracy	1389979.143	1222793.032	1.136724782	0.338235717

P-Value is greater than .05 alpha value, statistically insignificant

Regression Analysis Table 2

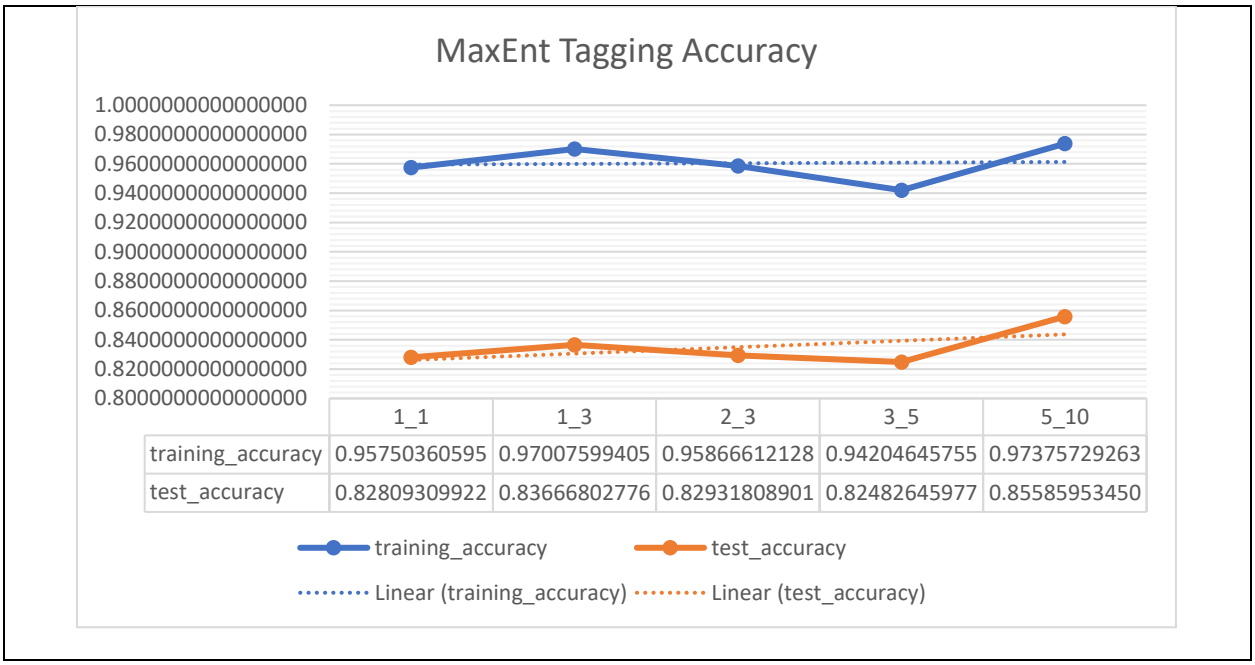
Regression analysis on: Input x=feat_thres Output y=test_accuracy

Regression Statistics	
Multiple R	0.816384431
R Square	0.666483537
Adjusted R Square	0.555311383
Standard Error	0.008311459
Observations	5

	Coefficients	*Standard Error*	*t Stat*	*P-value*
Intercept	0.821919704	0.006492359	126.5980099	1.08666E-06
feat_thres	0.002962122	0.00120978	2.4484808	0.091803567

