IR 1 - Homework 2

Puck de Haan
11305150
pdehaan274@gmail.com

Tim van Loenhout
10741577
timvanloenhout@gmail.com

Ard Snijders
12854913
ardsnijders@gmail.com

1 LOADING AND PROCESSING THE AP DATASET

TQ 1.1. If we were using a dataset in the medical domain, we would have chosen a slightly different approach to the preprocessing. Since in the medical domain a small detail can make a big difference, it might not be a good idea to remove all stopwords and apply stemming. Lowercasing on the other hand would still be beneficial, and maybe removing only the most frequent stopwords, such as determiners, would be as well.

2 WORD2VEC / DOC2VEC MODELS

Implementation

Word2vec. For the word2vec model, batches are sampled by starting at a random target word in the complete corpus, consisting of word id's. From here word id pairs are constructed for a tunable number of surrounding context words. The used window size is 10, which is based on the hyper-parameter tuning of the doc2vec model. A pair is only considered if both the target and context word occur in the corpus at least 50 times. Consequently, each target word id occurs in the batch list multiple times, each time with another corresponding context word id. Furthermore, for each pair, a tunable number of random id's within the range of the vocabulary size are used as negative samples and stored in a matrix. We used 10 negative samples per pair, also based on the doc2vec tuning. For an example of a retrieved batch, see figure 1.

```
corpus           ['this', 'is', 'a', 'test', 'corpus', 'for', 'batch', 'sampling']
target_words     [381, 381, 381, 381, 726, 726, 726, 726, 2, 2, 2, 2]
context words    [1018, 8828, 726, 2, 8828, 381, 2, 1625, 381, 726, 1625, 8274]
negative samples tensor([[18298,  3277, 24381,  8861, 17718],
        [17416, 23537, 16763, 22339, 19356],
        [10720,  7785, 20397, 13278,  2466],
        [13413,   626, 15116, 21011, 13439],
        [ 7925, 16448, 25143, 24949,  1520],
        [ 4034, 11621, 15837,  3834, 22756],
        [11609, 19723,  8062,   963, 22224],
        [18322, 16983,  1944,  4669, 21825],
        [22659, 22443,    35, 15130, 11628],
        [14370,  4683, 13886,  1023, 17650],
        [  956, 22214,  1832, 10703,  8678],
        [24525, 22486, 21233,  9669,  6987]])
```

Figure 1: Word2vec batch sampling example, with batch size 12, window size 5 and 5 negative samples.

During training, the batch lists and matrix are converted to long tensors and fed forward through an embedding layer (dimension 200, based on the doc2vec tuning). The embedding layers are initialized with the Pytorch Xavier initialization. Subsequently, by using matrix multiplication, the retrieved word embeddings are compared, resulting in a similarity score for each positive sample pair. As these scores are negatively correlated with the loss – that is high scores should result in a decrease of the loss – the sign of the scores are switched. A similarity score is also obtained for the negative samples, however for these, the sign is not switched as we want to enforce the embedding to be similar to its semantically similar words, and different from all others. Finally, the losses for the positive and negative pairs are summed and used for backpropagation, after which the parameters are updated using the SparseAdam optimizer with learning rate 2e-3.

After 20000 iterations with batches of size 1024, the trained matrix consisting of word embeddings can be used to construct a document/query representation. This is done by assuming the document/query to be a sum of its parts, hence its embedding can be constructed by taking the mean over the embeddings of all its individual words. Subsequently, the cosine similarity can be computed for a query and document by taking the normalized dot product over their dense vector representations, indicating the document relevance. Finally, based on the rankings for a set of queries, the MAP and NDCG score for the trained model is computed and stored in table 2.

Doc2vec. For the implementation of the doc2vec model, we used the gensim library [2]. The model was trained using all documents with a variable vector dimension, window size and vocabulary size. We used 20 epochs to train the model, since that way the training time was still feasible and the gensim documentation recommended this amount of epochs for a dataset our size.

When the model was trained, document vectors could be inferred from the model. To do so, we set the the epochs hyperparameter to 200, since it increased the inferred vectors stability and made for better results during evaluation.

