Performance Comparison of Different Query Expansion and Pseudo-Relevance Feedback Methods

A comparison of Bo1, KL, RM3, and Axiomatic Query Expansion against BM25

Laurens de Swart

Supervisors: Avishek Anand, Jurek Leonhardt

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Laurens de Swart
Final project course: CSE3000 Research Project
Thesis committee: Avishek Anand, Jurek Leonhardt, Alan Hanjalic

Abstract

This paper is an analysis of the performance and logic behind different query expansion models. Query expansion and pseudo relevance feedback are techniques for adding more terms to a query based on the results of an initial query and the data in the body of documents. Four different query expansion models that are provided in the pyterrier python library and its extensions have been analysed, namely Bo1, KL, RM3, and Axiomatic query expansion. It was found that Axiomatic query expansion often does not perform any query expansion, and when it does, has no increase in performance. Bo1 and KL, although different in exact logic, have similar results most of the time. The most significant difference is the execution time, with Bo1 being faster with larger datasets and KL being faster with many documents on smaller datasets. Lastly, RM3 while not having a dominant performance has a lot of potential for good results with the right combination or parameters.

1 Introduction

Information retrieval, and more specifically ad-hoc retrieval — the process of returning a list of relevant documents from a collection, is a crucial part of the online experience. Search engines rely on it, and any proper searching function makes use of it. One of the main challenges of ad-hoc retrieval is to find relevant documents even when the terms don't match exactly, like finding them through synonyms. To improve the relevance of the results of search queries, methods like Query Expansion (QE) and Pseudo Relevance Feedback (PRF) can be used. These methods utilize techniques to add more terms to the given query based on the data from the corpus as a whole as well as the top document of an initial query with a regular information retrieval model. QE and PRF are not just one model but are abstract concepts, and there are many different models, each with different ways of calculating the weighs of the terms of a query. Presumably none of these models is completely dominant over all the others when it comes to complexity, speed, and mostly accuracy. In this paper, different well known models will be compared to each other on a variety of datasets and queries in order to get an understanding of what these models struggle with and excel at.

With a better understanding of what the different models do well and don't do well with, it could help with deciding which model to use in a certain scenario. Knowing what makes a model strong or weak in certain areas might also aid with designing new implementations in the future, as this area is still being developed.

One of the most commonly used models for information retrieval is BM25, so this will also be used as a benchmark to compare against. The main research question this paper will therefore be answering is: "What gain in retrieval performance do different QE and PRF methods achieve compared to standard BM25 across different domains and retrieval tasks?". To better answer that question, it can be subdivided into subquestions. The first of which is : "How well do different QE and PRF methods handle ambiguous queries compared to standard BM25?" and "What kind of queries do the different QE and PRF methods struggle with the most?". And lastly, to measure performance outside just accuracy, the question of "How do the different QE and PRF methods compare in terms of execution time and resources?" will be answered.

2 Methodology

Firstly, it needs to be decided which models will be used for the comparisons. The library that will be used for the information retrieval is called pyterrier [12] and comes with several already implemented query expansion models. The tested models will therefore be Bo1, KL, RM3, and Axiomatic. To get a good measure across various different queries, the tests will be run on multiple datasets. These datasets are included with pyterrier, but some extra datasets from the ir_datasets [11] will also be used. The golden standard for these types of datasets are the TREC datasets, but other datasets will also be used.

To answer the first research question of finding an objective way of measuring the performance of the different methods, the ir-measures library [10] will be used. This library allows for various different statistical measurements to be used and is well integrated with pyterrier. These different performance measures include the measures that will be needed for objective measuring.

To see how well ambiguous queries do as well as what the different models excel at and struggle with, the best and worst queries will be extracted to see if there is any observable pattern between them.

In order to understand the later results, it is important to understand the models first. All models use similar variables, which mean the following things:

- $tf(t, d)$ = The term frequency of term t in document d, so the amount of occurrences of a specific word in the document.
- $os(d)$ = The original score of a document given by the model that feeds into the current model.
- R = The (pseudo-)relevant documents.
- C = The entire collection of documents (the corpus).
- Q = The original query.

2.1 Bo1

The Bo1 model is the most effective variant of the Divergence From Randomness (DFR) term weighting model [9] [15]. This weighting model is based on Bose-Einstein probability [13]. Each candidate expansion term t gets a score, which is measured with equation 1 as used in [13] rewritten from the original [1, p. 165].

$$S(t) = \left(\sum_{d \in R} tf(t,d)\right) \cdot \log_2\left(\frac{1 + f_{avg}(t,C)}{f_{avg}(t,C)}\right) + \log_2(1 + f_{avg}(t,C)) \quad (1)$$

where

$$f_{avg}(t, C) = \sum_{d \in C} tf(t, d)/|C|$$

Here, the function $f_{avg}(t, C)$ is used to calculate the average frequency of a term t in some collection of documents C. The function $S(t)$ can be broken down to be understood better. The summation counts up the total amount of occurrences of the term t in the pseudo-relevant documents. This summation then acts as a weight for the first logarithm based on how common the term is in the pseudo-relevant documents. Do notice that this weight is linear, unlike the rest of the formula, which has logarithmic scales.

