Information Retrieval 1 - Homework 2

Alexandra Lindt
Student number: 12230642
University of Amsterdam
alexandra.lindt@student.uva.nl

David Biertimpel
Student number: 12324418
University of Amsterdam
david.biertimpel@student.uva.nl

Vanessa Botha
Student number: 10754954
University of Amsterdam
vanessa.botha@student.uva.nl

1 THEORY QUESTIONS

TQ 1.1: If we do search on a dataset in the medical domain, our goal is to have a high recall since we don't want to miss out on potentially relevant documents. Therefore, we would include all of the preprocessing steps (tokenization, stemming, stopword removal) as all of them enhance recall. Note that we might need domain-specific stemming to not prune medical terms in wrong ways.

TQ2.1: On the one hand the idea of dense word embeddings is great as we can efficiently find abstract representations of words in a vector space that is specifically optimized to capture word similarities. However, when it comes to retrieval we want to rank and retrieve documents. Relying solely on such word embeddings may be problematic as we have to aggregate queries and documents somehow into a single representation. A naive way of doing so would be the mean aggregation, however, this would certainly discard useful information in the document and query respectively making retrieval imprecise. Another problem could be that we don't have any control over the semantic properties the embedding captures, as it relies heavily on the respective corpus. One set of training documents may not generalize well to another document set even if they share the same topic domain.

TQ2.2: The most prominent difference is that TF-IDF and BM25 are both count based approaches, whereas Word2vec and Doc2vec embed dense representations of word/documents in a high dimensional vector space. This may capture higher order relationships that count based methods are not able to exploit. While TF-IDF or BM25 rank documents based on their count based scores, Word2vec or Doc2vec use similartiy measure such as the cosine similarity to retrieve the most similar words/documents. With well trained vector embeddings this may lead to more stable and semantically more meaningful results.

TQ 3.1: Both methods are able to capture conceptual similarity between documents (topics) and reduce the dimensionality of the term-document matrix. LSI does this by SVD to identify features that capture most of the variance, whereas LDA is a generative probabilistic model of a corpus that assumes that each document is a mixture of various topics and each topic is a mixture of various words. Whereas LDA is able to capture both synonymy and polysemy, LSI is only able to capture synonymy [1]. While LDA seems to model the topics better, we consider LSI to be more suitable for IR since it is much faster during evaluation time.

2 WORD2VEC / DOC2VEC MODELS

2.1 Implementation details

During the Word2vec [2] implementation we faced numerous design choices regarding how much data processing we want to perform during model training. We opted for keeping the training loop clean and fast and thus decided to do large parts of the data processing before training. This way we are able to use the Pytorch Dataset and Dataloader structures which allow us to efficiently sample mini-batches over the whole saved data without running into memory errors. This comes at the expense of processing the data beforehand (removing infrequent words, building up Word2vec dataset). Although, we put a lot of emphasis in implementing this prepossessing in an efficient manner, the processed data becomes very memory heavy with an increasing amount of documents. For this reason we only consider the first $100,000$ of the total $\sim 160,000$ documents. Further, we filter each word with a token frequency ≤ 50. Nevertheless, our preprocessing implementation runs comparably fast and takes around 10 minutes on an *i5-8250U* (given 16 GB of RAM).
For the Doc2vec model [3] we do not filter out infrequent words in the default setting because it was not mentioned to be a necessity in the corresponding paper. However, note that we prune the vocabulary size during the grid search. We train the Doc2vec model for only 2 epochs, since we empirically observed that it gets worse when it is trained too long. The default model parameters (taken from the paper) are a window size of 8 and a vector size of 400. We found that in order to make the model perform reasonably well, it is neccessary to set the parameter dm = 0 such that PV-DBOW is used as the model's training algorithm instead of PV-DM. We want to mention that even with the usage of `delete_temporary_training_data()` (as suggested in a canvas discussion) we ran into memory errors when evaluating our Doc2vec model on a machine with 8GB of RAM.