Analysis Questions

AQ 2.1. Table 1 shows that for some words, such as 'ship', 'war' and 'car', the most similar words based on the word2vec representation make sense semantically. For instance, 'militari', 'power' and 'countri' are likely to occur in the same context as 'war'. However, for some other words, such as 'president', the most similar words do not seem to show much semantic similarity. For instance, '1.25', 'scrap' and 'consists' are not words that are typically associated with

IR 1, February/March, MSc AI, UvA
2020.

'president'. We did use a window size of 10, based on the hyper-parameter tuning for doc2vec, so despite these words not directly relating to 'president', they could still occur in the context window. Nevertheless, the absence of any apparent similar words indicates that the model might not have been training long enough to construct a semantically meaningful representation for all words.

ship	president	war	car
injur	abolish	militari	injur
soldier	scrap	countri	passing
southeast	wedtech	iran	truck
port	hideout	rebel	villag
border	1.25	power	bomb
aircraft	conven	communist	ethnic
river	consists	rule	southwest
plane	guidanc	claim	riot
coast	3-4	arm	patrol

Table 1: Most similar words to 'ship', 'president', 'war' and 'car' based on the word2vec embeddings cosine similarity.

AQ 2.2. To test the first implementation of our Doc2Vec model, we used the first few paragraphs of a New York Times article: "Trump Has a Problem as the Coronavirus Threatens the U.S.: His Credibility" [1]. The top 10 most similar docs from the dataset all had a similarity measure between 0.39 and 0.32. In our opinion, they were all fairly similar documents since they all mentioned a president of the US. If the reader is interested to take a look at some of the top-ranked documents given this query, the top three documents were: **AP880513-0248, AP890426-0122, AP891120-0081**. However, apart from the fact that the main subject of these documents was the president, there was not a lot of similarity. This is probably the case since the AP news dataset consists of news articles from 30 years ago, so there are definitely no documents in the data that overlap a lot with current news items.

Theory Questions

TQ 2.1. Embedding models might not be suitable for retrieval, because these models do not take into account the word order. That is, the embeddings for words in a document/query are computed independently due to which the temporal information is not captured. Another thing that should be taken into account is that an embedding model needs a lot of training on a very large dataset, before the word embeddings can be taken to be representative of the real word meanings. Thus, for companies like Google, that have vast amounts of data and hardware, word embedding might be a nice addition to their IR models.

TQ 2.2. In Doc2Vec/Word2Vec the model "learns" the word embeddings on its own, while in the methods encountered earlier we specified the embeddings for all words. In the previous assignment we specified what should be the feature vector of a specific word or document beforehand, using formulas with the term frequencies, document frequencies or intricate combinations of both (e.g. TF-IDF or BM25). Now, however, we just specify the dimension of the feature vectors and the model uses words and their contexts in the corpus to come up with the "best" feature vectors for all words.

3 LATENT SEMANTIC INDEXING AND LATENT DIRICHLET ALLOCATION

Analysis Questions

AQ 3.1. For LSI, it appears that the TF-IDF term-document matrix yields the most coherent topics about particular subjects. For instance, one of the top TF-IDF-weighted LSI topics is comprised of terms such as "cent", "bushel", "stock", "percent", "lower", "higher", which are all market/trade related terms. Similarly, another topic contains terms such as "soviet", "palestinian", "israel", "bush", "gorbachev" and "arab", which arguably all belong to the subject of politics. For LSI with (binary) BOW-weighted matrices, topics appear to be more mixed across different subjects, with single topics containing both politics and market related terms. Within topics, there are still some logical groupings of terms to be found, - one topic includes political terms such as "presid", "parti", "democrat" and "state" - but the same topic also includes less political terms such as "market", "price", "million" and "dukaki". These results suggest that LSI models trained on TF-IDF weighted term-document matrices are more capable of capturing single particular subjects, compared to LSI models trained on BOW term-document matrices.

In the case of LDA, training a model for 500 topics yields significantly different results. There is a large variety of topics, and most of the topics themselves are more specific. For instance, one topic includes the words 'fire', 'flame', 'burn', 'firefight', 'blaze'; Another topic includes words such as 'moon, 'earth', 'atmospher' and 'planet'.

Based on these results, it seems that LSI is more appropriate for finding the most prevalent topics in the corpus (given that the corpus is comprised of news articles, it seems logical that the top topics are related to subjects such as politics and business/economics). In contrast, LDA works well for finding a more complete set of various topics that occur in the corpus, where each topic is comprised of a set of words that belong to a very particular domain - though it should be noted that arguably, the degree of this granularity is directly related to the 'desired' number of topics as provided to the model prior to training, and may be subject to change for higher/lower values of this parameter.

