CSI5386: Natural Language Processing
Assignment 1 : Corpus analysis and word embeddings
Report

Submitted By:
Reshma Sri Challa 300071545
Raj Kumar Endla 300058021

Work Split-up:

Task	Reshma Sri	Raj Kumar
Part 1	a)- d)	e)- g)
Part 2	Analogy Evaluation	Similarity Evaluation
Report	Part 1	Part 2

1. Part 1 : Corpus processing: tokenization, and word counting [50 points]

Libraries Used: We used NLTK and its sub modules: nltk.tokenize, nltk.FreqDist for all the text processing tasks. Regular expressions were used to remove all the non-important characters.

Also plotted the Frequencies of most frequent 70 words at each stage of Part 1.

a) Tokenizers Output: we implemented a function my tokenizer ():
   - Used Contractions library to fix tokens like don’t to do not
   - Removing non-english words (since there were some Chinese words)
   - Removing emojis/images
   - Using TweetTokenizer function of nltk library to tokenize the corpus, and stored in **microblog2011_tokenized.txt file**

Output of first 20 sentences :

1	'Save', 'BBC', 'World', 'Service', 'from', 'Savage', 'Cuts', 'http://www.petitionbuzz.com/petitions/savews'
2	'a', 'lot', 'of', 'people', 'always', 'make', 'fun', 'about', 'the', 'end', 'of', 'the', 'world', 'but', 'the', 'question', 'is', '..', '"', 'ARE', 'you', 'READY', 'FOR', 'IT', '?', '..'
3	'ReThink', 'Group', 'positive', 'in', 'outlook', ':', 'Technology', 'staffing', 'specialist', 'the', 'ReThink', 'Group', 'expects', 'revenues', 'to', 'be', 'marg', '...', 'http://bit.ly/hFjtmY'
4	'"', 'Zombie', '"', 'fund', 'manager', 'Phoenix', 'appoints', 'new', 'CEO', ':', 'Phoenix', 'buys', 'up', 'funds', 'that', 'have', 'been', 'closed', 'to', 'new', 'business', 'and', '...', 'http://bit.ly/dXrlH5'
5	'Latest', '::', 'Top', 'World', 'Releases', 'http://globalclassified.net/2011/02/top-world-releases-2/'
6	'CDT', 'presents', 'ALICE', 'IN', 'WONDERLAND', '-', 'Catonsville', 'Dinner', 'has', 'posted', '"', 'CDT', 'presents', 'ALICE', 'IN', 'WONDERLAND', '"', 'to', 'the', '...', 'http://fb.me/GMicayT3'
7	'Territory', 'Manager', ':', 'Location', ':', 'Calgary', ',', 'Alberta', ',', 'CANADA', 'Job', 'Category', ':', 'bu', '...', 'http://bit.ly/e3o7mt', '#jobs'
8	'I', 'cud', 'murder', 'sum', '1', 'today', 'n', 'not', 'even', 'flinch', 'I', 'am', 'tht', 'fukin', 'angry', 'today'
9	'BBC', News', '-', 'Today', '-', 'Free', 'school', 'funding', 'plans', '"', 'lack', 'transparency', '"', '-', 'http://news.bbc.co.uk/today/hi/today/newsid_9389000/9389467.stm'
10	'Manchester', 'City', 'Council', 'details', 'saving', 'cuts', 'plan', ':', 'http://bbc.in/fYPYPC', '...', 'Depressing', '.', 'Apparently', 'we', 'are', '4th', 'most', 'deprived', '&', 'top', '5', 'hardest', 'hit'
11	'http://bit.ly/e0ujdP', ',', 'if', 'you', 'are', 'interested', 'in', 'professional', 'global', 'translation', 'services'
12	'Fitness', 'First', 'to', 'float', 'but', 'is', 'not', 'the', 'full', 'service', 'model', 'dead', '?', 'http://bit.ly/evflEb'
13	'David', 'Cook', '!', 'http://bit.ly/fkj2gk', 'has', 'the', 'mostest', 'beautiful', 'smile', 'in', 'the', 'world', '!'
14	'Piss', 'off', '.', 'Cnt', 'stand', 'lick', 'asses'
15	'BEWARE', 'THE', 'BLUE', 'MEANIES', ':', 'http://bit.ly/hu8iJz', '#cuts', '#thebluemeanies'
16	'Como', 'perde', 'os', 'dentes', 'no', 'World', 'Of', 'Warcraft', '-', 'Via', 'Alisson', 'http://ow.ly/1beBPo'
17	'How', 'exciting', '!', 'RT', '@BunchesUK', ':', 'Hello', '!', 'what', 'is', 'happening', 'in', 'your', 'world', '?', 'we', 'are', 'all', 'gearing', 'up', 'for', '#Valentines', 'with', 'bouquets', 'flying', 'out', 'the', 'door', '.'
18	'I', 'would', 'very', 'much', 'appreciate', 'it', 'if', 'people', 'would', 'stop', 'broadcasting', 'asking', 'me', 'to', 'add', 'people', 'on', 'BBM', '.'
19	'@samanthaprabu', 'sam', 'i', 'knw', 'you', 'r', 'a', 'cricket', 'fan', 'r', 'you', 'watching', 'any', 'of', 'the', 'world', 'cup', 'matches'
20	'John', 'Baer', ':', 'Who', 'did', 'not', 'see', 'this', 'coming', '?', ':', 'TO', 'THOSE', 'who', 'know', 'Ed', 'and', 'Midge', 'Rendell', '-', 'heck', ',', 'to', 'the', 'Philly', 'world', 'at', 'la', '...', 'http://bit.ly/ii6WEO'