The first logarithm is not very intuitive at first sight, but can be rewritten as $\log_2(1 + \frac{1}{f_{avg}(t,C)})$, here, the $1+x$ is used to avoid the behaviour of logarithms between 0 and 1 where they would become negative which could lead to the entire result of $S(t)$ becoming negative. The fraction within the logarithm leads to an inverse relationship between the average frequency of a term and its weight. The less common a word is, the larger this part of the weight becomes. This acts as a counterbalance against the summation. A common word like 'the' would see a large number coming from the summation, but since this word would be very common across the entire corpus this logarithm would then act as a counterbalance and see that the term 'the' is not a good term to expand with as it is so common.

The last logarithm like the first logarithm uses the format of $1 + x$ to get the desired logarithmic behaviour, but unlike the first logarithm, the relationship and the term frequency across the entire corpus is not inverted, giving a higher score to common words. This is presumably done to improve the effectiveness of adding a term to the query. If a term is found once in one of the pseudo-relevant documents but nowhere else, it would get a decently large weight from the summation and the first logarithm, but adding this term to the query would not result in any extra relevant documents. This term might be relevant to this query but if there are no other documents containing the term then it wouldn't yield any better accuracy and would take up a spot of feedback terms, which are limited.

So Bo1 selects the least common terms from the total collection with the largest amounts of occurrences in the pseudo relevant documents, while trying to make sure that the terms it does select appear enough in the total collection for it to be worth expanding with.

This function $S(t)$ is then used in equation 3, this score is part of the equation of calculating the score for candidate expansion terms. This calculated score is used to merge the original query with the candidate expansion terms. These equations 2-4 are as mentioned in [13] are used to calculate this score, but with equation 2 slightly modified to correspond to the original in [1, p. 159].

$$score_{orig}(t) = \frac{tf(t, Q)}{\max\limits_{t' \in Q} tf(t', Q)} \tag{2}$$

$$score_{exp}(t) = \frac{S(t)}{\max\limits_{t' \in d \in R} S(t')} \tag{3}$$

$$score(t) = score_{orig}(t) + score_{exp}(t) \tag{4}$$

Equation 2 is used to get the normalised weight of the term in the query. Usually this would lead to either a value of 1 or 0 when there is no term that appears more than once in a query. A 1 if the term appears in the query and 0 if it does not. If there exists a term in the query that appears more than once in the query, then this result becomes some fraction. This $score_{orig}$ was presumably added to the score calculation to ensure that the term from the original query would formally still appear in the expanded query, although with a term that appears more than once this effect could backfire.

Equation 3 similarly served to normalize the score to map the scores to a score between 0 and 1. This is done by dividing the scores of the candidate terms by the highest scoring term in the pseudo-relevant documents.

Lastly, equation 4 then combines the scores of equations 2 and 3 to get the final score. This score can range from 0 to 2. Although for terms that do not appear in the original query this score can only range from 0 to 1 because their $score_{orig}$ will always be 0. This guarantees that the terms of the original query will appear in the expanded query for queries where all terms appear once. Queries with terms that appear more than once will allow for the possibility of a term that appears in the original query to have a lower score than a term that didn't appear in the original query.

This scoring system is also used in the KL query expansion in section 2.2. When experimenting, it will be interesting to see if the amount of times a term appears in the query has any relation to the accuracy measures, as it would be the only significant way a query can impact the $score_{orig}$ component of the score.

2.2 KL

The KL model is a model based on the Kullback-Leibler divergence approximation [1, p. 164]. This model is similar in structure to the Bo1 model mentioned in section 2.1, it also uses the equations 2-4, but it has a different definition for $S(t)$. Its definition of $S(t)$ is calculated by equations 5-7, as used in [13].

$$p_r(t) = \frac{\sum\limits_{d \in R} tf(t, d)}{\sum\limits_{d \in R} \sum\limits_{t' \in d} tf(t', d)} \tag{5}$$

$$p_c(t) = \frac{\sum\limits_{d \in C} tf(t, d)}{\sum\limits_{d \in C} \sum\limits_{t' \in d} tf(t', d)} \tag{6}$$

$$S(t) = p_r(t) \cdot \log \frac{p_r(t)}{p_c(t)} \tag{7}$$

Equations 5 and 6 are very similar in structure, the only difference between them being the collection of documents they use. Equation 5 calculates the unigram probability distribution of the pseudo relevance documents, while 6 calculates it for the entire collection of documents in the corpus. These equations calculate the percentage of the amount of occurrences of the term t compared to the total amount of terms

in the collection R or C depending on the equation. The final step in equation 7 then calculates the Kullback-Leibler divergence of the p_r compared to p_c. The Kullback-Leibler calculates how different two distributions are from each other [8].