Theory Questions

TQ 3.1. Both methods take as input a word-document matrix, with either the corresponding BOW values or TF-IDF values. LSA is used to decompose the word-document matrix with SVD and return a low-rank approximation of the matrix, that only captures the highly-informational dimensions. Subsequently, documents and queries can be represented as vectors in this lower-dimensional semantic space and matched using cosine similarity. LDA, on the other hand, is a probabilistic generative model that models a corpus by learning the latent topics that are present. Each topic consists of a set of words and each document can then be represented by a mixture of topics.

4 RETRIEVAL AND EVALUATION

Analysis Questions

AQ 4.1. The retrieval performance of the models with default settings was tested for both all queries and also for the validation set of queries (76-100). For both LSI models the default number of topics was 500, for the default doc2vec model we used a window size of 10, vector dimension of 300 and a vocabulary size of 50000. For the word2vec model we used a window size of 5, a vector dimension of 200, 10 negative samples per target word and only took words into account that appeared more than 50 times in the dataset.

	MAP		nDCG	
	All	76-100	all	76-100
TF-IDF	0.220	0.182	0.582	0.541
word2vec	0.0026	0.0043	0.29	0.28
doc2vec	0.0040	0.0033	0.293	0.292
LSW-BoW	0.081	0.058	0.449	0.423
LSI-TF-IDF	0.143	0.110	0.526	0.488

Table 2: Retrieval performance in terms of MAP and nDCG for models with default parameters. *The parameters for word2vec are based on the optimal parameters obtained from the doc2vec tuning.

AQ 4.2. We used significance testing to determine whether the difference in results between models could be due to chance. Based on a significance treshold of $p = 0.05$ it can be seen that all differences in results were significant, except for the difference between the word2vec and doc2vec models (Table 3). This could be dueto the fact that both models use roughly the same technique. That is despite the doc2vec model using an additional doc represenation vector, both models rely on the combination of individual word2vec embeddings.

	MAP		nDCG	
	t	p	t	p
TF-IDF / word2vec	12.3	$2 * 10^{-24}$	17.5	$6 * 10^{-38}$
TF-IDF / doc2vec	12.3	$2 * 10^{-24}$	16.7	$6 * 10^{-36}$
TF-IDF / LSI-B	7.71	$1.6 * 10^{-12}$	7.22	$2.6 * 10^{-11}$
TF-IDF / LSI-T	4.54	$1.1 * 10^{-5}$	3.42	$2.6 * 10^{-4}$
word2vec / doc2vec	-1.14	0.25	-1.50	0.14
word2vec / LSI-B	-7.46	$6.8 * 10^{-12}$	-14.4	$5.6 * 10^{-30}$
word2vec / LSI-T	-9.94	$3.8 * 10^{-18}$	-18.4	$5.5 * 10^{-40}$
doc2vec / LSI-B	-7.27	$1.9 * 10^{-11}$	-12.7	$1.6 * 10^{-25}$
doc2vec / LSI-T	-9.86	$5.8 * 10^{-18}$	-17.3	$2.5 * 10^{-37}$
LSI-B / LSI-T	-6.40	$2.0 * 10^{-9}$	-8.20	$1.1 * 10^{-13}$

Table 3: T-statistic (t) and p-value (p) for combinations of the models: word2vec (word), doc2vec (doc), TF-IDF, LSI-BoW (LSI-B), LSI-TF-IDF (LSI-T).

AQ 4.3. We tested the parameter settings for both doc2vec and LSI as specified in the assignment and retrieved the following optimal parameters:

- *doc2vec:* {200, 10, 50000}
- *LSI:* {2000}

It should be noted that training LSI with a larger number of topics (5000, 10000) was not feasible on our machines, which is why we only report the performance up to 2000 topics. For both BOW and TFIDF-weighted matrices, LSI models had greater retrieval performance for higher values for the number of topics parameter - this suggests that possibly, LSI performance would have improved still for 5000 or 10000 topics. Unfortunately we were not able to run the word2vec model again with tuned parameters, since we encountered some issues with this model.