2.2 Analysis Questions

AQ2.1: In order to find the word similarities shown in Table 1, we train our *Word2vec* model over 500 epochs with a word-window size of 10, embedding size of 300, with batch-size 1024 and a learning rate of 0.003. We choose the words *politics*, *money*, *pay*, *president* and *accident*, as they all seem to be classic words appearing in news articles. As a sanity check we observe that for each of the words the most similar word is always itself. Note that we only can display the stemmed version of each words due to the standard preprocessing.
While the predicted words in Table 1 are not obvious in every case, we observe a lot of evidently similar words or words that would occur in the same context. *Politics* is linked to similar words such as *subject*, *argument*, *critic* or *intent*. For the word *pay* nearly all connections make sense and can be seen as similar. Also the words linked to *president* are highly similar (*ambassador*, *meet*, *men*). Conversely, when looking at the words *money* and *accident* we find natural connections only in a few cases. For example top matching words for *money* like *wo*, *also* and *wife* seem rather unrelated. However, in the big picture the predicted similarities clearly show that the Word2vec model was able to learn meaningful word embeddings.

politics	money	pay	president	accident
healthi	wo	compar	ambassador	caught
truce	also	would	cabrera	737
subject	abl	regret	complain	attend
claim	wife	permit	opposit	commonli
sheriff	abil	less	senior	sawyer
argument	plant	take	meet	injur
critic	renew	dozen	examin	1973
william	lose	1	dozen	191
weather	short	hundr	collect	swear
intent	offer	much	men	mammoth

Table 1: Top 10 similar words according to the cosine similarity of their trained word embedding.

AQ2.2: We used a Spaghetti Carbonara recipe as our input document for the trained Doc2vec model. In the ten documents with the highest ranking we had three interviews with cookbook authors, three other recipes, three documents on clothing/fashion and also one about travelling south America. The documents that were recipes or interviews with cookbook authors were the ones with the highest ranking. This makes a lot of sense, since they all contain multiple cooking related words, just like our input document does. After closer inspection of the more fashion-related documents and the document on travelling, we assume that they were connected to our recipe because they contained words that describe measuring (i.e. 'large'/ 'medium' / 'small'), words connected to nature (i.e. 'fruit' / 'flower' /'garden') and even some food words (i.e. 'salad'/ 'fussili'). All of those words a likely to appear in similar contexts as the food related words in our query document.

3 LSI AND LDA

3.1 Implementation details

For training of the LSI and LDA [4] model we used the default parameter settings as provided in the Gensim modules and set num_topics to 500. Our TA suggested to remove the tokens that occur less than 150 times in the corpus. However, by doing so we were not able to get reasonable results. We found empirically that filtering tokens occurring in less than 25 documents and more than 50% documents worked best. After training the LDA model using the *TF-IDF* corpus as suggested in the assignment, we found it nonfeasible to run an evaluation on the model, even after upgrading our RAM to 25GB in a google colaboratory and implementing the evaluation process in a more computationally efficient way. Therefore, we decided to train the LDA model using the BoW corpus as suggested by one of the TAs in a canvas discussion.

3.2 Analysis Questions

AQ3.1: When looking at the five top significant topics of our LSI model using BoW we can observe topics related to politics and finance. To exemplify, the first topics contain tokens as 'state', 'govern', 'presid', 'nation', whereas the following topic contains words as 'percent', 'price', 'rate', 'cent', 'market' and 'increase'. We observe that tokens such as 'million', 'billion', 'say', 'people' are shared across the top 5 topics. Those are words that occur frequently in newspapers in different contexts, which causes them to have co-occurrences with words from different topics.
By using a BoW term-document matrix, we give equal weight to co-occurrences with such frequent words and words that are actually distinct for a certain subject. By using a TFIDF term-document matrix we tackle this since we take into account the occurrence frequencies of a tokens within the corpus. When looking at the top 5 topics of LSI using TFIDF, we observe less of those generic words within the topics. More informative words are included which makes that the topics become a bit more specific. To further illustrate, in the third topic we observe tokens as 'soybean', 'wheat', 'corn', 'oil', 'grain'.
With the LDA model using BoW the distinction between the subjects of the topics is more clear. E.g. in the third topic we observe tokens as women, abort, men, life, sex, pregnanc and in the fifth topic space, launch, engin, mission, shuttl, satellit. That LDA is able to find the best topics is not surprising in the sense that LDA specifically models the topics, whereas in the LSI model the topics rather arise as a "side product" of dimensionality reduction.

4 RETRIEVAL AND EVALUATION

4.1 Implementation details

For the retrieval and evaluation we closely follow the provided *TF-IDF* structure and replace `tfidf_search.search()` with the our respective retrieval function. For the Word2Vec retrieval we implement `mean`, `max` and `min` aggregation but use `mean` aggregation throughout all experiments to make the results comparable with the other models.

4.2 Analysis Questions

AQ4.1: The following Tables 2 and 3 display the retrieval performance on *all queries* and *queries 76-100*.