b) Tokens :

- Total number of tokens in the corpus : **888994**
- Total number of unique tokens : **115704**
- Type/token ratio : **0.1301516095721681**

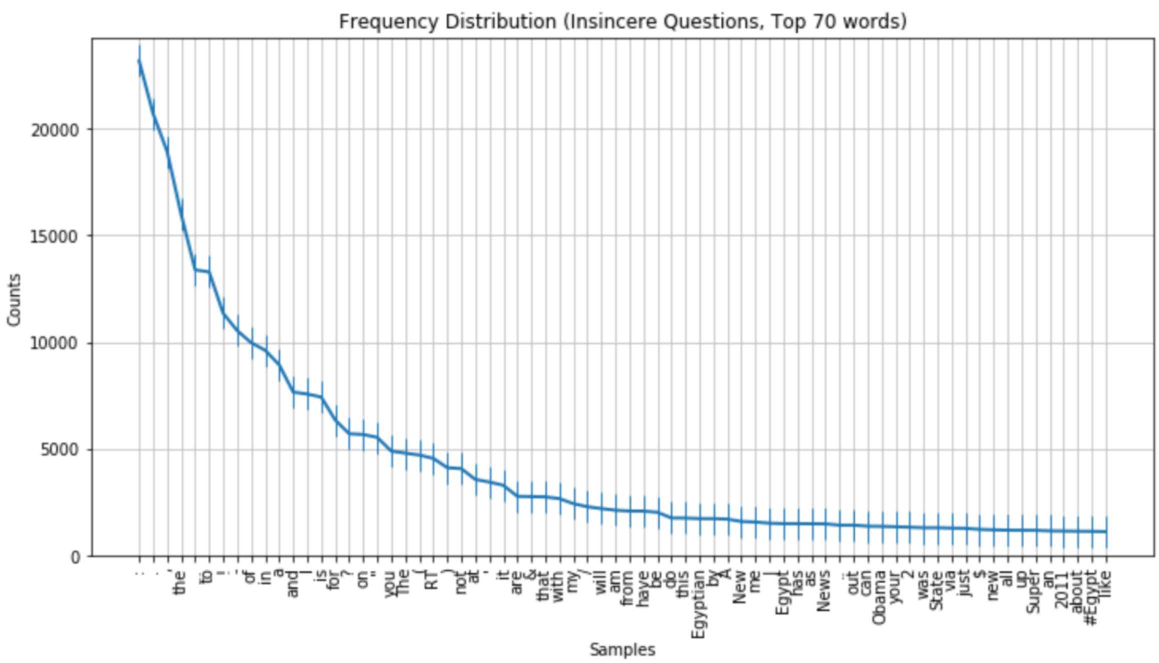
c) For each token, token name and frequency is stored in file named **tokens.txt**

- Output for first 100 lines :

1. : -> 23175
2. . -> 20696
3. , -> 18926
4. the -> 15989
5. ... -> 13381
6. to -> 13281
7. ! -> 11347
    - -> 10536
8. of -> 9966
9. in -> 9611
10. a -> 8932
11. and -> 7654
12. I -> 7562
13. is -> 7421
14. for -> 6345
15. ? -> 5701
16. on -> 5667
17. " -> 5531
18. you -> 4888
19. The -> 4794
20. (-> 4703
21. RT -> 4549
22.) -> 4111
23. not -> 4066
24. at -> 3563
25. ' -> 3438
26. it -> 3287
27. are -> 2768
28. & -> 2755
29. that -> 2743
30. with -> 2663
31. my -> 2427
32. / -> 2282
33. will -> 2195
34. am -> 2119
35. from -> 2082

