Learning to rank medical publications

Mick van Hulst
Radboud University
Nijmegen, Gelderland, Netherlands
mick.vanhulst@gmail.com

Jelle Piepenbrock
Radboud University
Nijmegen, Gelderland, Netherlands
jellepiepenbrock@gmail.com

Lena Shakurova
Radboud University
Nijmegen, Gelderland, Netherlands
lenashakurova.work@gmail.com

ABSTRACT

Improving the retrieval of relevant medical information from large health databases could contribute to a better health care system. There is a long history of systems produced to obtain relevance rankings for documents (Learning To Rank). In this project we wanted to experiment with various features and algorithms that are commonly used in Information Retrieval systems and see which ones can be applied to the medical field. We used the dataset provided during the TREC-9 conference, namely the OHSUMED dataset. Pointwise Logistic Regression and Pairwise RankSVM were used to rerank a set of documents for queries. We experiment with various combinations of features to find the set which performs best on our training data and find that Partial Term Frequencies and the cosine distance between TF-IDF of the query and the document to be relevant features. With the final set of features we improve the Microsoft baseline, where our pointwise method outperforms our pairwise ranking method.

KEYWORDS

Information Retrieval, Pointwise Logistic Regression, Pairwise Rank SVM

1 INTRODUCTION

When using an information retrieval system, it can be computationally infeasible to rank the entire database of documents. To overcome this problem, the ranking process can be divided into two stages, where the first stage consists of retrieving a top-k list of relevant documents and the second stage consists of reranking those top-k documents.

This research focuses on an approach for the reranking of top-k documents in the medical domain. Specifically, we focus on a task called 'Learning to Rank', which entails the usage of machine learning algorithms for the purpose of ranking documents. The process of ranking is based on a previously generated ranking, which is used as the training data. The objective of the task is to optimize the ranking such that relevant documents are ranked higher than less relevant documents.

The training data that is used in this project originates from the TREC-9 conference, which was held in 2000. During the conference, one of the seven tracks included a Filtering Track[1]. The goal of the Filtering Track was to retrieve the documents from a document stream that matched the user's query. For this task, the OHSUMED dataset was used. This dataset consists of a set of queries, a set of medical documents and the corresponding relevance scores for the query-document pairs.

[1]https://trec.nist.gov/pubs/trec9/papers/overview_9.pdf

Our objective was to use the data provided at the TREC-9 competition to compare the results of two learning-to-rank approaches, namely pointwise logistic regression and pairwise ranking (RankSVM). As a baseline, we used one of the submissions made during TREC-9 competition. Furthermore, we applied the aforementioned algorithms to test various subsets of features so that we can conclude which features work best for the given domain, models and corresponding dataset.

Our project is aiming to answer the following research questions:

(1) Which features contribute the most to the reranking performance of RankSVM and Pointwise LR?
(2) How does pairwise ranking (RankSVM) compare to the pointwise logistic regression ranking method?

The following sections cover related work, the methods that were used, our data source, the flow of the experiments, results of the experiments and finally the discussion of the results.

2 RELATED WORK

In Information Retrieval literature, there were different ranking approaches proposed. Liu [6] suggests the following classification of ranking approaches: the pointwise approach, the pairwise approach, and the listwise approach. In this paper we will consider pointwise and pairwise approaches.

For **pointwise ranking**, the assumption is that each query-document pair in the training data is labeled with a relevance score (relevant/non-relevant). The ranking task can then be tackled as a classification problem. The objective of the model is to predict the relevance score given a single query-document pair, represented by features. Documents are then ranked according to the probability estimates of predicted relevance scores. Cooper et. al [1] proposes a staged logistic regression approach to predict relevance scores and mention the computational simplicity and effective retrieval of the model. We will refer to this model as Pointwise Logistic Regression (Pointwise LR). A disadvantage of this approach is that by maximizing the prediction of the relevance score, the model does not necessarily optimize the ranking of the documents. Therefore, while the relevance predictions might not be wrong, the ranking might still be suboptimal.