Method	MAP	nDCG
TF-IDF	**0.2161**	**0.5800**
Word2Vec	0.0054	0.3060
Doc2vec	0.0185	0.3564
LSI-BoW	0.0885	0.4589
LSI-TF-IDF	0.1644	0.5490
LDA-BoW	0.0022	0.2798

Table 2: *all queries*

We clearly see that *TF-IDF* significantly outperforms all other approaches. Following this *Doc2vec* and *LSI-TF-IDF* have a comparable performance, with Doc2vec having slight advantages.

AQ4.2: Given the results of the previous exercise, where we observe considerable differences in performance between models, it is not surprising that we see highly significant results in Table 4.
Assuming $\alpha = 0.1$, the only model comparisons where the H_0 (=both models follow the same distribution) is not rejected for both metrics are the pairs *TF-IDF – LSI-TF-IDF* and *Word2vec - LDA-BoW*. In the first case the p-value for the nDCG metric is not significant

Method	MAP	nDCG
TF-IDF	**0.1826**	**0.5449**
Word2Vec	0.0022	0.2924
Doc2vec	0.0063	0.3271
LSI-BoW	0.0609	0.4248
LSI-TF-IDF	0.1173	0.4984
LDA-BoW	0.0019	0.2864

Table 3: *queries 76-100*

	MAP (p-value)	nDCG (p-value)
TF-IDF - Word2Vec	$2.0710 \cdot 10^{-24}$	$3.3450 \cdot 10^{-36}$
TF-IDF - Doc2vec	$1.8677 \cdot 10^{-22}$	$5.3267 \cdot 10^{-27}$
TF-IDF - LSI-BoW	$1.1043 \cdot 10^{-11}$	$2.0665 \cdot 10^{-10}$
TF-IDF - LSI-TF-IDF	0.0015	0.0501
TF-IDF - LDA-BoW	$1.8386 \cdot 10^{-24}$	$1.3656 \cdot 10^{-38}$
Word2vec - Doc2vec	$2.7489 \cdot 10^{-05}$	$3.7990 \cdot 10^{-12}$
Word2vec - LSI-BoW	$4.5992 \cdot 10^{-12}$	$7.8960 \cdot 10^{-28}$
Word2vec - LSI-TF-IDF	$4.5642 \cdot 10^{-20}$	$2.1945 \cdot 10^{-41}$
Word2vec - LDA-BoW	0.01187	$4.2425 \cdot 10^{-20}$
Doc2vec - LSI-BoW	$4.6403 \cdot 10^{-10}$	$8.8101 \cdot 10^{-17}$
Doc2vec - LSI-TF-IDF	$8.8982 \cdot 10^{-19}$	$3.6145 \cdot 10^{-33}$
Doc2vec - LDA-BoW	$2.6647 \cdot 10^{-07}$	$1.6225 \cdot 10^{-19}$
LSI-BoW - LSI-TF-IDF	$2.1673 \cdot 10^{-10}$	$1.2194 \cdot 10^{-15}$
LSI-BoW - LDA-BoW	$3.8074 \cdot 10^{-12}$	$2.2188 \cdot 10^{-29}$
LSI-TF-IDF - LDA-BoW	$3.2941 \cdot 10^{-20}$	$9.75759 \cdot 10^{-43}$

Table 4: Significance values across implemented models.

($\alpha/2 \leq 0.0501$), in the latter case the same is true for the MAP metric ($\alpha/2 \leq 0.01187$). Consequently, for the vast majority of methods it matters which method to use for retrieval as the observed difference in performance did most likely not occur by chance.

AQ4.3: Note that we consider the test queries to be all queries but queries $76-100$. The following Tables 5 and 6 display the retrieval performance of the models after parameter optimization on *all queries* and *test queries*.

Method	MAP	nDCG
Word2vec	0.0086	0.3099
Doc2vec	0.0202	0.3590
LSI-BoW	0.1126	0.4997
LSI-TF-IDF	0.1785	0.5655

Table 5: *all queries on tuned models*

AQ4.4: In contrast to the t-tests before we do not see as many significant results. This makes sense as the models after parameter tuning only marginally outperform its default parameters.

