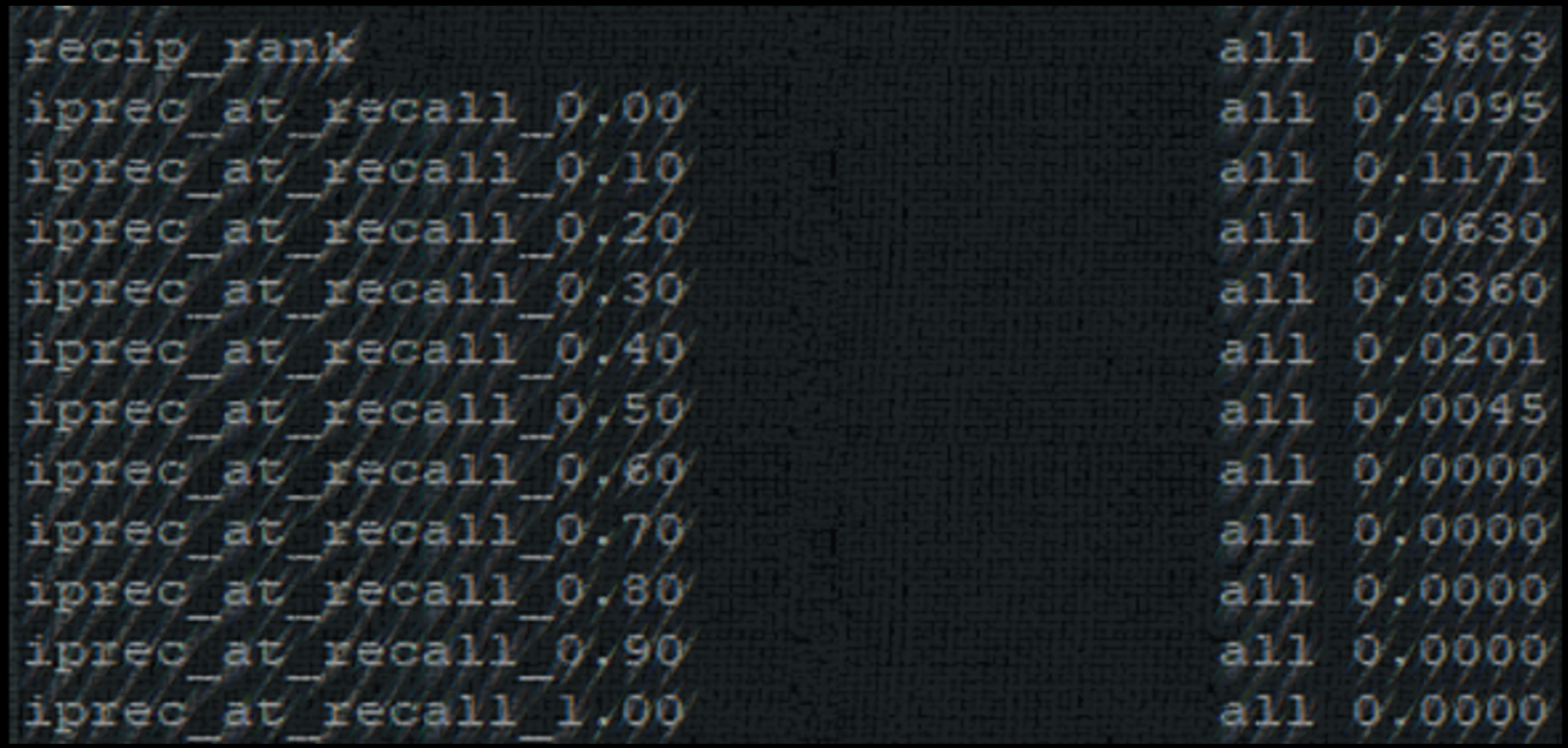
Natural Language Processing

(Assignment 2)

By Urmi Patel (501064008)



Approach 1:

1. Used tokenized ‘Abstract’ from dataset
2. Preprocessed given dataset (for 30 articles)

   Used ‘Description’ tag to get similar articles

   - Tokenization
   - Remove punctuation
   - Remove stop word
3. Tried 3 methods for embedding
   - word2vec method
   - Fasttext (Facebook)
   - Centroid method

