**5.2 (Question 2, 3, 4)**

Base on our twitter corpus regarding Trump, we sort all the nouns according to the number of appearances they have. We choose 5 most regular words as our key words: Trump, politics, president, candidate and opinion.

Word2vec: Word2vec is a toolkit provided by Google, providing a method to build vectors model based on continuous bag-of-words (CBOW) and skip-gram architectures. It can translate a word into its vector representation, as well as training a model from a large corpus to support various analysis tasks.

Base on vector representation of words, we can measure their relations with each other. We perform 3 tasks based on 3 questions of section 5.2 of the assignment.

Question 2) Given vector representation of words in our corpus, we simply do a normalized dot product (cosine distance) to compute their “distance”. The distance is 1 if they are equal (in direction), smaller “distance” means the words are more different. (*Code: /5.2/NLP 5.2/word-analogy.c*)

Outcome for our test:

|  |
| --- |
| Word: politics Position in vocabulary: 1764  Word Distance  ------------------------------------------------------------------------  soaring. 0.859098  4.21 0.855930  times 0.851998  #Trump. 0.845365  knows 0.843575  good 0.836803  Current-American 0.834254  stands 0.830490  liberals 0.825006  its 0.824097  Enter word (EXIT to break): Trump  Word: Trump Position in vocabulary: 38  Word Distance  ------------------------------------------------------------------------  #Stumped 0.647869  http://t.co/O5v9oX3pydCarson 0.641029  Donald 0.639887  @LindaSuhler: 0.601711  Blasts 0.562340  vote 0.542295  Michigan! 0.536790  @WashTimes: 0.530438  #Trump2016Carson 0.529623  Compares 0.506741  Enter word (EXIT to break): president  Word: president Position in vocabulary: 847  Word Distance  ------------------------------------------------------------------------  taking 0.874387  chance 0.872044  worse 0.865401  Josh 0.856510  gave 0.852727  class 0.849465  likes 0.843745  Muslims 0.836635  defense 0.835478  doesn't 0.833406  Enter word (EXIT to break): candidate  Word: candidate Position in vocabulary: 578  Word Distance  ------------------------------------------------------------------------  political 0.819578  somebody 0.811906  told 0.809150  dinner 0.808545  America. 0.799716  few 0.797652  liberal 0.797198  hard!- 0.791712  donate 0.789009  threatening 0.787662  Enter word (EXIT to break): opinion  Word: opinion Position in vocabulary: 481  Word Distance  ------------------------------------------------------------------------  American 0.864201  Islamophobe 0.853458  #LiberalDelusion 0.845501  Voter? 0.843300  an 0.813199  correcting 0.799435  donate 0.793326  racist, 0.698671  tune 0.657279  tunnel, 0.647378 |

Question 3) We change the corpus to Google News dataset to search for closest words to each of the five chosen keywords.

This dataset is much bigger than ours (3.4 GB in size compare to 6.4 MB), thus the result is also better: words are more natural and correct.

Outcome for our test: (*Code: /NLP/NLP 5.2/word-analogy.c*)

|  |
| --- |
| TMT:word2vec-mac-master MinhTri$ ./word-analogy GoogleNews-vectors-negative300.bin  Enter word (EXIT to break): politics  Word: politics Position in vocabulary: 2029  Word Distance  ------------------------------------------------------------------------  partisan\_politics 0.683224  Politics 0.674026  political 0.671894  politcs 0.622195  poltics 0.594164  Lisa\_Vorderbrueggen\_covers 0.586606  partisanship 0.573556  politicians 0.570558  politician 0.569530  politicking 0.568017  Enter word (EXIT to break): Trump  Word: Trump Position in vocabulary: 13034  Word Distance  ------------------------------------------------------------------------  Donald\_Trump 0.810392  impersonator\_entertained 0.594226  Ivanka\_Trump 0.592458  Ivanka 0.560721  mogul\_Donald\_Trump 0.559245  Trump\_Tower 0.548555  Kepcher 0.546859  billionaire\_Donald\_Trump 0.544727  Trumpster 0.541282  tycoon\_Donald\_Trump 0.538397  Enter word (EXIT to break): president  Word: president Position in vocabulary: 348  Word Distance  ------------------------------------------------------------------------  President 0.800628  chairman 0.670875  vice\_president 0.670023  chief\_executive 0.669128  CEO 0.659013  pesident 0.626521  Vice\_President 0.621666  executive 0.618248  prez 0.576191  Presdient 0.571838  Enter word (EXIT to break): candidate  Word: candidate Position in vocabulary: 1620  Word Distance  ------------------------------------------------------------------------  candidates 0.794275  candiate 0.705062  Candidate 0.677797  challenger 0.628802  canidate 0.623805  candidacy 0.618346  candi\_date 0.616838  nominee 0.590141  mayoral\_candidate 0.589086  cadidate 0.587563  Enter word (EXIT to break): opinion  Word: opinion Position in vocabulary: 1966  Word Distance  ------------------------------------------------------------------------  opinions 0.716355  opinon 0.633364  opnion 0.561680  Opinions 0.549686  opinons 0.549233  Opinion 0.541371  views 0.524808  viewpoint 0.524092  opionion 0.487257  veiws 0.469507 |