36. have -> 2078
37. be -> 2024
38. do -> 1755
39. this -> 1751
40. Egyptian -> 1725
41. by -> 1724
42. A -> 1702
43. New -> 1590
44. me -> 1565
45. | -> 1510
46. Egypt -> 1490
47. has -> 1489
48. as -> 1484
49. News -> 1481
50. .. -> 1418
51. out -> 1418
52. can -> 1372
53. Obama -> 1363
54. your -> 1345
55. 2 -> 1323
56. was -> 1298
57. State -> 1298
58. via -> 1280
59. just -> 1270
60. $ -> 1222
61. new -> 1199
62. all -> 1186
63. up -> 1180
64. Super -> 1178
65. an -> 1151
66. 2011 -> 1144
67. about -> 1134
68. #Egypt -> 1127
69. like -> 1115
70. we -> 1113
71. Bowl -> 1095
72. i -> 1089
73. but -> 1065
74. now -> 1060
75. s -> 1059
76. -> 1023
77. de -> 1012
78. they -> 1000

79. so -> 980
80. US -> 953
81. get -> 943
82. In -> 932
83. what -> 904
84. 1 -> 896
85. or -> 891
86. Union -> 883
87. To -> 872
88. people -> 856
89. White -> 837
90. who -> 832
91. no -> 825
92. [-> 821
93. % -> 821
94. his -> 812
95. us -> 803
96. Social -> 792
97. he -> 789
98. more -> 787
99. #Jan25 -> 742
100. S -> 734



d) Number of tokens appeared only once in the corpus : **79216**

e) Using function clean_data() :

- Removed URLs
- Remove username
- Remove all special characters, single characters
- Removing all spaces (single and multiple)
- Removing digits
- Removing spaces from start and end
- Converting to lower case

Tokens :

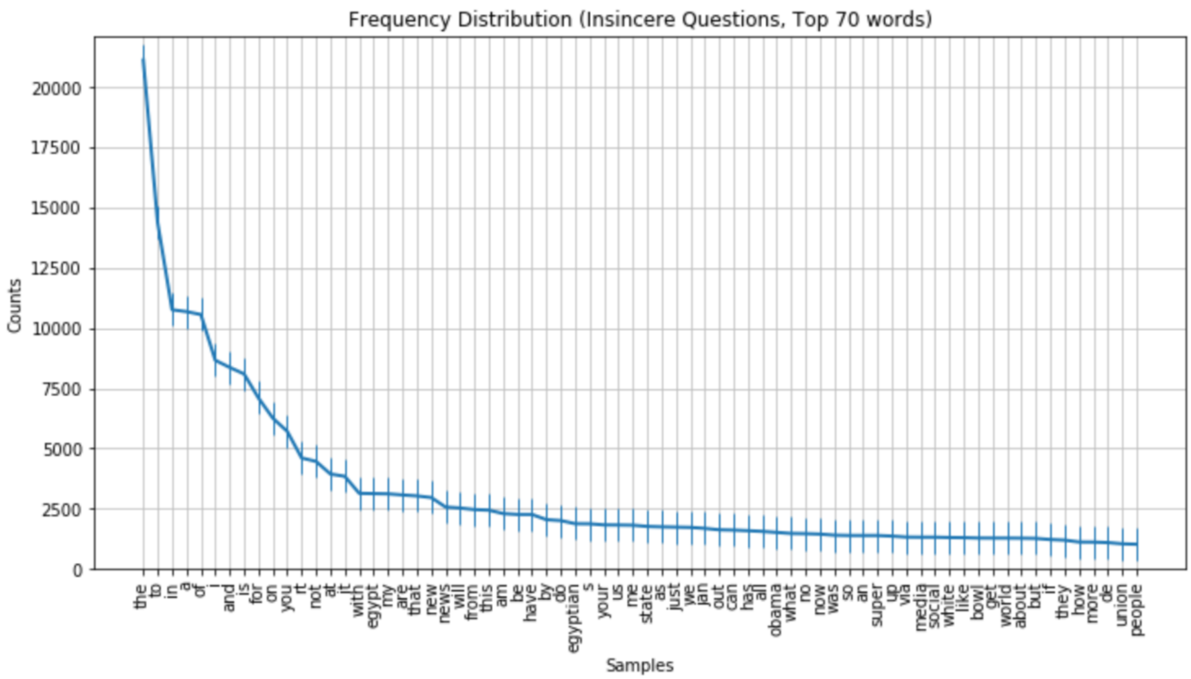
- Total number of tokens in the corpus : **689057**
- Total number of unique tokens : **54716**
- Type/token ratio : type/token ratio: **0.07940822614422528**