   Trained model with parameters like, size=300, window=8, epochs=10
4. Generated TF-IDF vector

Word2Vec and FastText

- Word2Vec treats each word in corpus like an atomic entity and generates a vector for each word.
- FastText treats each word as composed of character n-grams. So, the vector for a word is made of the sum of the character n-grams.

```
print('Words similar to "health" are: ', word_vectors.most_similar(positive='health'))
```

```
Words similar to "size" are:  [('healthcare', 0.6715487837791443), ('heath', 0.6228415966033936), ('welfare', 0.5154505968093872), ('goods', 0.5103880167007446), ('social', 0.5087597370147705), ('medical', 0.48441797494888306), ('policies', 0.45780831575393677), ('policy', 0.45311176776885986), ('childcare', 0.44871985912323), ('government', 0.4453142881393934326)]
```

```
print('Words similar to "health" are: ', ft_model.wv.most_similar('health'))
```

```
Words similar to "size" are:  [('philhealth', 0.7763347625732422), ('subhealth', 0.7709438800811768), ('ecohealth', 0.7696387767791748), ('ethealth', 0.7689062356948853), ('healthcare', 0.7504271268844604), ('healthlinks', 0.7399744391441345), ('healthcore', 0.7309108376502991), ('healthmap', 0.7276439666748047), ('publichealth', 0.7214188575744629), ('healthtrust', 0.7161760926246643)]
```

- FastText model takes more time to train but it performs better than Word2Vec and allows rare words to be represented appropriately.

Approach 1 (cont’d):

5. Sentence embedding
   - Use of n-grams really does play a key role in word embeddings and hence, I will proceed with using FastText embeddings as a basis for sentence embeddings.
   - Used centroid method to derive the sentence embeddings. It derives sentence embeddings as the sum of individual word embeddings in a sentence weighted by their tf-idf score and divided by the sum of these tf-idf scores.

6. Finding Similarity
   - Used cosine distance to find the similarity score for each topic

Output File

- TOPIC -> topic id of given file
- PMCID -> unique article id of each file
- SCORE -> similarity score
- RUN_NAME -> any file name

TOPIC_NO	Q0	PMCID	RANK	SCORE	RUN_NAME
27	0	3603484	0	0.3222645509385643	file_27
27	0	3849800	1	0.3222492195132514	file_27
27	0	3082229	2	0.3222288647361635	file_27
27	0	3315738	3	0.3222651053785833	file_27
27	0	3220242	4	0.3221670886894805	file_27
27	0	3015811	5	0.3220648874356956	file_27
27	0	3204556	6	0.3220471827184549	file_27
27	0	2885343	7	0.3220030127118218	file_27
27	0	2803799	8	0.3219592746694377	file_27
27	0	3015381	9	0.3219230099080117	file_27
27	0	3016801	10	0.3219096203251019	file_27
27	0	1560165	11	0.3218812933195112	file_27
27	0	3092938	12	0.3218614695861285	file_27
27	0	3449196	13	0.3218519488378881	file_27
27	0	3515931	14	0.3217313564143275	file_27
27	0	2783120	15	0.3216084146201170	file_27

Evaluation:

- Download and unzip trc_eval_9.0.7 folder
- Open Terminal window
- Compile trec_eval.c file
- Run output result file and compare with already given ‘qrels2014.txt’ file

  (./trec_eval qrels2014.txt. result_file.txt)