Question 4) We switch back to our corpus to find three linguistic regularities. Again the distance is normalized dot product.

Outcome for our test: (Code: *NLP/NLP 5.2/regularities.c*)

|  |
| --- |
| TMT:word2vec-mac-master MinhTri$ ./regularities corpus.bin  Enter three words (EXIT to break): president politician bad  Word: president Position in vocabulary: 847  Word: politician Position in vocabulary: 0  Out of dictionary word!  Enter three words (EXIT to break): president man bad  Word: president Position in vocabulary: 847  Word: man Position in vocabulary: 173  Word: bad Position in vocabulary: 897  Word Distance  ------------------------------------------------------------------------  media 0.622181  Enter three words (EXIT to break): candidate trump nice  Word: candidate Position in vocabulary: 578  Word: trump Position in vocabulary: 542  Word: nice Position in vocabulary: 2321  Word Distance  ------------------------------------------------------------------------  allday 0.689042  Enter three words (EXIT to break): election vote rich  Word: election Position in vocabulary: 1324  Word: vote Position in vocabulary: 175  Word: rich Position in vocabulary: 3511  Word Distance  ------------------------------------------------------------------------  poll! 0.594140 |

**5.4 (Question 6)**

From the dataset created in section 5.3, we are able to establish the training and testing set for our model with 3-fold validation technique. We alternatively select 2 sets as the training set and the other as the testing set. We first apply POS tagging on all three sets with TextBlob (<https://github.com/sloria/textblob-aptagger/tree/master>), then use Weka (<http://www.cs.waikato.ac.nz/ml/weka>) to filter our strings with its StringToWordVector and subsequently build an SVM Classification model.

1. S1 + S2 as training set and S3 as testing set:

* Precision 0.78
* Recall 0.67
* F-Measure 0.83

1. S2 + S3 as training set and S1 as testing set:

* Precision 0.73
* Recall 0.72
* F-Measure 0.79

1. S1 + S3 as training set and S2 as testing set:

* Precision 0.69
* Recall 0.76
* F-Measure 0.81

1. Average:

* Precision: 0.73
* Recall: 0.72
* F-Measure: 0.91

Due to human errors, S1 may contain some annotation errors. We can compare the annotations by the SVM model trained with S2 + S3 when testing it on S1 with the annotations made by human to mark down the differences between them. Then we manually check again to fix possible errors according to these differences.

**5.5 (Question 7)**

In this section we repeat the process in 5.4 with different sets. Each set is used to create a classification model with Weka and LibSVM, then has its performance tested with 3-fold cross validation. The first set is the 300-opnionated-document set in section 5.4, the second set is the same as the first set except all strings are filter with Google’s deep learning library Word2vec (*Code: /5.4\_5.5/NLP\_5455/word2vec.c*), and the last set is the set of 300 random strings from an online opinionated Twitter corpus (<http://markahall.blogspot.sg/2012/03/sentiment-analysis-with-weka.html>).

Performance of each set is as followed:

1. Performance of the first set (section 5.3’s 300 documents):

* Precision 0.78
* Recall 0.67
* F-Measure 0.83

1. Performance of the second set (deep learning set):

* Precision 0.84
* Recall 0.73
* F-Measure 0.88

1. Performance of the third set (random set):

* Precision 0.73
* Recall 0.76
* F-Measure 0.75

As seen from the results above, the model developed with the help of the deep learning tool Word2vec gives the best performance. The main reason is that by having its features extracted with Word2vec, the documents keep most of its significant features while getting rid of insignificant ones. SVM also performs better with nominalized vectors instead of word vectors.

We can further increase the model performance by including some information from WordNet or Wikipedia as these knowledge bases provide valuable tags about the emotion expression level of words. A possible algorithm to include those tags is to assign weights to words in our corpus, with the weights be determined according to the levels of the tags.