Assignment 2: Retrieval Models

Information Retrieval

Artifical Intelligence - University of Amsteram

Fije van Overeem
10373535
fije@hotmail.com

Diede Rusticus
10909486
10909486@student.uva.nl

ABSTRACT

This is an assignment report for the course Information Retrieval at the University of Amsterdam. We analysed 10 retrieval models through experiments on query and document data sets. Finally, we compared lexical models to Latent Semantic Models (LSMs) through four performance metrics. Statistical tests show that Dirichlet Prior has the highest percentage of wins in its model comparisons and is concluded to be the best performing model. Furthermore, we conclude that the LSMs perform worse than the lexical models.

1. INTRODUCTION

We are provided with a list of 150 queries and an index of over 160,000 documents. Validation and test labels provide the relevance information per query-document pair. Our task is to predict these relevances through lexical Information Retrieval (IR) methods and Latent Semantic Models (LSM). For each implemented model, an evaluation framework called Trec Eval which is used to extract the Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), Recall and Precision measures per query.

2. EXPERIMENTS

The approach is described for two tasks of the assignment. Task 1 implements and compares lexical IR methods and Task 2 experiments with Latent Semantic Models (LSM). All methods described are evaluated for their performance using TREC Eval.

Apart from that, in Task 2 LSM's are used to re-rank one of the lexical IR methods top 1,000 document list for each query. It simulates how large ranking events would occur: first rank documents with a computational 'cheap' method, then improve results with a more computational complex method. We expect to learn which LSM does this best, and why.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2.1 Task 1

The following lexical methods for IR are implemented using the Pyndri framework: TF-IDF, BM-25, Jelinek-Mercer, Dirichlet Prior, Absolute Discouting and Positional Language Models. The hyperparameters for the smoothing algorithms are optimised on a validation set. All models are then evaluated again on the test set and compared.

Pipeline

An inverted index is made of all unique query tokens. The postings are all documents where the query token occurs at least once. An extra value is added to this index which represents the collection frequency. This inverted index will help us optimise computational cost as the relevant documents of a query can easily be retrieved.

Next, all stop words are deleted from the collection. Pyndri facilitates this by giving all stop words the same token id of 0. Next to this collection, a copy of the collection is made, with documents only containing its unique words (or vocabulary).

A dictionary with query ids and their query strings is created and another for the query ids for the validation and test set, accompanied by their relevant documents are retrieved.

Only the relevant documents of a query are scored. However, Trec Eval requires at least 1,000 documents per query. So for queries with less than 1,000 relevant documents, random documents are picked from the collection for scoring. For TF-IDF and BM-25 we set the scores of these random documents to 0 already because they do not use smoothing to avoid zero probabilities.

Jelinek-Mercer, Dirichlet Prior and Absolute Discouting contain hyperparameters that need to be optimised. Jelinek-Mercer is tested with $\lambda = 0.1, 0.2, ..., 0.9$. Dirichlet Prior with $\mu = 500, 1000, 1500, 2000$. And Absolute Discouning with $\delta = 0.1, 0.2, ..., 0.9$. Each parameter setting is evaluated on the validation set and the performance measures of Trec Eval tell us the best performing hyperparameter setting. With these outcomes, the three methods are again evaluated on the test set.

Positional Language Models can use different smoothing and kernel functions. A Gaussian, Triangle, Cosine, Circle and Passage kernel are implemented. However, only the Gaussian kernel is used for evaluation due to computational limitations. PLM can use either Dirichlet Prior of Jelinek-Mercer as smoothing method to avoid zero probabilities. Only Dirichlet Prior with the best NDCG scoring parameter setting is used for the PLM.

2.2 Task 2

It was asked to train four different LSMs: Word2Vec, LSI, LDA and Doc2Vec. LSMs carry the property of being able to represent the meaning of words. This makes it possible to overcome some flaws of querying only with syntax, where one matches the spelling of a word with the words in a document instead of their meanings.

2.2.1 Word2Vec and Doc2Vec

In Word2Vec and Doc2Vec, words and documents (or paragraphs) respectively, are represented by vectors. Each value in the vector stands for a direction in multi-dimensional space which represents a piece of the semantics of the words. The more values, the more the vector can embed meaning. Unfortunately this also means extra training time, so you have a model quality/training time trade-off.
It means we had to make a choice for the latent variables, specified later on. Other variables that directly add to the quality of the models is the amount of documents the model is trained on, the amount of 'epochs' of backpropagation, and the 'window'-size: the amount of neighbour words word2vec takes into account when training the model with skip-gram or CBOW.

Query and document representation For the query and document representation we took an average of all the (relevant) words in the query/document. (Except for the document representation in doc2vec because here a document is already represented as one vector). This is the simplest form of handling multiple vectors to have a vector representation for multiple words.