- It will give you bunch of scores

Result for Approach 1

- Generated top 1000 most similar articles for each of 30 given topics (order highest to lowest)
- Total 30000 rows with score and article id
- Total number of relevant documents (according to the qrels file) = 3356
- Total number of relevant documents retrieved (in the results file) = 443

```
zsh: command not found: .trec_eval
urmi@Urmis-MacBook-Pro trec_eval-9.0.7 2 % ./trec_eval qrels2014.txt Out_final.txt
runid                   all     file_30
num_q                   all     30
num_ret                 all     30000
num_rel                 all     3356
num_rel_ret             all     443
map                     all     0.0020
gm_map                  all     0.0007
Rprec                   all     0.0065
bpref                   all     0.0632
recip_rank              all     0.0190
iprec_at_recall_0.00    all     0.0262
iprec_at_recall_0.10    all     0.0129
iprec_at_recall_0.20    all     0.0057
iprec_at_recall_0.30    all     0.0009
iprec_at_recall_0.40    all     0.0000
iprec_at_recall_0.50    all     0.0000
iprec_at_recall_0.60    all     0.0000
iprec_at_recall_0.70    all     0.0000
iprec_at_recall_0.80    all     0.0000
iprec_at_recall_0.90    all     0.0000
iprec_at_recall_1.00    all     0.0000
P_5                     all     0.0000
P_10                    all     0.0067
P_15                    all     0.0067
P_20                    all     0.0050
P_30                    all     0.0033
P_100                   all     0.0060
P_200                   all     0.0070
P_500                   all     0.0070
P_1000                  all     0.0148
urmi@Urmis-MacBook-Pro trec_eval-9.0.7 2 %
```

Approach 2:

- Used stemmed tokenized abstract
- Preprocessed given dataset (for 30 articles), also stemmed
- Used Doc2vec method and trained the model
- Parameter used (size=100, epoch=20 and so on)
- Analyzed the output by using model.infer_vector()

```
TOPIC_NO Q0 PMCID RANK SCORE RUN_NAME
1 0 2437844 0 0.6684660911560059 DocVec
1 0 2904791 1 0.6469342112541199 DocVec
1 0 2008204 2 0.6301003694534302 DocVec
1 0 3177926 3 0.619623064994812 DocVec
1 0 2740240 4 0.6143605709075928 DocVec
1 0 1629293 5 0.6127108931541443 DocVec
1 0 2933598 6 0.6096243858337402 DocVec
1 0 2111731 7 0.6095649003982544 DocVec
1 0 2917397 8 0.6093626618385315 DocVec
1 0 2652425 9 0.6047068238258362 DocVec
1 0 2589807 10 0.6042107939720154 DocVec
```

Hyper Parameter Tuning (Approach 2)

Vector Size	Number of epochs	Score
N= 50	20	0.49-0.53
N=100	20	0.67-0.71
N=300	20	0.37-0.41

Without stemming		
N=50	20	0.47-0.49
N=100	20	0.50-0.53

Result for Approach 2

- Generated top 1000 most similar articles for each of 30 given topics (order highest to lowest)
- Total 30000 rows with score and article id
- Total number of relevant documents (according to the qrels file) = 3356
- Total number of relevant documents retrieved (in the results file) = 775
- P_5 -> 0.1800 (Precision of the first 5 documents)
- P_10 -> 0.1500
- P_15 -> 0.1333
- P_20 -> 0.1283

num_q	all	30
num_ret	all	30000
num_rel	all	3356
num_rel_ret	all	775
map	all	0.0344
gm_map	all	0.0052
Rprec	all	0.0724
bpref	all	0.1118
recip_rank	all	0.3683
iprec_at_recall_0.00	all	0.4095
iprec_at_recall_0.10	all	0.1171
iprec_at_recall_0.20	all	0.0630
iprec_at_recall_0.30	all	0.0360
iprec_at_recall_0.40	all	0.0201
iprec_at_recall_0.50	all	0.0045
iprec_at_recall_0.60	all	0.0000
iprec_at_recall_0.70	all	0.0000
iprec_at_recall_0.80	all	0.0000
iprec_at_recall_0.90	all	0.0000
iprec_at_recall_1.00	all	0.0000
P_5	all	0.1800
P_10	all	0.1500
P_15	all	0.1333
P_20	all	0.1283
P_30	all	0.1144
P_100	all	0.0800
P_200	all	0.0603
P_500	all	0.0377
P_1000	all	0.0258

Conclusion

- By looking at the results, I observed that when FastText method was used to find similarity between Articles and Topics, less impressive results compare to Doc2Vec were provided
- Realized that different parameter tuning also plays very important role in any task.
- Combining FastText and Centroid method proved to be very risky for this task.

Reference Links

- http://www2.aueb.gr/users/ion/docs/BioNLP_2016.pdf
- https://radimrehurek.com/gensim/models/word2vec.html
- https://www.tutorialspoint.com/gensim/gensim_doc2vec_model.htm
- http://www.rafaelglater.com/en/post/learn-how-to-use-trec_eval-to-evaluate-your-information-retrieval-system

Thank you