This problem is tackled by the **pairwise approach** where the ranking task is seen as a classification problem. Given a pair of two documents, both represented as feature vectors, a binary classifier is trained to predict whether documents are in the correct order. The model learns a function that maximizes the expected similarity between two orderings (the ground truth and the learned ordering). After that, documents are ranked in the order of decreasing score [3]. Therefore, as opposed to the pointwise approach, this method focuses solely on learning if the ordering of the pair of documents is correct or not. Pairwise learning-to-rank algorithms using SVM, called RankSVM, was suggested by [3].

The aforementioned approaches are not the state-of-the-art approaches and there are newer methods nowadays that outperform RankSVM and Pointwise LR. An example of this is a method that combines the LambdaRank method and boosting, LambdaSMART [9]. We decided to limit ourselves to the discussion of RankSVM and Pointwise LR because they are relatively simple and easy to implement.

3 METHODS

This section introduces the methods that were implemented for the reranking of the aforementioned dataset, namely RankSVM (i.e. pairwise ranking) and pointwise logistic regression. Both of the methods apply a two-step process for reranking, where the first step consists of classification and the second step consists of the reranking itself. The difference between both methods is explained in the remainder of this section.

3.1 Pointwise Logistic Regression Rerank

Pointwise Logistic Regression is a pointwise ranking method that classifies a document as either relevant or non-relevant for a given query with a certain probability. We first generate features separately for each pair of a document and a query (d, q). The resulting features serve as an input to the algorithm. At the classification stage, the model is trained to directly predict the relevance score based on the document-query features. Reranking of the documents consists of ordering the set of documents for a particular query based on the probability estimates calculated by the model.

3.2 Pairwise RankSVM

RankSVM is a pairwise ranking method that reduces the problem of ranking to a binary problem. Training data for RankSVM consists of pairs of documents with a binary score that represents whether documents are ordered in a correct way (concordant, where the top document is more relevant than the bottom one, or discordant pair, where the opposite is true). Given a document-document pair(d, d), we can define a number of features. We first generate features separately for each of the two documents and a query (d, q). After this, we create features for a pair of documents by subtracting the features of the second document from the first document [4].

The resulting features serve as an input to the algorithm. At the classification stage, RankSVM is trying to predict whether two documents are ordered correctly in regards to the query. This is repeated for all possible pairs of different documents for a corresponding query. Then follows the reranking step. If the ordering of a pair is correct, the document at index position 0 of a pair is given a point. If the ordering is incorrect, the document at index position 0 of a pair is subtracted a point. After repeating this for all pairs, the result is a score for all documents, which after ordering, results in the final ranking.

4 EXPERIMENTAL SETUP

The experiment consists of several blocks: text normalisation, feature engineering and implementing logistic regression and RankSVM.

To adapt the approaches to medical domain, we did some feature engineering and apart from basic features suggested by previous research [5, 7] created new features, that we thought would be beneficial for searching queries in medical documents, as will be described in 4.2. We conduct several experiments with different feature combinations and choose the best combination performed on Pointwise Logistic Regression k-fold cross-validation.

As our baseline, we used run submitted by Microsoft Research team participated in TREC-9 competition in Filtering Track. We compared our final rankings for RankSVM, Pointwise Logistic Regression and Microsoft baseline approach and reported Mean Average Precision score, as will be described in 4.3. Our objective was to beat Microsoft baseline MAP score, which was 0.41.

4.1 Dataset

In our project we used OHSUMED dataset provided by TREC-9 Filtering Track [2]. This is a subset of a dataset on medical publications MEDLINE. The TREC-9 version of the dataset contains 63 queries on 348566 medical documents from 1987-1991. 16140 of the query-document pairs are labeled with a relevance rank: not relevant (0), possibly relevant (1) and definitely relevant (2)[2]. In line with TREC-9 final evaluation, we treated both possibly relevant and definitely relevant documents as relevant, making it a binary problem [8]. The documents that were not rated in TREC data were assigned the relevance rank 0.

As the training data for our models, we used the submission made by Microsoft Research team that participated in TREC-9 competition in Filtering Track, batch filtering task by the name of ok9, input.ok9bf2po [3]) [8].