To break this down, the equation mostly relies on its logarithm. This logarithm compares the 2 percentages, if the term t takes up a larger percentage in the pseudo-relevant documents than in the entire corpus, then the fraction is going to be greater than 1. Likewise, when the term is less common in the pseudo-relevant documents than in the rest of the corpus, this fraction will be less than 1. The logarithm then maps this fraction to have anything less than 1 become less than 0 and anything greater than 1 to be greater than 0 while also putting it on a logarithmic scale. But because weights cannot be negative in these weighting models, any result less than 0 gets clamped to 0. So any document where the term is less common in the pseudo relevant documents than the rest of the corpus always gets a weight of 0. After this logarithm is applied the weights will be somewhat normalised, so then the multiplication with p_r serves as a weight for the term.

So the KL expansion model gives high weight to terms that are a lot more common within the pseudo relevant documents, and further boosts terms that appear more often in the pseudo relevant documents compared to other terms in those same documents.

2.3 RM3

RM3 is different from Bo1 and KL query expansion, because it actually uses the scores from the model that feeds into it, rather than just a set of the most relevant documents. This can be seen in equation 8, as it uses the $os(d)$ function, which gets the score of a document assigned by the model that feeds into this model. This equation is not the exact equations from the original paper [5] but rather what is in the actual implementation of pyterrier PRF library [12].

$$S(t) = \frac{1}{|R|} \sum_{d \in R} \frac{tf(t,d)}{|d|} \cdot os(d) \tag{8}$$

Equation 8 mostly relies on the original score, which gets weighted by the percentage of the document that the current term makes up. This weight is calculated for every document in the pseudo-relevant documents and then averaged. So this model gives the largest weights to the terms that come from a high scoring document from the initial query, that also take up a significant portion of that document.

Ideally, the set R would include the entire collection of documents, but that would be feasible due to computational limits [5, p. 4]. This really highlights the difference between this model and the ones mentioned before, which rely on set R being the most relevant documents only. However, this does mean that RM3 is highly dependent on the quality of the scores of the model that feeds into it, and the difference in even the distribution of the scores could impact its effectiveness.

RM3 calculates its final score differently from the methods before, it uses equation 9.

$$score(t) = \alpha \cdot score_{orig}(t) + (1 - \alpha) \cdot score_{exp}(t) \tag{9}$$

where

$$0 \leq \alpha \leq 1$$

$$score_{orig}(t) = \frac{tf(t,Q)}{|Q|}$$

$$score_{exp}(t) = \frac{S(t)}{\sum_{d \in R} \sum_{t' \in d} S(t')}$$

Here, the constant α is a property of the RM3 model that can be defined when first initialising the model. This constant is then used to linearly interpolate between $score_{orig}$ and $score_{exp}$. With an alpha of 1, candidate terms only get weighted by their term frequency in the original term, so no query expansion will take place. With an alpha of 0, candidate terms fully rely on the S function, and thus the score calculated by the model that feeds into the RM3 model.

2.4 Axiomatic

Axiomatic query expansion, like other query expansion models, receives a set of the most relevant documents from an initial query. But unlike other models, the Axiomatic model adds irrelevant documents to the set of pseudo-relevant document. An Axiomatic model has a property R which is the amount of non-relevant documents it analyses. The model's core functionality comes from the Mutual Information equation, see equation 12. This was possibly done to remedy erroneous query expansion for common terms by adding non-relevant document that likely also contain those common terms and thereby make them less correlated within the pseudo-relevant set. The definition of S(t) for the Axiomatic model is defined in equation 10. These equations are slightly modified from the original [18] to be more consistent with the rest of the equations.

$$S(t) = \sum_{q \in Q} s(q,t)/|Q| \tag{10}$$

where

$$s(q,t) = \begin{cases} idf(q) & \text{if t = q} \\ idf(q) \cdot \beta \cdot \frac{MI(q,t)}{MI(q,q)} & \text{if t} \neq \text{q} \end{cases} \tag{11}$$

$$MI(q,t) = \sum_{X_q, X_t \in \{0,1\}} p(X_q, X_t|R) \cdot \log \frac{p(X_q, X_t|R)}{p(X_q|R)p(X_t|R)} \tag{12}$$

This model uses a quite different set of equations compared to the other models, including some new variables. The idf of q, which is the inverse document frequency, so how rare the term is within the corpus. β is some defined constant in the model which indicates the 'trustworthiness' of the semantically-related term. Equation 12 calculates the semantic distance between two terms using mutual information (MI) within the pseudo-relevant documents. Within equation 12 X_q and X_t denotes the presence or absence of terms q and t respectively with a 1 or a 0 [18, p. 371]. The function p is used to calculate the probabilities of seeing either 2 terms together or just a single term within a set of documents.