Output for first 100 frequent words :

1. the -> 21120
2. to -> 14363
3. in -> 10761
4. a -> 10682
5. of -> 10562
6. i -> 8664
7. and -> 8368
8. is -> 8097
9. for -> 7100
10. on -> 6261
11. you -> 5714
12. rt -> 4610
13. not -> 4467
14. at -> 3944
15. it -> 3847
16. with -> 3142
17. egypt -> 3131
18. my -> 3122
19. are -> 3075
20. that -> 3035
21. new -> 2968
22. news -> 2573
23. will -> 2532
24. from -> 2469
25. this -> 2444
26. am -> 2302
27. be -> 2266
28. have -> 2265

29. by -> 2050
30. do -> 2011
31. egyptian -> 1883
32. s -> 1878
33. your -> 1832
34. us -> 1828
35. me -> 1820
36. state -> 1770
37. as -> 1752
38. just -> 1740
39. we -> 1728
40. jan -> 1688
41. out -> 1630
42. can -> 1617
43. has -> 1591
44. all -> 1562
45. obama -> 1521
46. what -> 1478
47. no -> 1469
48. now -> 1450
49. was -> 1406
50. so -> 1393
51. an -> 1392
52. super -> 1392
53. up -> 1368
54. via -> 1327
55. media -> 1322
56. social -> 1322
57. white -> 1308
58. like -> 1302
59. bowl -> 1286
60. get -> 1285
61. world -> 1284
62. about -> 1280
63. but -> 1272
64. if -> 1226
65. they -> 1200
66. how -> 1123
67. more -> 1119
68. de -> 1099
69. union -> 1048
70. people -> 1028
71. he -> 1022

72. security -> 1013
73. love -> 1003
74. or -> 1003
75. who -> 1002
76. airport -> 999
77. day -> 983
78. release -> 954
79. president -> 929
80. law -> 926
81. his -> 921
82. one -> 920
83. today -> 903
84. time -> 899
85. good -> 888
86. video -> 879
87. house -> 873
88. jobs -> 866
89. over -> 851
90. when -> 849
91. protests -> 846
92. show -> 842
93. our -> 822
94. service -> 819
95. got -> 810
96. go -> 791
97. lol -> 789
98. mubarak -> 785
99. cairo -> 779
100. job -> 771



f) removing stop words from the corpus
Tokens :

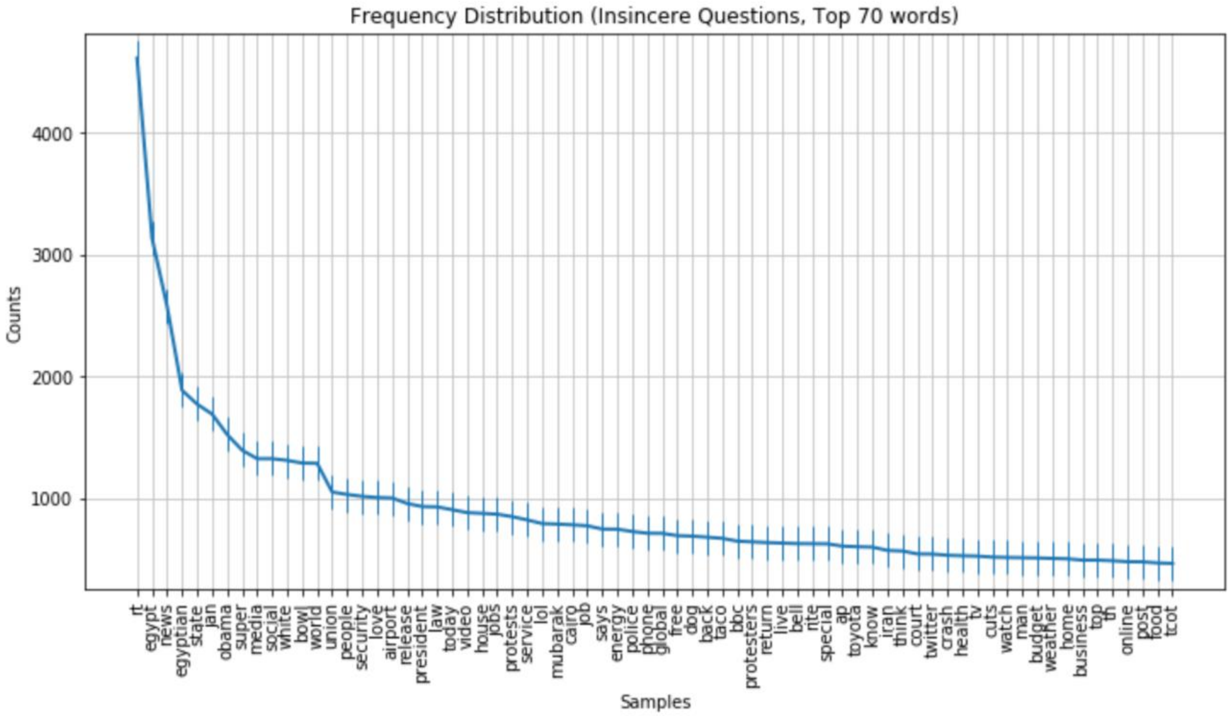
- Total number of tokens in the corpus : **405093**
- Total number of unique tokens : **54043**
- Type/token ratio : **0.13340887154307776**