We took the 30 top documents given by the Microsoft system for each of 63 queries available. The data was shuffled and further split into train, validation and test sets, with the following proportions: training set contained 40 queries, validation set 10 queries and test set 13 queries. For the features evaluation, K-folding was used, where a random subset of 13 topics was selected to be the test set, after which 5-fold cross-validation was performed (with 40 train topics and 10 validation topics) to obtain the best feature groups, by using the validation set as a guideline.

It is important to mention that the input data for Pointwise Logistic Regression and RankSVM was passed in different formats. Training data for Pointwise LR model consists of query-document pair features and relevance scores. Training data for RankSVM consists of pairs of documents and a score 1 or 0, where 1 means that the first document is more relevant than the second. This resulted into 1890 query-document pairs for Pointwise Logistic regression approach and 54,810 document-document pairs for RankSVM approach.

[2]https://trec.nist.gov/data/t9_filtering.html
[3]https://trec.nist.gov/results/trec9/filtering_input.html

4.2 Feature engineering

The first step of feature engineering consisted of text normalization, where we solely tokenized our words and casted them to lower-case letters.

As an inspiration for our feature engineering we used Microsoft Learning-to-Rank Data Sets features [5, 7] and enriched them with some new features.

We divided our features into 9 feature groups: Query Term Coverage, General Statistics, IDF, Partial TF, TF, TF-IDF, Cosine, Character Counts, POS Tags. A detailed description of the feature groups can be found in Appendix 7.1. Apart from some basic Microsoft features (namely Query Term Coverage, General Statistics, IDF, TF and TF-IDF), we came up with new feature groups, such as Cosine, Partial TF, Character counts and POS (part of speech) tags.

We expected Partial TF to work well, because in some cases certain keywords are subwords of larger words that inherently mean the same thing (e.g. 'fear' is a subword of 'fearful'). Enabling Partial matching for TF would result in more information in our features. We expected the cosine distance between tf-idf of the query and the document to be a descriptive feature as well since it represents the relation between query and document. The POS tags and character count feature groups were expected to contain information about the character of the text (for example: a title contains a dash or question mark).

We calculated different features for three different text streams associated with the documents (keywords, titles, abstracts) and then concatenated them. The keywords are MeSH topics, medical terms attached to each document in the database. Medical professionals use these to narrow down search terms. These three streams were chosen because they are expected to contain information about the relevance of the document for a query. This resulted into 27 feature groups max which equals to 456 features in total. We tested all the possible combinations of those feature groups on our models and reported mean average precision.

4.3 Evaluation

For evaluating the classification stage of our algorithms (binary prediction), we report ROC-AUC score. ROC-AUC could measure whether a random relevant document would receive a higher score than a random non-relevant document, which is important for the later ranking application.

For evaluating the final ranking we use mean average precision evaluation metric (MAP), commonly used in information retrieval. Average precision (AP) is calculated by computing a precision at every position in the ranked sequence of documents and taking the average value of precision of all documents ranked 'relevant' divided by the number of relevant documents in the collection. Mean average precision (MAP) is a mean of average previsions calculated for each of the queries.

We used k-fold cross validation to define the best set features (checking the performance of all sets in the powerset of the best 10 performing feature groups on the validation set) and all our final results are reported on the unseen test set.

5 RESULTS

5.1 Pointwise Logistic Regression and selection of best features

The Pointwise Logistic regression proved to be able to distinguish documents that were relevant for a query and documents that were not relevant for a query.

Performance for feature groups on Pointwise Logistic Regression can be found in Appendix 7.2. The best combination of the feature groups included 6 feature groups, namely 'ABS Cosine', 'KW Cosine', 'KW - Query Term Coverage', 'T Cosine', 'ABS Partial TF', 'ABS - Query Term Coverage' where ABS stands for abstract, KW for keyword and T for title stream. As expected, features that reflected the relation between term and query were most relevant (such as cosine, query term coverage, term frequency and partial term frequency). The performance of best combinations of features calculated on Pointwise Logistic Regression using k-fold cross-validation can be found in Appendix 7.3.