Equation 10 is used to simply calculate the average score s of term t for every term q within the original query. Equation

11 then, if t matches with the term q, simply takes inverse document frequency. However, if the terms t and q do not match exactly, the score can still be used by weighting the inverse document frequency by its semantic distance. If the terms have a low semantic distance, which is a high score MI, the terms are most likely related, and adding it to the query would potentially improve the accuracy. This semantic distance is calculated between the current term and the term in the query and is normalized against the query term against itself, because that would be the maximum semantic distance possible. But since the initial query results are not perfect and there are almost always errors in the ranking of the pseudo-relevant set, this semantic distance is weighted with β to try to account for these inaccuracies.

Equation 12 is used to calculate the mutual information between two terms q and t. It does this by going through all the combinations of X_q and X_t that are either 0 or 1. For each combination, it calculates how related they are by dividing the probability of seeing the terms together within the pseudo-relevant set of document by the probabilities of seeing the individual terms multiplied together. If the 2 terms were completely uncorrelated, then the following would hold: $E[X_q X_t] = E[X_q] \cdot E[X_t]$ [14]. Therefore, $\frac{E[X_q X_t]}{E[X_q] \cdot E[X_t]}$ would tend to 1, and the logarithm of that fraction would tend to 0, so unrelated terms would get very low scores for their MI. However, if they are related, their probabilities would be correlated and $E[X_q X_t]$ would be higher or lower than $E[X_q] \cdot E[X_t]$. This deviation is then weighted by the chance of the terms appearing together in the pseudo-relevant set, because the higher that probability is, the more relevant the value of the fraction is.

3 Experimental Setup and Results

3.1 Metrics

To measure the accuracy of a model, experiments have to be run on it, in this case pyterrier provides a built-in solution for these experiments. Each dataset that will be used also has a set of topics, which are queries where the 'correct' rankings of documents have already been decided, which are stored as so called qrels. Since the 'correct' rankings are already known, it is then only a matter of how to measure how close a model's results are to that ranking. There are many measures provided in the ir_measures library, and the following will be used in this paper:

- nDCG is the Discounted Cumulative Gain [6]
- (M)AP is the (Mean) Average Precision, which like the name implies gives the average precision score averaged over all queries [3].
- (M)RR is the (Mean) Reciprocal Rank with values ranging from 0 to 1, where 1 is a perfect result [7]. It is a commonly used measure

Each metric is tested at different cut-offs, namely 10, 100, and 1000. This entails that the metric is only calculated for the top n documents. This can give a better idea of where a model gains or loses the most accuracy. The lower cut-offs show how well a model performs in getting the absolute most relevant documents, while lower cut-offs show how well it performs in the overall query result. These cut-offs values are noted with an '@' symbol.

In order to get an idea of the execution time, the experiments have been executed separately for each model, so the time could be tracked. This execution time does however also include the time it took to run the BM25 model twice for the input and output of the query expansion model. To account for this, there is also a corrected time column in the experiment results, which is the time it took to run that model minus two times the time it took to run the BM25 model on the same dataset. This isn't a perfect way to account for it because it doesn't take into consideration the overhead of running the experiment. If the time it takes to run the query expansion model is less than 2 times the overhead time of running the experiment, this corrected time could be negative.

3.2 Datasets

There are many datasets included within the ir_datasets library, however not every dataset has the right requirements to be tested. The dataset needs to have a corpus or index as well as topics and qrels. If a dataset has neither a corpus nor an index, then it is significantly more difficult to use with the pyterrier library, so for convenience's sake the datasets were not viable options. Otherwise, if a dataset does not have topics or qrels then it is not possible to run experiments on these datasets, as they require them to calculate the accuracy measures, as well as the library code not knowing what queries to test. Considering these requirements, the following datasets have been selected:

- MS MARCO-passage [2], which is described as "A passage ranking benchmark with a collection of 8.8 million passages and question queries."[1]. More specifically, the "trec-dl-2019/judged", "trec-dl-2020/judged", and "trec-dl-hard" datasets have been used. These are subsets of TREC Deep Learning queries that were judged by NIST assessors.
- Beir ArguAna [17; 16] or "beir/arguana", "A version of the ArguAna Counterargs dataset, for argument retrieval."[2].
- Antique [4], "ANTIQUE is a non-factoid question answering dataset based on the questions and answers of Yahoo!"[3]
- Deep TREC Learning Docs [2], another dataset from MS MARCO. For this paper the training variant of this dataset was used, this dataset is very large and is in this case used as a stress test.

Because some of these datasets are very large and would take an incredibly long time to run, the amount of topics that are tested have been limited to the first 1000 topics. All the datasets have been tested with the stated metrics, models, and datasets. The RM3 does have an extra parameter, in pyterrier

[1] https://ir-datasets.com/msmarco-passage.html#msmarco-passage/

[2] https://ir-datasets.com/beir.html#beir/arguana

[3] https://ir-datasets.com/antique.html#antique/test

it's called the lambda, but in the equations it's stated as α. After some trial and error, it was found that an α value of 0.8 seemed to achieve good results in different data sets.