Output for first 100 frequent words :

1. rt -> 4610
2. egypt -> 3131
3. news -> 2573
4. egyptian -> 1883
5. state -> 1770
6. jan -> 1688
7. obama -> 1521
8. super -> 1392
9. media -> 1322
10. social -> 1322
11. white -> 1308
12. bowl -> 1286
13. world -> 1284
14. union -> 1048
15. people -> 1028
16. security -> 1013
17. love -> 1003
18. airport -> 999

19. release -> 954
20. president -> 929
21. law -> 926
22. today -> 903
23. video -> 879
24. house -> 873
25. jobs -> 866
26. protests -> 846
27. service -> 819
28. lol -> 789
29. mubarak -> 785
30. cairo -> 779
31. job -> 771
32. says -> 744
33. energy -> 743
34. police -> 724
35. phone -> 710
36. global -> 709
37. free -> 690
38. dog -> 686
39. back -> 677
40. taco -> 668
41. bbc -> 645
42. protesters -> 639
43. return -> 633
44. live -> 629
45. bell -> 625
46. rite -> 624
47. special -> 623
48. ap -> 604
49. toyota -> 599
50. know -> 596
51. iran -> 569
52. think -> 564
53. court -> 540
54. twitter -> 539
55. crash -> 529
56. health -> 525
57. tv -> 522
58. cuts -> 514
59. watch -> 511
60. man -> 509
61. budget -> 507

62. weather -> 503
63. home -> 500
64. business -> 489
65. top -> 489
66. th -> 484
67. online -> 476
68. post -> 475
69. food -> 466
70. tcot -> 462
71. right -> 460
72. pm -> 460
73. blog -> 452
74. address -> 452
75. organic -> 452
76. car -> 451
77. peace -> 450
78. attack -> 448
79. big -> 423
80. help -> 421
81. protest -> 411
82. museum -> 411
83. mexico -> 406
84. pakistan -> 406
85. fifa -> 402
86. haiti -> 401
87. check -> 398
88. life -> 394
89. work -> 390
90. government -> 389
91. internet -> 388
92. jordan -> 382
93. recovery -> 379
94. date -> 377
95. reuters -> 374
96. call -> 372
97. cut -> 369
98. auto -> 368
99. watching -> 364
100. black -> 363



g) All pairs of consecutive words (bigrams):