After the best feature group combination was determined using cross-validation, a final evaluation was performed, where the algorithm was trained on all non-test samples and predicted on the unseen test set. The coefficients of this final model are plotted in Appendix 7.5. Most descriptive feature group are KW Query Term Coverage, KW Normalized QTC, KW Cosine, ABS partial TF Sum Normalised, ABS Partial TF Mean Normalised, ABS Partial TF Variance Normalised and ABS Cosine.

The area under the ROC curve and MAP score for the final model are reported in Table 1. The ROC-AUC indicates that the model can differentiate between relevant and non-relevant documents, and the MAP score is higher than the Microsoft run baseline of 0.41.

5.2 Pairwise classification

RankSVM performed worse than pointwise LR and was more time-consuming. Due to the increased size of the data and due to time constraints (i.e. as we create pairs for all possible documents for a query), we were not able to evaluate features on RankSVM in the same manner as we did for pointwise LR. Training one instance of RankSVM for a decent size set of features already set us back an hour or more. We did, however, test RankSVM for the feature set that we choose as our final set for our point-wise method. Table 1 shows the AUC-ROC and MAP that the RankSVM was able to achieve for classifying whether or not the order of the pairs were or not. One can notice that the MAP is higher than our baseline, but it is not as high as the respective scores for our point-wise method.

Final result comparing RankSVM Pointwise LR methods can be found in the table below.

Table 1: RankSVM/Pointwise LR results (after cross-validation, with the best possible feature set on the entire training and test set). ROC-AUC stands for area under receiver operating characteric curve.

Model	Classification		Reranking
	ROC AUC (train)	ROC AUC (test)	MAP (test)
Pointwise LR	**0.66**	**0.73**	**0.68**
Pairwise RankSVM	0.54	0.50	0.44
Microsoft baseline	–	–	0.41

6 DISCUSSION

In this section we will reflect on our results and formulate answers to the questions stated in the introduction.

6.1 Features

The coefficients of the final features are shown in figure 1 in Appendix 7.5. We find that the Keyword query term coverage (QTC) and the cosine distance between keywords, title, abstract and the query were important features, in addition to the partial term frequency features. The keyword query-term coverage is a sensible important feature, as well as the TF-IDF vector of the query stream as they both capture the general meaning and topic of the text. The importance of partial term coverage is interesting, especially the importance of the variance component of the feature group. The strength of individual feature groups can be examined in Appendix 7.3. The cosine distance and TF features are the strongest. The lack of TF feature groups in the strongest subset of feature groups (Appendix 7.4) suggests that cosine and TF carry some redundant information.

Partial TF worked well in this case, but in some cases it might introduce errors, so it is not a method that can blindly be applied. In the context of a medical keyword, for example 'men', might be matched with 'menopause'. Here matching subwords would have introduced an error as both the word is matched to the subword, but they are probably not in the same realm of meaning.

6.2 RankSVM vs. pointwise logistic regression

First we will discuss how pairwise ranking (RankSVM) method compares to pointwise logistic regression. The results for pointwise logistic regression were considerably higher than our pair-wise methods, which tells us that the chosen features become more of a differentiating factor when predicting relevance versus the order of document relevance. This made us question whether or not the RankSVM was working and/or if the method was correct. To test this, we first appended the relevance judgement as a feature, which proved that the RankSVM itself was working as the score after appending the class was 1.0 for the training and test set. We also tried training a different model, to test whether the problem was specific to SVM. We used a logistic regression model instead of SVM and observed no differences in terms of ROC-AUC and MAP scores. A reason for the performance of RankSVM might be because the ranked list that we train on is ordered by relevance, which in turn consists of two levels (relevant and non-relevant). This means that our ranked list is inherently flawed as we do not know how the relevant documents are ordered with respect to each other and we also do not know how the non-relevant documents are ordered with respect to each other.