Each query expansion pipeline is performed the same way, which is BM25 >> QE model >> BM25. This means that it first does a BM25 query, the results of that query will then be fed into the query expansion model. The query expansion model only takes the top document from that query, the exact amount depends on the fb_docs constant defined in the model. The query expansion model then adds a number of terms to the query with a specified weight. The amount of terms that get added to the query is decided by the fb_terms constant defined in the model. Then, this new query is run with the second BM25 model.

3.3 Results

The Axiomatic query expansion in all results except for the ArguAna dataset has the exact same results as the BM25 model. Here the results are still extremely similar to the baseline (± 0.3) and the measures are still identical with each other no matter what combination fb_docs/fb_terms it has. It is also notable that the execution time is also negligible in all datasets except for ArguAna, where it has a similar execution time to the other models. Even though this is the only dataset where it seems to be working, it is still outperformed by BM25.

The query expansion performed best on the "msmarco-passage/trec-dl-2019/judged" dataset, in this dataset it gained the most over the BM25 model, specifically in the @10 cut-off. Especially the RM3 model performed well in the RR metric, while Bo1 and KL both excelled at the nDCG and AP scores. These results can be seen in table 1

The worst performing query expansion dataset is the "trec-deep-learning-docs" dataset. Here the RM3 model only slightly underperforms compared to BM25, but Bo1 and KL both have significantly lower scores in all metrics and only seems to get worse with more feedback documents and scores. These results can be found in table 2.

Bo1 and RM3 are very close in execution time through the entire experiment, KL is close in execution time on smaller datasets with a small amount of feedback documents and feedback terms but has a significantly faster execution time on those same smaller datasets but with more feedback documents. On large datasets like "trec-deep-learning-docs" the KL model is slower with 966 seconds compared to around 650-700 seconds for Bo1 and RM3 with 50 feedback documents and 50 feedback terms.

The full results of the entire experiment can be found in appendix A.

Model	RR@10	nDCG@10	AP@10
BM25	0.6397	0.4795	0.1083
fb_docs = 3, fb_terms = 10			
Bo1	0.6245	0.5086	0.1210
KL	0.6152	0.5057	0.1197
RM3	0.6582	0.5072	0.1189
Axiomatic	0.6397	0.4795	0.1083
fb_docs = 10, fb_terms = 40			
Bo1	0.5859	0.5031	0.1201
KL	0.6118	0.5039	0.1216
RM3	0.6488	0.4853	0.1073
Axiomatic	0.6397	0.4795	0.1083
fb_docs = 50, fb_terms = 50			
Bo1	0.6720	0.5120	0.1239
KL	0.6789	0.5048	0.1218
RM3	0.6605	0.4975	0.1150
Axiomatic	0.6397	0.4795	0.1083

Table 1: Experiment results for dataset "msmarco-passage/trec-dl-2019/judged"

4 Responsible Research

4.1 Reproducability

For ensuring reproducibility, this paper has described in detail the datasets, models, and measures used to obtain all the results, allowing the verification and further development upon these findings. The inner logic and motivation behind every model has been explained in detail, informing future researchers of their functionality and applicability. The code written to produce the results has also been uploaded to an online private repository, where the exact results in this paper can be reproduced.

4.2 Plagiarism

All the used papers and origins of the models, datasets, and measures that have helped with obtaining relevant results have been properly attributed, as well as papers that aided in notation or understanding of the models. Information and ideas derived from other works have been paraphrased appropriately and cited accordingly to give credit to the original authors.

5 Discussion

From all the results, the most curious seems to be the Axiomatic query expansion model. Its scores are identical to the baseline model BM25 in all experiments except for the ArguAna dataset. Even if the model performed a minimal amount of query expansion, it would have some difference is metrics and from further manual testing it was confirmed that in most cases the model does not perform any query expansion. This is also in line with the execution times, where in almost all cases the execution times are negligible. Looking through the source code of this library, nothing seems too off. The only thing of note is that the code that is equivalent to the function $s(q, t)$ from equation 11 is missing the normalisation score of $MI(q, q)$. It could also have to do with the mutual information function itself. It looks for any correlation between the query and the term, no matter if it's negative correlation. However, a term with a negative correlation to the query term does not seem like the kind of term that would be semantically relevant in a search query. The results of the Axiomatic query expansion should likely be disregarded.