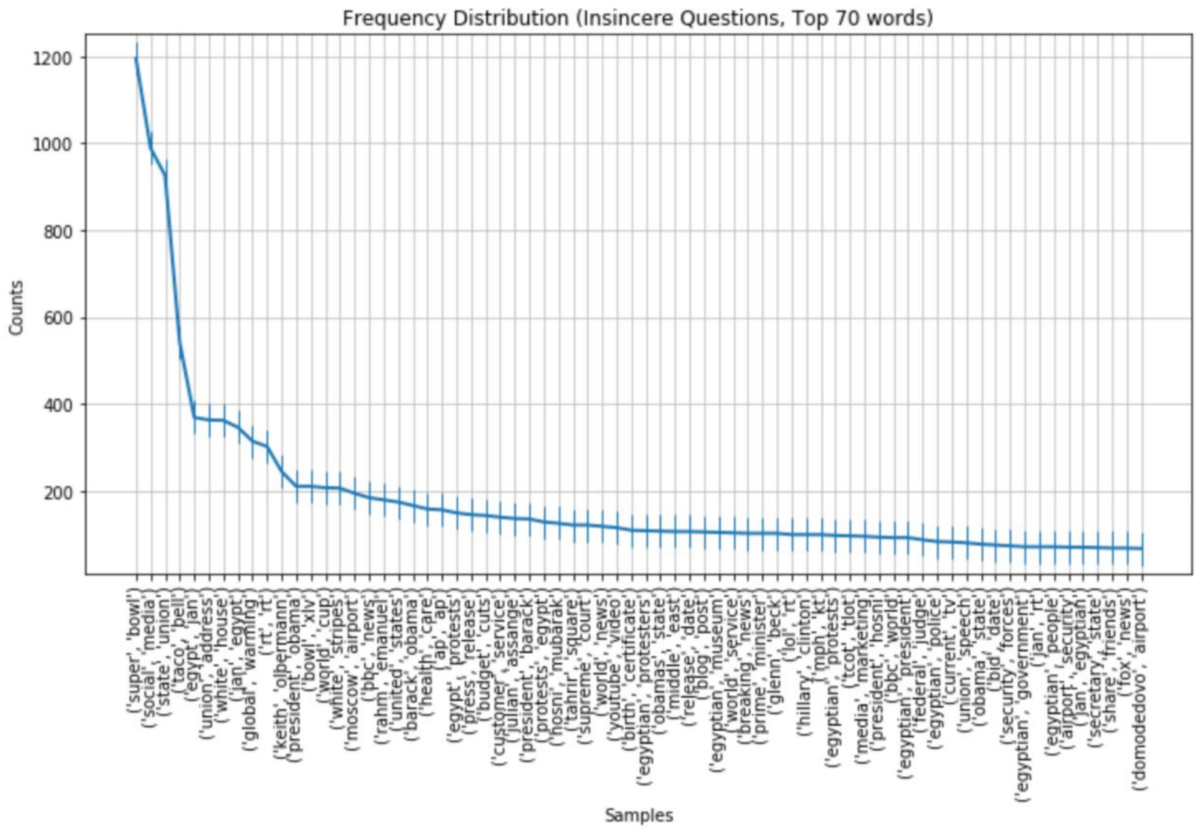
- Using function n_grams : using ngrams from nltk

Output for first 100 frequent words :

('super', 'bowl') -> 1195
('social', 'media') -> 989
('state', 'union') -> 926
('taco', 'bell') -> 542
('egypt', 'jan') -> 369
('union', 'address') -> 363
('white', 'house') -> 362
('jan', 'egypt') -> 346
('global', 'warming') -> 314
('rt', 'rt') -> 302
('keith', 'olbermann') -> 244
('president', 'obama') -> 210
('bowl', 'xlv') -> 210
('world', 'cup') -> 207
('white', 'stripes') -> 206
('moscow', 'airport') -> 194
('bbc', 'news') -> 184
('rahm', 'emanuel') -> 179
('united', 'states') -> 174
('barack', 'obama') -> 166

('health', 'care') -> 158
('ap', 'ap') -> 156
('egypt', 'protests') -> 149
('press', 'release') -> 145
('budget', 'cuts') -> 143
('customer', 'service') -> 139
('julian', 'assange') -> 136
('president', 'barack') -> 135
('protests', 'egypt') -> 128
('hosni', 'mubarak') -> 125
('tahrir', 'square') -> 121
('supreme', 'court') -> 121
('world', 'news') -> 118
('youtube', 'video') -> 115
('birth', 'certificate') -> 109
('egyptian', 'protesters') -> 108
('obamas', 'state') -> 107
('middle', 'east') -> 106
('release', 'date') -> 106
('blog', 'post') -> 105
('egyptian', 'museum') -> 104
('world', 'service') -> 103
('breaking', 'news') -> 102
('prime', 'minister') -> 102
('glenn', 'beck') -> 102
('lol', 'rt') -> 99
('hillary', 'clinton') -> 99
('mph', 'kt') -> 99
('egyptian', 'protests') -> 97
('tcot', 'tlot') -> 96
('media', 'marketing') -> 95
('president', 'hosni') -> 93
('bbc', 'world') -> 92
('egyptian', 'president') -> 92
('federal', 'judge') -> 87
('egyptian', 'police') -> 83
('current', 'tv') -> 82
('union', 'speech') -> 80
('obama', 'state') -> 77
('bid', 'date') -> 75
('security', 'forces') -> 73
('egyptian', 'government') -> 71
('jan', 'rt') -> 71
('egyptian', 'people') -> 71
('airport', 'security') -> 70
('jan', 'egyptian') -> 70

```
('secretary', 'state') -> 69
('share', 'friends') -> 68
('fox', 'news') -> 68
('domodedovo', 'airport') -> 67
('phone', 'hacking') -> 67
('special', 'olympics') -> 67
('international', 'airport') -> 66
('fifa', 'soccer') -> 65
('egyptian', 'embassy') -> 65
('bowl', 'super') -> 65
('gabrielle', 'giffords') -> 65
('tear', 'gas') -> 64
('rt', 'egyptian') -> 64
('cowboys', 'stadium') -> 64
('kate', 'middleton') -> 64
('unemployment', 'rate') -> 64
('egyptian', 'army') -> 63
('global', 'war') -> 63
('cell', 'phone') -> 62
('egypt', 'rt') -> 62
('egypt', 'egyptian') -> 61
('anthony', 'hopkins') -> 61
('louis', 'vuitton') -> 61
('president', 'obamas') -> 60
('state', 'tv') -> 60
('climate', 'change') -> 59
('care', 'law') -> 59
('social', 'networking') -> 58
('jan', 'jan') -> 58
('shorty', 'award') -> 58
('iranelection', 'iran') -> 58
('weight', 'loss') -> 57
('judge', 'rules') -> 57
('state', 'hillary') -> 57
('cairo', 'egypt') -> 57
```