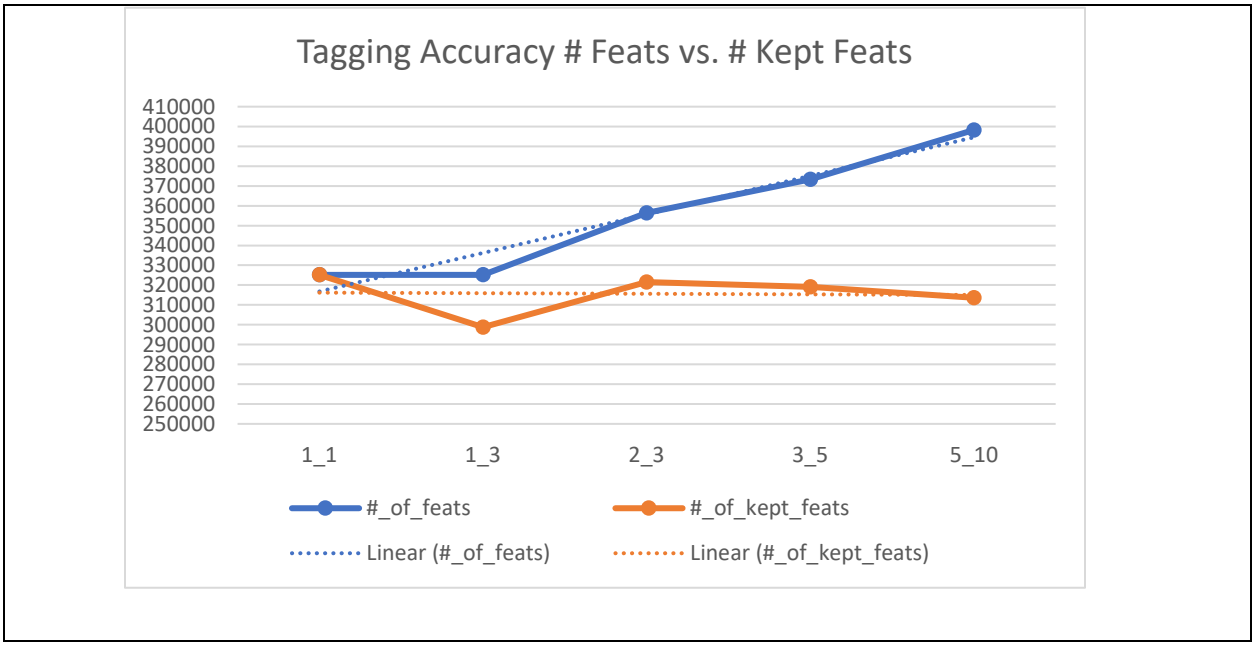
Correlation, R, shows a strong positive correlation, the P-Value is greater than the standard alpha of .05 (95%), however it's below .10, putting it into a 90% range. Is statistically significant to look further into.

MaxEnt Tagging Accuracy Plot 1



	1_1	1_3	2_3	3_5	5_10
training_accuracy	0.95750360595	0.97007599405	0.95866612128	0.94204645755	0.97375729263
test_accuracy	0.82809309922	0.83666802776	0.82931808901	0.82482645977	0.85585953450

Feats Plot 1



THE BELOW INFORMATION IS NOT FOR GRADING

****

***** FOR REFERENCE ONLY! This table represents Tagging accuracy output values prior to including an additional command to mallet that preserves case sensitivity

mallet import-file --token-regex "[^\s]+" --preserve-case

From the mallet documentation:

--preserve-case. MALLET by default converts all word features to lowercase.

--token-regex. MALLET divides documents into tokens using a regular expression. As of version 2.0.8, the default token expression is '\p{L}[\p{L}\p{P}]+\p{L}', which is valid for all Unicode letters,
and supports typical English non-letter patterns such as hyphens, apostrophes, and acronyms. Note that this expression also implicitly drops one- and two-letter words.

Table 1.a: Tagging accuracy with different thresholds

Expt Id	rare thres	feat thres	training accuracy	test accuracy	# of feats	# of kept feats	running time(min)
1_1	1	1	0.7476695873070548	0.5349122090649244	325157	325157	232 sec
1_3	1	3	0.7115239714968461	0.5700285830951409	325157	298784	204 sec
2_3	2	3	0.7513293578179157	0.5912617394855043	356312	321519	256 sec
3_5	3	5	0.7643107791005576	0.6194365046957943	373390	319095	178 sec
5_10	5	10	0.7781963789800004	0.6382196815026542	398230	313552	91 sec

***Further Data Investigation

$ vectors2info --input res_1_1/final_train.vectors --print-matrix sic > info_res_1_1/final_train.vectors_info.txt

$ vectors2info --input res_1_1/final_train.vectors --print-labels TRUE > info_res_1_1/final_train.vectors_info.txt

$ classifier2info --classifier res_1_1/me_model > info_res_1_1/me_model_info.txt

$ vectors2classify --training-file res_1_1/final_train.vectors --testing-file res_1_1/final_test.vectors --trainer MaxEnt --output-classifier me_model --report train:accuracy train:confusion test:raw test:accuracy test:confusion >me_model.stdout 2>me_model.stderr