	MAP		nDCG	
	All	76-100	all	76-100
doc2vec	0.0075	0.0056	0.307	0.302
LSI-BoW	0.106	0.067	0.489	0.444
LSI-TF-IDF	0.151	0.113	0.540	0.492

Table 4: Retrieval performance in terms of MAP and nDCG for models with tuned parameters.

AQ 4.4. If we assume a significance treshold of $p = 0.05$, all models, except for LSI with TF-IDF, show a significant difference in performance score between using the default and optimized hyper-parameters (Table 5).

Despite LSI-TF-IDF obtaining higher scores after using the optimal settings (MAP: 0.143 to 0.151 and nDCG: 0.526 to 0.540), the model cannot be concluded to perform better as a result of the parameter tuning. That is, p-value, which

is based on the balance between the observed amount of variance between the queries and the number of queries, indicates a too high probability of the observed increase in performance being the result of chance. Nevertheless, as there appears be some positive correlation between the number of topics and the model performance, increasing the topics even further than the used optimal of 2000, might result in a significant difference. However, we do not have the computational means to investigate this.

	MAP		nDCG	
	t	**p**	**t**	**p**
doc2vec	3.09	$2.3 * 10^{-3}$	6.95	$1.1 * 10^{-10}$
LSI-BoW	4.02	$9.4 * 10^{-5}$	6.53	$9.9 * 10^{-10}$
LSI-TF-IDF	0.98	0.34	1.92	0.056

Table 5: T-metric (t) and p-value (p) between default and tuned models.

AQ 4.5. Insufficient time.

AQ 4.6. First of all, we will take a look at the queries that received the highest MAP scores with the word2vec model (Table 7). All query have a very low score, which could be caused by the fact that not all words in the vocabulary have yet been trained properly. For instance, for the worst two queries, the most similar words to the words in those queries do no show any semantic similarity. For isntance, the most similar words to 'machine' are ['1.7', 'regular', 'larger', '64', '4.3', '2', 'livestock', 'metric', 'merchandis']. For the best query on the other hand, the individual word embeddings are more representative, as can be seen for instance from the most similar words to 'bank', which are ['largest', 'japan', 'earlier', 'southeast', 'capit', 'loan', 'declin', '11', 'profit']

Id	Query	MAP
86	Bank Failures	0.0664
56	Prime (Lending) Rate Moves	0.044
164	Generic Drugs - Illegal Activities by Manufacturers	0.0297
66	Natural Language Processing	0.00
171	Use of Mutual Funds in an Individual's Retirement Strategy	0.00
63	Machine Translation	0.00

Table 6: Queries with best and worst MAP score from word2vec model.

Secondly, a query-level analysis will be done for the MAP results achieved with the doc2vec model of which the results can be found in Table 7. The best MAP scores that were retrieved with the doc2vec level are still fairly low, this could indicate that something might have gone with the training of the model. It could also indicate that the query, or document formats were not suitable for this retrieval model. We are not sure why the query "MCI" made for the highest MAP score, but it could be that it is an abbreviation that is seen frequently in news documents. However, it does seem logical that both queries 98 and 97 resulted in a very low MAP score. The model apparently failed in retrieving the meaning of "Fiber Optics" with the AP news dataset as training data, and was not able to retrieve similar documents.

Id	Query	MAP
57	MCI	0.2771
135	Possible Contributions of Gene Mapping to Medicine	0.1325
146	Negotiating an End to the Nicaraguan Civil War	0.07403
60	Merit-Pay vs. Seniority	$5.63 * 10^{-5}$
98	Fiber-Optics Equipment Manufacturers	$7.44 * 10^{-5}$
97	Fiber Optics Applications	$8.99 * 10^{-5}$

Table 7: Queries with best and worst MAP score from doc2vec model.

Overall, it can be seen that for both **LSI-BOW** and **LSI-TFIDF**, the queries with high MAP are largely the same (Table 8, Table 9). Possibly, the reason why these queries perform well is because most of them contain words about rather specific, concrete concepts ('Motherhood', 'Buyouts', 'Greenpeace') and/or are comprised of combinations of words that are frequently used in the same contexts, but only in a limited set of domains/topics ('Surrogate' and 'Motherhood', 'Vietnam' and 'Veterans', 'Leveraged' and 'Buyouts', 'Oil' and 'Spills').