Model	RR@10	nDCG@10	AP@10
BM25	0.2271	0.2868	0.2285
fb_docs = 3, fb_terms = 10			
Bo1	0.2043	0.2643	0.2069
KL	0.2034	0.2628	0.2053
RM3	0.2041	0.2643	0.2057
Axiomatic	0.2271	0.2868	0.2285
fb_docs = 10, fb_terms = 40			
Bo1	0.1664	0.2217	0.1678
KL	0.1663	0.2234	0.1677
RM3	0.2129	0.2714	0.2148
Axiomatic	0.2271	0.2868	0.2285
fb_docs = 50, fb_terms = 50			
Bo1	0.1312	0.1723	0.1326
KL	0.1429	0.1870	0.1444
RM3	0.2088	0.2643	0.2101
Axiomatic	0.2271	0.2868	0.2285

Table 2: Experiment results for the "train" variant of the dataset "trec-deep-learning-docs"

The RM3 model like was stated in [5] performs better with a larger amount of feedback documents, but it doesn't always necessarily perform better with the most amount of feedback documents. In "msmarco-passage/trec-dl-2019/judged" and "msmarco-passage/trec-dl-2020/judged" it performs very well with the least amount of document, but besides those datasets it seems to perform best somewhere between the 10 and 50 documents. RM3 is highly reliant on the actual score that are fed into it, in these experiments it was only tested with BM25 as input model. It could be that different input models could lead to better performance, a more exhaustive experiment would have to be run with multiple input and maybe output models, and then also with multiple lambda/α constants.

The Bo1 and KL model seem to perform very similarly in almost all cases. This seems to align with the similarity in logic that is behind them. There doesn't seem to be much of a rhyme or reason behind which model performs better in certain situations. The difference in execution time that was observed on large models could be due to the amount of dependence the different models have on the pseudo-relevant set. Bo1 only requires the term frequency of the term in the document, an amount that is score in the index. However, the KL model has to recalculate the percentage of the term in the pseudo-relevant set of documents every time, which is not stored directly in the index.

6 Conclusions and Future Work

In conclusion, in a lot of cases query expansion models don't add much accuracy to search results and often even perform worse. Both the Bo1 and KL models work decently and have a chance to improve the accuracy of query results. In respects to execution time, KL is quite a bit faster than Bo1 is smaller datasets with a larger number of feedback documents, but KL is significantly slower with large datasets. RM3 is not necessarily the best or the fastest, but it has many parameters that can be tweaked, like its lambda/α constant, the model that feeds into it, the amount of documents it analyses, and the amount of terms it adds. This balance of parameters can be very difficult to solve, and there is not just one answer that works for every dataset. Axiomatic query expansion in its current form seems to have some issues in the library implementation of pyterrier.

6.1 Future Work

For future work, the performance of Axiomatic query expansion can be analysed with a better version of the code and could be tried with a mutual information equation that doesn't reward negative correlation. RM3 could be tested with many more configurations to extract the most potential. All models could be tested more using different models for the initial query, adding a whole new dimension to fitting a search engine to a specific dataset.

A Full Experiment Results

A.1 MS MARCO passage

Metrics	RR(rel=2)			nDCG			AP(rel=2)				Corrected
	@10	@100	@1000	@10	@100	@1000	@10	@100	1000	Time	Time
BM25	0.6397	0.6402	0.6402	0.4795	0.4874	0.5934	0.1083	0.2322	0.2864	2.1775	-
fb_docs = 3, fb_terms = 10											
Bo1	0.6245	0.6267	0.6267	0.5086	0.5001	0.6109	0.1210	0.2488	0.3085	5.3382	0.9831
KL	0.6152	0.6176	0.6176	0.5057	0.4984	0.6082	0.1197	0.2472	0.3066	5.2921	0.9370
RM3	0.6582	0.6588	0.6588	0.5072	0.5065	0.6103	0.1189	0.2492	0.3069	5.2456	0.8905
Axiomatic	0.6397	0.6402	0.6402	0.4795	0.4874	0.5934	0.1083	0.2322	0.2864	3.4263	-0.9288
fb_docs = 10, fb_terms = 40											
Bo1	0.5859	0.5907	0.5907	0.5031	0.5160	0.6251	0.1201	0.2587	0.3199	18.3040	14.9763
KL	0.6118	0.6162	0.6162	0.5039	0.5136	0.6221	0.1216	0.2576	0.3180	17.3504	14.0227
RM3	0.6488	0.6494	0.6494	0.4853	0.4999	0.6041	0.1073	0.2400	0.2973	15.4450	12.1173
Axiomatic	0.6397	0.6402	0.6402	0.4795	0.4874	0.5934	0.1083	0.2322	0.2864	3.3676	0.0400
fb_docs = 50, fb_terms = 50											
Bo1	0.6720	0.6753	0.6753	0.5120	0.5111	0.6273	0.1239	0.2555	0.3190	16.3530	13.0163
KL	0.6789	0.6824	0.6824	0.5048	0.5072	0.6223	0.1218	0.2519	0.3148	13.2181	9.8814
RM3	0.6605	0.6631	0.6631	0.4975	0.5030	0.6082	0.1150	0.2432	0.3007	21.0050	17.6682
Axiomatic	0.6397	0.6402	0.6402	0.4795	0.4874	0.5934	0.1083	0.2322	0.2864	3.3477	0.0110