2. Part 2: Evaluation word embeddings [50 points]

The word embedding used are Glove (wiki-6B, twitter-27B, common-crawl-840B), SG_googleNews, LexVec (commoncrawl-W+C, commoncrawl-ngramsubwords-W), PDC, HDC, conceptnet_numberbatch, FastText.

Word2Vec: there are 2 types of model architectures:

1. CBOW: continuous bag of words, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (as name says it is bag of words assumption)
2. Skip-gram : continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words. CBOW is faster while skip-gram is slower but does a better job for infrequent words.

Dimension of these word vectors is typically set between 100 and 1000
Extensions of word2vec are paragraph2vec and doc2vec to construct embeddings from entire documents rather than individual words.

GloVe : Global Vectors is developed as an open source project at Stanford, which is a model for distributed word representation , model is an unsupervised learning algorithm, training is performed on aggregated global word-word cooccurrence statistics from a large corpus. Glove combines features of both global matrix featurization and local context window methods.

FastText : This is an extension to word2vec.Instead of feeding individual words into the Neural Network, FastText breaks words into several n-grams (sub-words). For instance, the tri-grams for the word *apple* is *app, ppl*, and *ple* (ignoring the starting and ending of boundaries of words). The word embedding vector for *apple* will be the sum of all these n-grams. After training the Neural Network, will have word embeddings for all the n-grams given the training dataset. Rare words can be properly represented since it is highly likely that some of their n-grams also appears in other words.

Lexvec :Is a method for generating distributed word representations that uses low rank, weighted factorization of the positive point-wise mutual information matrix via stochastic gradient descent, employing a weighting scheme that assigns heavier penalties for errors on frequent co-occurrence while still accounting for negative co-occurrence.

PDC : (Parallel Document Context) : In this model, target word is predicted by its surrounding context as well as the document it occurs in, this model can also be viewed as an extension to CBOW model by adding an extra document branch.

HDC : (Hierarchical Document Context): In this model the prediction is conducted in a hierarchical manner. Similar as the PDC model, the syntagmatic relation in HDC is modeled by the document-word prediction layer and the word context prediction layer models the paradigmatic relation.

ConceptNet : It is a whimsical double-dactyl name for pre-computed word embeddings built using conceptNet and distributional semantics, propagation is a step which is added to build process of conceptnet numberbatch which makes it easier to use Numberbatch to represent a larger vocabulary of words, especially in languages with more inflections than English.

Propagate step they pre compute the vectors for more words,especially for out of Vocab words which eliminates the need for looking into the whole ConceptNet graph, this improves greatly for users of numberbatch who aren't using OOV strategy at all.

Step 1: Download GitHub zip folder from https://github.com/kudkudak/word-embeddings-benchmarks
Step 2: Unzip the folder in the location where you want to run the python files and run the following commands in cmd prompt from this folder (use python 3.5 for analogy tasks):

1. pip install -r requirements.txt
2. pip install jupyterlab
3. python setup.py install
4. jupyter notebook

Step 3: To download embeddings, import fetch functions for each embedding from web.embeddings file

The word embedding that work from this folder are Glove, SG_googleNews, LexVec, PDC, HDC, conceptnet_numberbatch.