Id	Query	MAP
70	Surrogate Motherhood	0.7731
163	Vietnam Veterans and Agent Orange	0.661
173	Smoking Bans	0.5679
73	Demographic Shifts across National Boundaries	0.0007
89	"Downstream" Investments by OPEC Member States	0.0004
105	"Black Monday"	0.0002

Table 8: Queries with best and worst MAP score from LSI-BOW model.

For the queries with the lowest MAP, we can see a similarly shared set of queries across BOW and TF-IDF models. Possibly, this is because the queries themselves contain relatively

Id	Query	MAP
70	Surrogate Motherhood	0.8062
78	Greenpeace	0.7133
163	Vietnam Veterans and Agent Orange	0.6633
169	Cost of Garbage/Trash Removal	0.0011
73	Demographic Shifts across National Boundaries	0.0008
105	"Black Monday"	0.0004

Table 9: Queries with best and worst MAP score from LSI-TF-IDF model.

less-descriptive words ('Shifts', 'National', 'Boundaries'). For LSI-BOW, query 145 performs poorly because the phrase 'Pro-Israel Lobby' is in quotation marks, which after processing the query, renders the model to perceive the compound adjective 'Pro-Israel' as a single term. It is likely that the corpus does not contain any documents with the exact same phrase in quotation marks, and hence this is essentially an unseen word: observing the contents of the retrieved documents for this query, we find that the returned articles only contain the words 'lobbi' and 'influenc'.

A similar phenomenon occurs with query 169 for LSI-TF-IDF, where the word 'Garbage/Trash' is reduced to 'garbage/thrash' after processing; the model perceives 'Garbage/Trash' as a single word. It is probable that this particular combination of words does not occur across the corpus, which would force the model to rely solely on the less descriptive terms 'Cost' and 'Removal', explaining the low relevance of the retrieved results. Both TF-IDF and BOW LSI models perform poorly on the query 'Black Monday' - this is likely because the model only considers the terms 'black' and 'monday' in isolation, and is not capable of 'understanding' the meaning of the combination of these words (In popular culture, 'Black Monday' refers to the 1987 stockmarket crash). For other queries (112, 91, 89) it is less evident what causes the poor performance. Running query 89 on LSI-BOW yields many documents on OPEC countries, but not necessarily on their downstream investments; possibly the model is not able to capture the contextual meaning of "downstream" in petroleum industries, and its subsequent connection to oil-producing countries.

AQ 4.7. Finally, we looked into the queries with the highest variance in MAP score between the different retrieval models. These queries and the corresponding variance can be seen in Table 10.

It was hard to immediately find a reason for the high variance between the scores retrieved for these queries by the different models. All 5 queries are of a different length, have a different topic, have no punctuation marks in them and seem pretty clear from a linguistic standpoint. For query 69 it could be the case that some models did return more politics and war related documents, since the SALT II treaty falls into this topic, whereas other models did not grasp the underlying meaning of SALT. Both the LSI models scored really well on query 162, while the other models did not achieve a high score which made for a high variance. The semantic meaning of "Automobile Recalls" could have been grasped better in a topic-based model than an embedding, or term-based model. For the third-best query, "Poaching", it seems understandable that different models gave different results. Since the query consists of only one word, it is harder for the retrieval model to deduce what documents could have a similar meaning.

Id	Query	Var
69	Attempts to Revive the SALT II Treaty	0.1296
162	Automobile Recalls	0.1290
77	Poaching	0.1001
153	Insurance Coverage which pays for Long Term Care	0.0913
133	Hubble Space Telescope	0.0878

Table 10: Queries with highest inter-model MAP variance.

Conclusion

In conclusion, we found that it was really hard to give a good analysis on the results, since we have the feeling that errors or bugs might have been present in some of the models. Especially since the results of all self-implemented models were significantly lower than the results of the baseline model (TF-IDF). The results of the LSI models were somewhat okay, but the word2vec and doc2vec results were suspiciously low. We thought that the more "advanced" models, that were note solely term-based, would perform better than the term-based TF-IDF model, but that was definitely not the case. Unfortunately, due to lack of time and resources we were not able to uncover the reasons for these disappointing results.

REFERENCES

[1] A. Karni, M. Crowley, and M. Haberman. 2020. Trump Has a Problem as the Coronavirus Threatens the U.S.: His Credibility. *The New York Times* (26 02 2020). https://www.nytimes.com/2020/02/26/us/politics/trump-coronavirus-credibility.html

[2] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50. http://is.muni.cz/publication/884893/en.