- **Word2Vec** We used the following settings to train our model: 50.000 documents to train on (almost one third of the set of documents), 64 latent variables, a window size of 5 (the maximum amount of words between the current word and predicted word during forward propagation within skip gram or CBOW). We used skip-gram and negative sampling. Our model could've been of better quality if we had trained on the maximum amount of document (+- 160.000). Also, the vector space dimension could have been larger, but this only makes sense if you indeed use a larger training set of documents as well. More importantly, the more epochs you do backpropagation, the better your model is trained. We now used 5 epochs as it resulted in a feasible training time, if we had to continue improving our training models we would start with experimenting with a bigger training size and the amount of epochs. Lastly, we used skip-gram instead of CBOW, by default.
- **Doc2Vec** For Doc2Vec we did train on the whole document size, but we only trained with 32 latent variables and 3 epochs. This amount of epochs is probably on the very low side. One needs at least 15 epochs to get a good model, we assume. Time/model quality trade-off made us decide not to do so. Skip-gram/CBOW is documented here as 'distributed memory' (dm) and 'distributed bag of words'(DBOW). The documentation informed us that 'dm' outperforms dbow noticably, which is why we used 'dm'.

2.2.2 LSI

LSI compares how often words appear together in the same document and compares how often those occurrences happen in all of the documents that the dictionary has in its index. Both with LSI and LDA we make a bag-of-words of the documents, so the order of the words in the document get lost. It ignores the ordering of documents as well (which is important when being unbiased when making a re-rank of tf-idf). LSI can avoid the problems with synonymy, where multiple words have the same meaning (e.g. if you query 'great giant', you don't want to ignore documents about 'big giant', because great and big are, in some contexts, synonyms). This will affect the recall. LSI also avoids problems with polysemy, where a word has multiple meanings. This affects the precision.

2.2.3 LDA

For the query terms, LDA only gives back those variables with a probabilities, but to calculate cosine similarity between two vectors one needs to have vectors with the same length. This is why we put in 0-probabilities for the topics tht don't occur in the text, which could use some smoothing. For LDA we've chosen to train on all documents and use max 200 topics.

3. RESULTS

The results are shown of the hyperparameter optimisation of Task 1 and the model comparisons of the 10 models.

3.1 Hyperparameter optimisation

Figure 1, 2, and 3 show the results of different hyperparameter settings of the smoothing methods Jelinek-Mercer, Dirichlet Prior and Absolute Discouting.

For Jelinek-Mercer, it shows that $\lambda = 0.2$ has the highest NDCG measure. The winning values of other measures are as follows: Recall: 0.2, Precision: 0.9 and MAP: 0.2. With the exception of the winner for Precision, we chose the winning parameter setting to be $\lambda = 0.2$ and is used for the test set.

Dirichlet with $\mu = 1000$ shows the highest NDCG measure and is therefore used for the PLM. However, comparing also with the other performance metrics: Recall: 2000, Precision: 1500 and MAP: 2000, $\mu = 2000$ seems the optimal setting.

Absolute Discouting shows that $\delta = 0.8$ has the highest score. Again except for Precision, this is in coherence with the other metrics: Recall: 0.8, Precision: 0.1 and MAP: 0.8. $\delta = 0.8$ is therefore chosen for the test set.

3.2 Model Comparison

Figure 4 shows the results of Trec Evals evaluation metrics: Recall, NDCG, MAP and Precision. Every combination of models is checked for a significant difference in mean through a Two-tailed T-test with $\alpha = 0.05$. We need to avoid false rejections of the null hypothesis caused by the multiple comparison problem. We do this with the Ryan-Einot-Gabriel-Welch [2] which states that every independent comparison has an unadjusted P-value p. The adjusted P-value $\hat{p}$ is calculated to test with the significance level $\alpha = 0.05$. The equation is as follows:

$$\hat{p} = 1 - \prod_{i}^{P} (1 - p_i)$$

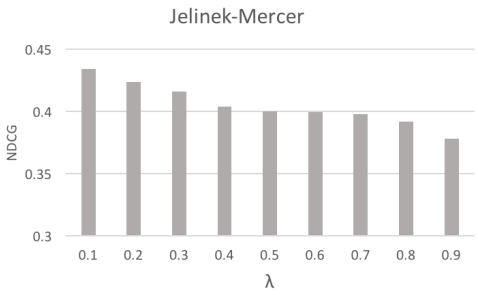

Figure 1: $\lambda = 0.1$ has the highest NDCG measure. However, comparing also with the other performance metrics, $\lambda = 0.2$ seems the optimal setting.

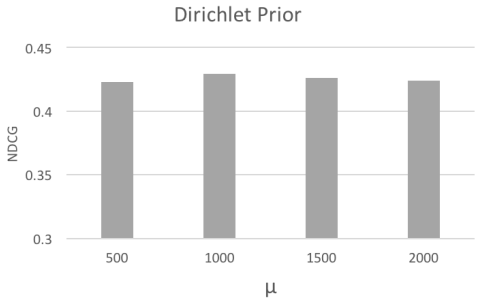

Figure 2: $\mu = 1000$ has the highest NDCG measure.