During our experimental phase, we observed that the resulting predictions for our point-wise methods were not as far removed from each other as one would expect. The expectation was that in terms of probabilities the non-relevant documents would be on the lower end of the predictions relevant documents would be on the higher end of the predictions (i.e. $p \ll 0.5$ and $p \gg 0.5$ respectively). Judging by the ROC-AUC score, the pointwise relevance prediction is not an easy task. For RankSVM, if the decision boundary is not clear then the binary classification is not as optimal as one would hope it to be. This means transitivity is lost and thus the following does not always hold: if ordering (doc1, doc2) and (doc2, doc3) is correct, then it follows that (doc1, doc3) is also correct. This could be an explanation as to why RankSVM did not perform well and pointwise logistic regression did. This hypothesis, however, was not fully tested and remains an open question for future research. Another option could be that there is not enough training data for the SVM to learn the required relationship.

It is important to mention that in Pointwise Logistic Regression if classification performs badly (as measured by discrete metrics, such as F1-score) there is still a possibility to get high MAP score and therefore good results on ranking task. Because Pointwise LR ranking is based not on the predicted class values themselves but on the probability estimates, incorrect classification does not necessarily mean an incorrect ordering of the probability values (hence the usage of AUC-ROC to monitor the performance).

6.3 Future research

Possible extensions would be to use the multiple relevance levels in the training of the algorithms. This may improve the ranking performance of both algorithms.

The use of additional data where there is not only a binary relevance score, but an entire ranking could improve the performance of the Pairwise SVM. This means that obtaining click-through data, or similar data, would be helpful.

REFERENCES

[1] William S. Cooper, Frederic C. Gey, and Daniel P. Dabney. 1984. User recovery and reversal in interactive systems. *SIGIR '92 Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval* (1984), 198––210.

[2] William Hersh, Chris Buckley, TJ Leone, and David Hickam. 1994. OHSUMED: An Interactive Retrieval Evaluation and New Large Test Collection for Research. (1994).

[3] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*. ACM, New York, NY, USA, 133–142. https://doi.org/10.1145/775047.775067

[4] Ching-Pei Lee and Chih-Jen Lin. 2014. Large-scale linear ranksvm. *Neural computation* 26, 4 (2014), 781–817.

[5] Sen Lei and Xinzhi Han. 2018. *Feature Selection and Model Comparison on Microsoft Learning-to-Rank Data Sets.*

[6] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. (2009).

[7] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. *CoRR* abs/1306.2597 (2013). http://arxiv.org/abs/1306.2597

[8] S. Robertson and D. Hull. [n. d.]. *The Trec-9 Filtering Track Final Report.*

[9] Qiang Wu, Chris JC Burges, Krysta M Svore, and Jianfeng Gao. 2008. *Ranking, boosting, and model adaptation.* Technical Report. Technical report, Microsoft Research.

7 APPENDIX

7.1 Feature description

Table 2: Features description

Features count	Feature group	Feature description	Feature name
0	Query Term Coverage	How many terms from the query occur in the document	Covered query term number
1	Query Term Coverage	Covered query term number divided by th enumber of query terms	Covered query term ratio
2	General Statistics		Query length
3	General Statistics		Stream length
4	IDF	1 divided by the number of documents that contain the query terms	IDF
5	Partial TF	How many times a query term is a substring of a term in the document and how many times a term in the document is a substring in a query term	Sum of partial TF (partial matching)
6	Partial TF		Min of partial TF
7	Partial TF		Max of partial TF
8	Partial TF		Mean of partial TF
9	Partial TF		Varience of partial TF
10	Partial TF		Normalised sum of partial TF
11	Partial TF		Normalised min of partial TF
12	Partial TF		Normalised max of partial TF
13	Partial TF		Normalised mean of partial TF
14	Partial TF		Normalised varience of partial TF
15	TF	How many times each of the query word occurs in the document	Sum of term frequency
16	TF		Min of term frequency
17	TF		Max of term frequency
18	TF		Mean of term frequency
19	TF		Varience of term frequency
20	TF		Normalized sum of stream length
21	TF		Normalized min of stream length
22	TF		Normalized max of stream length
23	TF		Normalized mean of stream length
24	TF		Normalized varience| of stream length
25	TF-IDF		Sum of words' tf*idfs
26	TF-IDF		Min of tf*idf
27	TF-IDF		Max of tf*idf
28	TF-IDF		Mean of tf*idf
29	TF-IDF		Varience of tf*idf
30	Cosine	Cosine distance between document and query tf-idf vectors	Tf-idf cosine distance
31 - 122	Character Counts	For each character count how many of those occur in the text (includes English alphabet, white space characters and punctuation marks)	92 character counts features
122 - 152	POS (Part-of-speech) Tags	For each of the POS tags count how many words in the document have this postag	31 POS tag features