For FastText :

- Word embeddings are available in link: fasttext.cc/docs/en/english-vectors.html,
- 4 variants of fasttext word embeddings are available, we have downloaded wiki-news-300d-1M.vec.zip
- Unzip this zip folder
- Go to C:\Users\user_name\web_data\embeddings and paste this folder
- Go to word-embeddings-benchmarks/web/embeddings.py – in fetch_FastText function , change the url_vec = "C:\Users\user_name\web_data\embeddings\ wiki-news-300d-1M.vec"

Step 4 : Go to examples folder (in Jupyter notebook)

Part A : **Evaluation on Similarity:**

1. open evaluate_similarity.ipynb :
2. Fetch all 8 datasets ,shown in the table
3. To fetch all the benchmark 8 datasets , import fetch functions for each dataset from web.datasets.similarity for similarity datasets
4. The given fetch functions for all the datasets work except TR9856, which can be done using the following.

   For fetch_TR9856():
   - Download https://www.research.ibm.com/haifa/dept/vst/files/IBM_Debater_(R)_TR9856.v2.zip
   - Unzip this zip folder
   - go to C:\Users\reshm\web_data\similarity and paste this folder in a folder named "IBM_Debater_(R)_TR9856.v2"
   - Now fetch_TR9856()

5. Run word embeddings on all benchmark datasets, and the results are as follows:

	Glove-Wiki	**Glove-Twitter**	**Glove-Commn Crawl**	**SG google news**	**LexVec-common crawl**	**LexVec-common Crawl Ngram subword**	**PDC**	**HDC**	**Concept-net**	**Fast Text-Wiki**
MTurk	0.63	0.56	0.69	0.68	**0.71**	**0.71**	0.67	0.65	**0.71**	0.70
MEN	0.73	0.57	0.80	0.75	0.80	0.80	0.77	0.76	**0.85**	0.79
WS353	0.54	0.46	0.73	0.70	0.69	**0.75**	0.73	0.71	**0.75**	0.73
R and G	0.79	0.67	0.76	0.76	0.76	0.74	0.79	0.80	**0.90**	0.84
Rare Words	0.36	0.23	0.45	0.49	0.48	0.53	0.47	0.46	**0.54**	0.51
SimLex999	0.37	0.12	0.40	0.44	0.41	0.47	0.42	0.40	**0.65**	0.44
TR9856	0.096	0.092	0.098	0.18	0.12	0.13	**0.20**	**0.20**	0.13	0.15
Average	0.502	0.386	0.561	0.571	0.567	0.59	0.578	0.568	**0.647**	0.594

Observations:

- The highlighted results for each word dataset are the best word embeddings.
- **ConceptNet seems to work well with most of the Datasets for similarity evaluation with an average of 0.647.**

Part B: Evaluation on Analogy:

1. Create evaluate_analogy.ipynb similar to evaluate similarity:
2. Fetch all the four 4 datasets, shown in the table
3. To fetch Google analogy and MSR benchmark datasets, import fetch functions for each dataset from web.datasets.analogy
4. To fetch MSR WordRep and SEMEVAL 2012 benchmark datasets, import fetch functions for each dataset from web.datasets.evaluate
5. Run word embeddings on all benchmark datasets, and the results are as follows:

	Glove-Wiki	**Glove-Twitter**	**Glove-Common Crawl**	**SG google news**	**LexVec-common crawl**	**LexVec-common Crawl Ngram subword**	**PDC**	**HDC**	**Concept-net**	**Fast Text-Wiki**
MSR WordRep (200 pairs)	0.19	0.06					0.21	0.21		
MSR WordRep (100 pairs)	0.21	0.08	0.24	0.25	0.23	0.22	0.24	0.24	0.15	**0.26**
Google analogy	**0.71**	0.42	0.70	0.40	**0.71**	0.58	0.74	0.73	0.38	0.59
MSR	0.61	0.42	0.74	0.71	0.60	0.68	0.59	0.56	0.53	**0.81**
SEMEVAL 2012	0.16	0.14	0.17	0.20	0.16	0.21	0.17	0.18	**0.23**	0.21
Average	0.422	0.265	0.462	0.39	0.425	0.422	0.435	0.427	0.322	**0.467**

Observations:

- The highlighted results for each word dataset are the best word embeddings.
- The results for WordRep dataset are on 200 pairs (only with few embeddings) and 100 pairs (all embeddings) as 1000 pairs runtime was 8-10 hours per each embedding.
- The results when considering more number of pairs might be increasing the negatives therefore having less evaluation score for 200 pairs, 100 pairs might be having more positives, therefore having greater evaluation score.
- Average scores are using 100 pairs for WordRep dataset and 1000 pairs for other datasets.
- The highlighted results for each word dataset are the best word embeddings.
- **FastText seems to work well with most of the Datasets for analogy evaluation with an average of 0.467**.