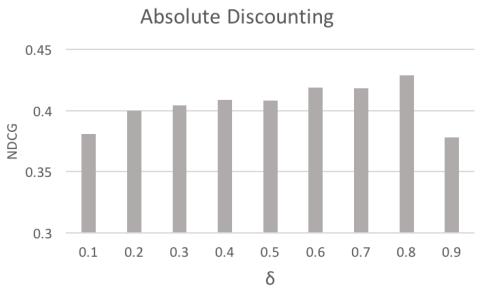

Figure 3: $\delta = 0.8$ has the highest NDCG measure.

Table 1 shows the resulting $\hat{p}$-values. For all found significant differences, the wins per metric are counted for each model and this resulted in the following score: Dirichlet: 16. TF-IDF, BM-25, PLM, A.D.: 12. Jelinek: 8. W2V, LSI, D2V: 0.

4. ANALYSIS

The results are analysed further for both tasks. And an overall analysis is also given.

4.1 Task 1

The significance tests shown in Table 1 does not show that one of the lexical IR methods outperforms the other. Recall has overall the highest scores and precision the lowest. The precision could be low on these methods because documents are only retrieved when matching words occur in them. There could be many more relevant documents without containing any of the query tokens. On the other hand, because these methods are good in retrieving the documents containing the word queries, it is likely that they are also relevant and on-topic.

4.2 Task 2

We expected that the LSM's would improve the results of the retrieval methods of task 1 - it did not. Even more, recall should actually be the same in tf-idf and all the other models because we calculate precision over the exact same 1000 documents. Something probably went wrong with assigning the right documents during evaluation. All evaluation methods perform the same with each LSM. It seems like not much has improved. We would expect doc2vec to create better results when the model would've been trained better. Still, it should first be figured out what went wrong with the evaluation or document assignment while calculating cosine similarity, to make sure the results make sence.

LSI has two problems which LDA overcomes. The first problem is that LSI does not handle unseen documents well, because it only assumes topics that occur in the documents that the model is trained on. The second problem is that the number of parameters which must be estimated grows linearly with the number of training documents[1]. We expect this not to matter that much in these results because we only re-rank on relatively short, already-seen documents. In fact, the LSI outperforms LSA, as seen in the results.

5. CONCLUSIONS

The results and the analysis show that there is not one retrieval model that outperforms all other models. The lexical methods all show similar performances and this also counts for the LSMs. For LSI, the T-test shows that we can only conclude that it performs worse than Dirichlet Prior. However, overall between the two types of models, we see that the lexical methods outperform the LSMs on almost all performance metrics. Part of this could be due to some implementation errors. When we point out one winning retrieval metric from our experiments, Dirichlet Prior performs the best with %44 wins of its model comparisons, followed by BM-25, Absolute Discounting and PLM with %33 wins.

Figure 4: Dirichlet has the most wins per metric and significance testing also shows that it has the highest performance

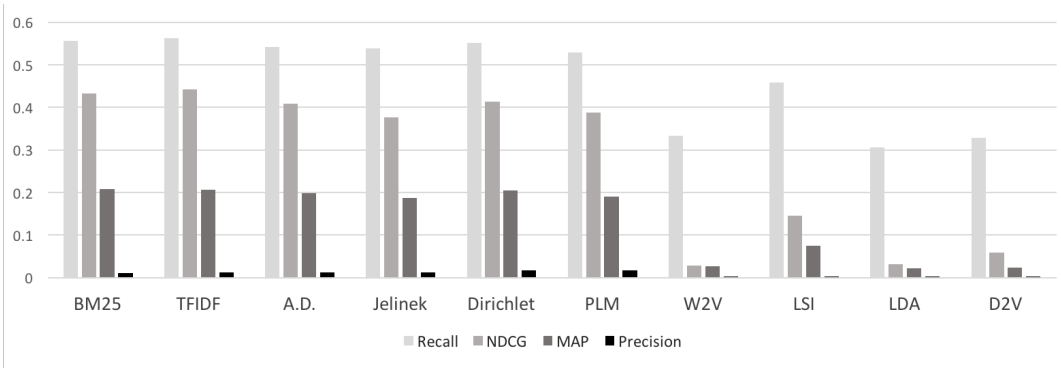

Table 1: Two-tailed T-test $\hat{p}$-values. Significant difference in mean when $\hat{p}$-value < 0.05.

	TF-IDF	BM25	Jelinek	Dirichlet	Abs. Dis.	PLM	W2V	LSI	LDA
BM25	1.0								
Jelinek	0.987	0.98							
Dirichlet	0.997	0.998	0.964						
Abs. Dis.	0.999	0.993	0.999	0.998					
PLM	0.907	0.898	0.999	0.999	0.992				
W2V	**0.0479**	**0.0353**	0.05	**0.0164**	**0.0487**	**0.0136**			
LSI	0.074	0.067	0.11	**0.0391**	0.095	0.084	0.644		
LDA	**0.0454**	**0.0329**	**0.0474**	**0.0156**	**0.0463**	**0.0127**	0.994	0.59	
D2V	**0.0428**	**0.0303**	**0.0445**	**0.0147**	**0.0437**	**0.0119**	0.987	0.529	0.989

6. REFERENCES

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[2] S. P. Wright. Adjusted p-values for simultaneous inference. *Biometrics*, pages 1005–1013, 1992.