7.2 Top 10 feature groups individual results

Table 3: Feature groups ranked by performance (ABS - abstract, KW - keywords and T - title stream)

Feature Group	Mean Validation MAP (5 folds)
ABS Cosine	0.549897
KW Cosine	0.527671
KW TF	0.522050
ABS TF	0.491927
KW - Query Term Coverage	0.474476
T Cosine	0.460525
ABS Partial TF	0.437935
ABS - Query Term Coverage	0.429816
KW Partial TF	0.426497
ABS TF-IDF	0.406383
ABS POS Tags	0.401065
KW Character Counts	0.398235
KW POS Tags	0.396493
KW General Statistics	0.396014
KW TF-IDF	0.393549
T Character Counts	0.389345
ABS General Statistics	0.388998
ABS Character Counts	0.368723
T General Statistics	0.364600
T TF-IDF	0.349383

7.3 Scores of top feature subset groups

Table 4: Scores of top feature subset groups

Feature Groups	Mean Validation MAP (5 folds)	Number of feature groups
('ABS Cosine', 'KW Cosine', 'KW - Query Term Coverage', 'T Cosine', 'ABS Partial TF', 'ABS - Query Term Coverage')	0.579328	6
('ABS Cosine', 'KW TF', 'KW - Query Term Coverage', 'T Cosine')	0.578810	4
('ABS Cosine', 'KW Cosine', 'KW - Query Term Coverage', 'ABS Partial TF', 'ABS - Query Term Coverage', 'ABS TF-IDF')	0.577741	6
('ABS Cosine', 'KW - Query Term Coverage', 'T Cosine')	0.577499	3
('ABS Cosine', 'KW - Query Term Coverage', 'T Cosine', 'ABS TF-IDF')	0.576175	4
('ABS Cosine', 'KW Cosine', 'KW - Query Term Coverage')	0.576139	3
('ABS Cosine', 'KW TF', 'T Cosine')	0.575641	3
('ABS Cosine', 'KW Cosine', 'KW - Query Term Coverage', 'ABS Partial TF', 'ABS - Query Term Coverage')	0.575631	5
('ABS Cosine', 'KW TF', 'KW - Query Term Coverage')	0.575200	3
('ABS Cosine', 'KW TF')	0.574649	2
('ABS Cosine', 'KW Cosine', 'KW - Query Term Coverage', 'T Cosine')	0.574387	4
('KW Cosine', 'KW - Query Term Coverage', 'T Cosine', 'ABS - Query Term Coverage')	0.574293	4
('ABS Cosine', 'KW Cosine', 'KW - Query Term Coverage', 'ABS - Query Term Coverage')	0.572945	4
('ABS Cosine', 'KW TF', 'KW - Query Term Coverage', 'ABS Partial TF', 'ABS - Query Term Coverage')	0.572673	5
('ABS Cosine', 'T Cosine', 'KW Partial TF')	0.572609	3
('ABS Cosine', 'KW Cosine', 'KW TF', 'KW - Query Term Coverage', 'ABS - Query Term Coverage')	0.572468	5
('ABS Cosine', 'KW Cosine', 'KW TF', 'KW - Query Term Coverage', 'ABS Partial TF', 'ABS - Query Term Coverage')	0.571725	6
('ABS Cosine', 'KW TF', 'KW - Query Term Coverage', 'T Cosine', 'ABS - Query Term Coverage')	0.571143	5
('ABS Cosine', 'KW Cosine', 'KW TF', 'KW - Query Term Coverage', 'T Cosine')	0.571087	5
('ABS Cosine', 'KW TF', 'KW - Query Term Coverage', 'T Cosine', 'ABS Partial TF')	0.570656	5