AQ4.5: The Doc2vec model's parameters are window size, vocabulary size and vector size. For the parameter tuning, we trained and evaluated a total number of 80 Doc2vec models with the possible parameter values given in the assignment. We found that when

Method	MAP	nDCG
Word2vec	0.0090	0.3121
Doc2vec	0.0132	0.3478
LSI-BoW	0.1208	0.5092
LSI-TF-IDF	0.1873	0.5761

Table 6: *test queries on tuned models*

	MAP (p-value)	nDCG (p-value)
Word2vec	0.0246	0.3191
Doc2vec	0.0331	0.0422
LSI-BoW	$9.5897 \cdot 10^{-05}$	$3.5300 \cdot 10^{-10}$
LSI-TF-IDF	0.1640	0.0449

Table 7: Significance values between default and tuned parameters.

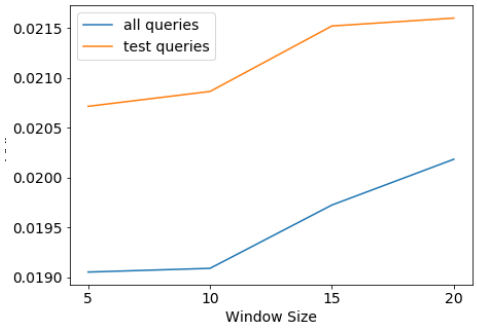
evaluated on the validation set (i.e. queries $76-100$), the Doc2vec model performs best with a window size of 20, vocabulary size of $25k$ and vector size of 200. We made three plots to investigate the retrieval performance in terms of MAP with respect to the value of the parameters. In each plot, we set two of the parameters to their value in the best performing model and investigate how the performance changes when we change the third parameter. The results are depicted in figure 1.

We can see in the plots that the window size is positively correlated to the performance of Doc2vec: The bigger the window the higher the MAP performance. This means Doc2vec profits from considering a bigger context in our case. The performance for different vocabulary size is not as conclusive. Nevertheless, we can see that it is in general lower for the bigger vocabulary size $50k$, $100k$ and $200k$ than for the smaller vocabulary sizes $10k$ and $25k$. This suggests that it is indeed beneficial to prune out infrequent and frequent words. The performance is very similar for all vector sizes. However, the smallest vector size of 200 clearly gives the best performance and the biggest vector size of 500 gives the worst. Apparently, it helped in our task to make the model use a smaller embedding size and therefore force it to represent words and documents in a smaller vector (probably results in less sparse document vector).

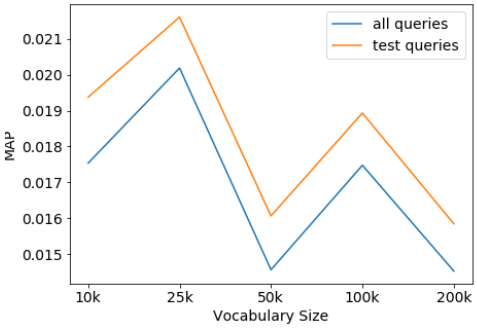
Note that we consider the validation queries to be queries $76-100$ and the test queries to be all queries but the validation queries. Overall, we can see from the plots that all Doc2Vec models perform better on only the test queries than on all queries. This is very reasonable, considering that we already saw in earlier questions that the Doc2Vec model performs clearly worse on the validation queries than on all queries (and the validation queries are left out in the test queries). It is interesting to observe that for all three parameters, the performance of the Doc2vec model on the test queries is proportional to its performance on all queries. In comparison, the LSI models both only show a small performance difference between test queries and all queries, which suggests that the models performed similarly on the validation queries than on other queries.

For the LSI models, we tuned the number of topics. We had to leave the topic sizes 5000 and 10000 out. They were simply not feasible to train or evaluate, even after upgrading to a 25GB RAM google

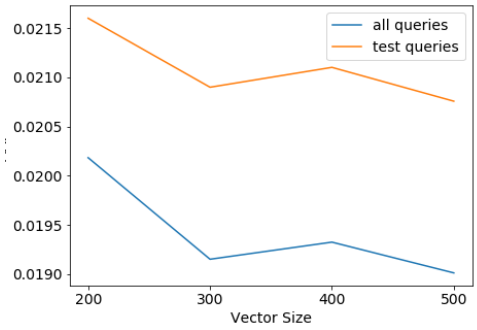
colaboratory. One TA told us that it was fine to leave them out. As apparent from figures 2 and 3, both LSI models perform the better the more topics they model which is reasonable. As expected, the LSI trained with the BoW corpus performed consistently worse than the one trained with the TF-IDF corpus.



(a) Doc2vec performance over window size



(b) Doc2vec performance over vocabulary size.