Table 3: Experiment results for the dataset "msmarco-passage/trec-dl-2019/judged"

Metrics	RR(rel=2)			nDCG			AP(rel=2)				Corrected
	@10	@100	@1000	@10	@100	@1000	@10	@100	1000	Time	Time
BM25	0.6147	0.6183	0.6184	0.4936	0.5026	0.5981	0.1827	0.2753	0.2930	2.1133	-
fb_docs = 3, fb_terms = 10											
Bo1	0.6133	0.6197	0.6197	0.4947	0.5325	0.6311	0.1837	0.2928	0.3141	6.6708	2.4443
KL	0.6140	0.6203	0.6203	0.4927	0.5320	0.6301	0.1817	0.2910	0.3124	6.6096	2.3830
RM3	0.6044	0.6091	0.6091	0.5005	0.5241	0.6217	0.1897	0.2901	0.3099	6.6984	2.4718
Axiomatic	0.6147	0.6183	0.6184	0.4936	0.5026	0.5981	0.1827	0.2753	0.2930	4.6291	0.4025
fb_docs = 10, fb_terms = 40											
Bo1	0.5724	0.5789	0.5789	0.4864	0.5299	0.6270	0.1769	0.2910	0.3117	23.5284	19.3824
KL	0.5699	0.5762	0.5762	0.4866	0.5237	0.6227	0.1760	0.2878	0.3087	22.5851	18.4390
RM3	0.6168	0.6212	0.6212	0.5004	0.5209	0.6185	0.1878	0.2865	0.3061	19.6396	15.4936
Axiomatic	0.6147	0.6183	0.6184	0.4936	0.5026	0.5981	0.1827	0.2753	0.2930	4.0406	-0.1054
fb_docs = 50, fb_terms = 50											
Bo1	0.6223	0.6300	0.6302	0.4897	0.5208	0.6218	0.1713	0.2844	0.3041	19.3580	15.2179
KL	0.6250	0.6319	0.6321	0.4922	0.5172	0.6176	0.1741	0.2811	0.3006	15.0699	10.9298
RM3	0.6103	0.6161	0.6163	0.4951	0.5097	0.6092	0.1783	0.2765	0.2958	27.3637	23.2237
Axiomatic	0.6147	0.6183	0.6184	0.4936	0.5026	0.5981	0.1827	0.2753	0.2930	4.0379	-0.1021

Table 4: Experiment results for the dataset "msmarco-passage/trec-dl-2020/judged"

Metrics	RR(rel=2)			nDCG			AP(rel=2)				Corrected
	@10	@100	@1000	@10	@100	@1000	@10	@100	1000	Time	Time
BM25	0.4151	0.4216	0.4221	0.2743	0.3098	0.3905	0.0944	0.1358	0.1471	2.8805	-
fb_docs = 3, fb_terms = 10											
Bo1	0.3987	0.4082	0.4087	0.2666	0.3114	0.3992	0.1028	0.1461	0.1586	7.3460	1.5850
KL	0.3927	0.4022	0.4027	0.2641	0.3101	0.3966	0.1007	0.1444	0.1569	7.1923	1.4313
RM3	0.4063	0.4128	0.4133	0.2720	0.3127	0.3983	0.0998	0.1415	0.1538	7.3763	1.6152
Axiomatic	0.4151	0.4216	0.4221	0.2743	0.3098	0.3905	0.0944	0.1358	0.1471	5.0742	-0.6869
fb_docs = 10, fb_terms = 40											
Bo1	0.3903	0.3969	0.3972	0.2725	0.3218	0.4058	0.1061	0.1503	0.1631	22.8186	17.7488
KL	0.3870	0.3955	0.3958	0.2636	0.3156	0.3990	0.1000	0.1439	0.1565	22.0163	16.9464
RM3	0.4287	0.4360	0.4365	0.2745	0.3185	0.4019	0.0996	0.1425	0.1547	19.3910	14.3212
Axiomatic	0.4151	0.4216	0.4221	0.2743	0.3098	0.3905	0.0944	0.1358	0.1471	4.7172	-0.3527
fb_docs = 50, fb_terms = 50											
Bo1	0.4010	0.4108	0.4111	0.2634	0.3067	0.3990	0.0930	0.1346	0.1482	20.1438	15.4198
KL	0.4012	0.4127	0.4130	0.2577	0.3056	0.3979	0.0935	0.1343	0.1480	16.0436	11.3197
RM3	0.4132	0.4221	0.4225	0.2649	0.3077	0.3933	0.0924	0.1347	0.1466	27.1663	22.4423
Axiomatic	0.4151	0.4216	0.4221	0.2743	0.3098	0.3905	0.0944	0.1358	0.1471	4.7169	-0.0070

Table 5: Experiment results for the dataset “msmarco-passage/trec-dl-hard”