7.4 Pointwise Logistic Regression feature importances

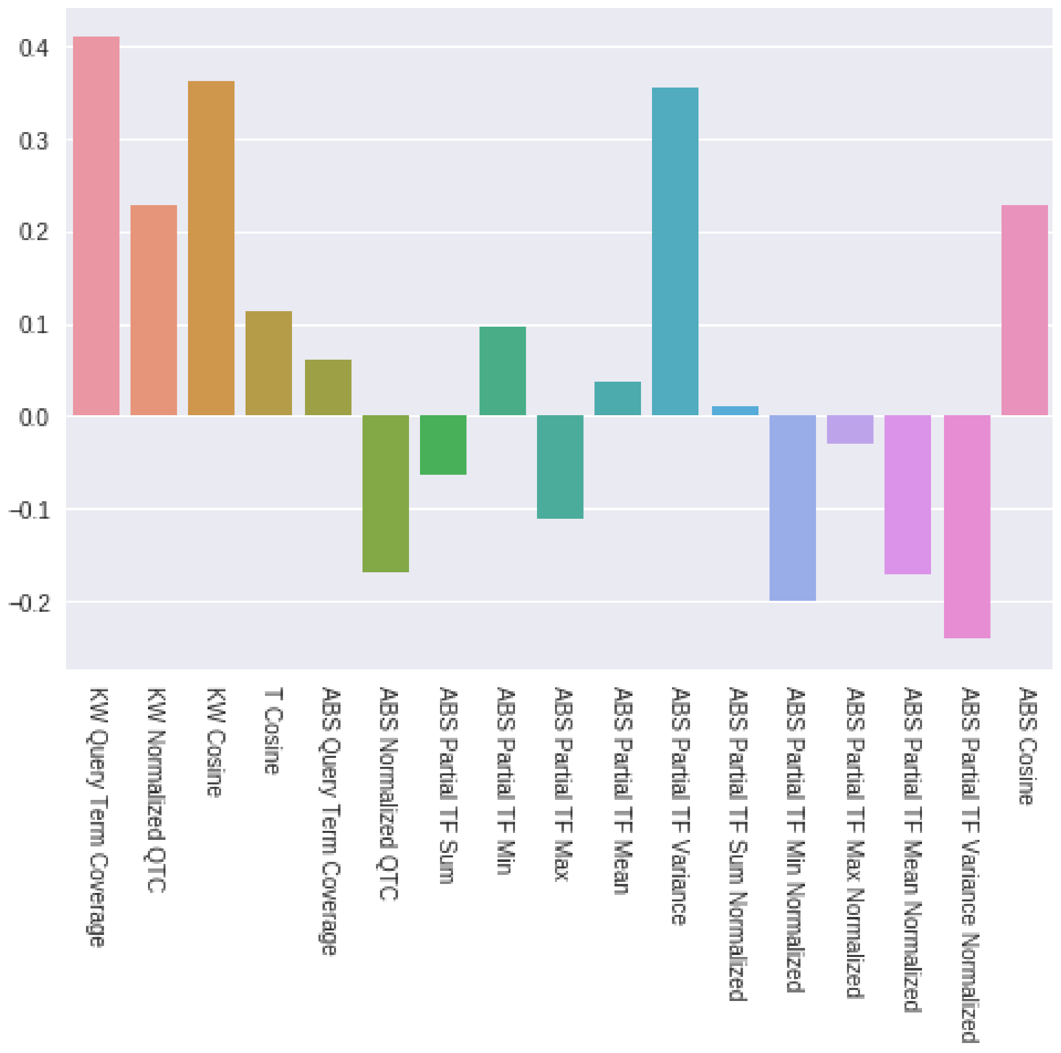

Figure 1: Features importances (ABS - abstract, KW - keywords and T - title stream)