(c) Doc2vec performance over vector size

Figure 1: Doc2vec performance for different parameter settings. Every subplot shows the change in performance when a certain parameter changes while the other two parameters are fixed to their values in the best performing model.

AQ4.6: Our analysis first of all reveals that queries with a relatively low number of relevant documents tend to result into lower MAP scores, whereas queries with a higher number of relevant documents tend to result in higher MAP scores. For example queries outside the scope of the data (in our case News articles) such as 'Natural Language Processing' or 'Funding Biotechnology' perform bad across all models.

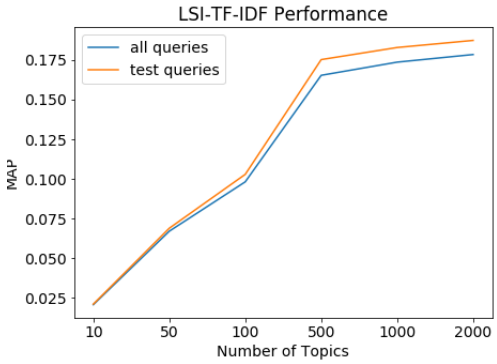


Figure 2: LSI-TF-IDF performance over number of topics.

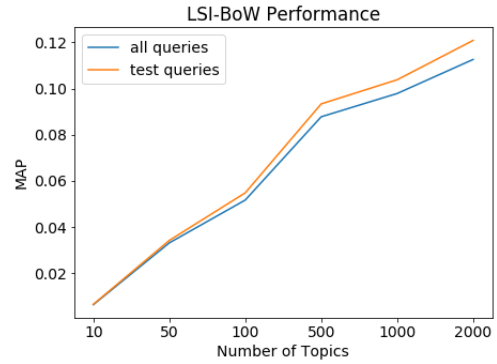


Figure 3: LSI-BoW performance over number of topics.

Now, in the case of Word2vec it becomes obvious that the model performs better on longer queries, that consist of multiple nouns related to the same topic. The reason may be that the aggregated average will have less variance if the words follow a consistent topic and thus converge better to a single 'topic mode'. Longer queries help in this case to make the aggregation more precise. The model seems to perform much worse if the queries are shorter and more precise which may be in conflict to the rough similarity the Word2vec predicts. However, it should be generally noted that the Word2vec performance is not great, therefore we expect the results to be of high variance and only conditionally interpretable.

Doc2vec shows a behavior similar to Word2vec as it also performs well on longer queries with lots of similar nouns. Interestingly it seems to struggle with queries only consisting of abbreviations but this may also be a behavior specific to the data.

In contrast to Word2vec and Doc2vec the LSI BOW seems to perform better on shorter, more precise queries. The reason for this could be that LSI extracts topics (the vector features) based on co-occurrences of words. Queries that contain combinations of words

that are informative about a certain subject can therefore be helpful to find the documents of the same topic. Having longer queries might add "noise" in that sense. The same can be observed for LSI-tfidf. We can also observe similar behaviour for the LDA, but the results here are more noisy, since in our case the LDA generally does not perform well, which limits the significance of the results.

AQ4.7: Table 8 shows the top 5 queries that have the highest variance in terms of MAP between the different retrieval models. On most of these queries LSI based models perform better than Word2vec and Doc2vec. This aligns with our previous observation that distribution based models perform better on more concise queries. For example, if we look at query 70, both LSI BoW and LSI TF-IDF scored best on this query in terms of MAP, while Doc2vec had a low score on this query. This may explain the high variance of such requests, because while LSI BoW and LSI TF-IDF perform well in these examples, the other models may fail entirely creating a big performance gap. In general, it should be noted that the overall performance across all models has a high variance, which may bias the variance estimates of some queries.

	Query id	Query	Variance
1	70	Surrogate Motherhood	0.1598
2	78	Greenpeace	0.0951
3	163	Vietnam Veterans and Agent Orange	0.0891
4	173	Smoking Bans	0.0867
5	162	Acid Rain	0.0696

Table 8: Top 5 queries that have the highest variance in terms of MAP.

REFERENCES

[1] Tru Cao, Ee-Peng Lim, Zhi-Hua Zhou, Tu-Bao Ho, David Cheung, and Hiroshi Motoda. *Advances in Knowledge Discovery and Data Mining: 19th Pacific-Asia Conference, PAKDD 2015, Ho Chi Minh City, Vietnam, May 19-22, 2015, Proceedings*, volume 9077. Springer, 2015.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[3] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1188–II–1196. JMLR.org, 2014.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, March 2003.