A.2 Arguana

Metrics	RR(rel=1)			nDCG			AP(rel=1)				Corrected
	@10	@100	@1000	@10	@100	@1000	@10	@100	1000	Time	Time
BM25	0.1766	0.1836	0.1837	0.2603	0.2902	0.2924	0.1759	0.1829	0.1829	20.2789	-
fb_docs = 3, fb_terms = 10											
Bo1	0.1708	0.1793	0.1794	0.2510	0.2862	0.2885	0.1701	0.1786	0.1787	70.1550	29.5972
KL	0.1703	0.1790	0.1791	0.2499	0.2862	0.2884	0.1696	0.1783	0.1784	69.9291	29.3712
RM3	0.1664	0.1752	0.1753	0.2438	0.2814	0.2839	0.1658	0.1745	0.1747	69.4420	28.8841
Axiomatic	0.1764	0.1834	0.1835	0.2602	0.2900	0.2922	0.1757	0.1826	0.1827	69.7353	29.1775
fb_docs = 10, fb_terms = 40											
Bo1	0.1506	0.1592	0.1593	0.2330	0.2698	0.2715	0.1499	0.1585	0.1586	73.7515	34.4521
KL	0.1546	0.1633	0.1634	0.2366	0.2729	0.2751	0.1539	0.1626	0.1627	72.5635	33.2641
RM3	0.1648	0.1731	0.1732	0.2459	0.2810	0.2831	0.1641	0.1723	0.1724	71.7110	32.4116
Axiomatic	0.1764	0.1834	0.1835	0.2602	0.2900	0.2922	0.1757	0.1826	0.1827	69.5169	30.2176
fb_docs = 50, fb_terms = 50											
Bo1	0.1426	0.1527	0.1528	0.2181	0.2624	0.2646	0.1420	0.1521	0.1522	74.5036	35.1375
KL	0.1520	0.1621	0.1621	0.2287	0.2710	0.2732	0.1515	0.1614	0.1615	75.1204	35.7543
RM3	0.1542	0.1640	0.1641	0.2305	0.2725	0.2745	0.1535	0.1633	0.1634	78.3234	38.9573
Axiomatic	0.1764	0.1834	0.1835	0.2602	0.2900	0.2922	0.1757	0.1826	0.1827	71.9999	32.6338

Table 6: Experiment results for the dataset “beir/arguana”

A.3 Antique

Metrics	RR(rel=3)			nDCG			AP(rel=3)				Corrected
	@10	@100	@1000	@10	@100	@1000	@10	@100	1000	Time	Time
BM25	0.5086	0.5142	0.5143	0.5104	0.5559	0.6068	0.1242	0.1924	0.1976	3.4733	-
fb_docs = 3, fb_terms = 10											
Bo1	0.4830	0.4900	0.4901	0.5034	0.5484	0.6029	0.1214	0.1902	0.1956	8.1113	1.1646
KL	0.4838	0.4907	0.4908	0.5012	0.5460	0.6014	0.1209	0.1891	0.1945	8.0029	1.0563
RM3	0.4540	0.4609	0.4609	0.4993	0.5490	0.6019	0.1152	0.1860	0.1912	8.0551	1.1085
Axiomatic	0.5086	0.5142	0.5143	0.5104	0.5559	0.6068	0.1242	0.1924	0.1976	7.7308	0.7842
fb_docs = 10, fb_terms = 40											
Bo1	0.5082	0.5158	0.5159	0.5050	0.5514	0.6067	0.1230	0.1931	0.1984	12.2745	5.3358
KL	0.5033	0.5108	0.5109	0.5013	0.5482	0.6042	0.1213	0.1906	0.1960	11.4169	4.4782
RM3	0.4848	0.4916	0.4917	0.5012	0.5539	0.6056	0.1175	0.1884	0.1935	10.7715	3.8327
Axiomatic	0.5086	0.5142	0.5143	0.5104	0.5559	0.6068	0.1242	0.1924	0.1976	7.2732	0.3345
fb_docs = 50, fb_terms = 50											
Bo1	0.5029	0.5110	0.5111	0.4967	0.5429	0.6037	0.1211	0.1908	0.1965	13.0175	6.4048
KL	0.4880	0.4962	0.4963	0.4945	0.5381	0.5996	0.1175	0.1852	0.1908	11.4017	4.7890
RM3	0.5024	0.5084	0.5084	0.5020	0.5499	0.6044	0.1197	0.1889	0.1941	12.8409	6.2282
Axiomatic	0.5086	0.5142	0.5143	0.5104	0.5559	0.6068	0.1242	0.1924	0.1976	7.4684	0.8557

Table 7: Experiment results for the dataset “antique/test”

A.4 TREC Deep Learning

Metrics	RR(rel=1)			nDCG			AP(rel=1)				Corrected
	@10	@100	@1000	@10	@100	@1000	@10	@100	1000	Time	Time
BM25	0.2271	0.2387	0.2394	0.2868	0.3479	0.3670	0.2285	0.2402	0.2409	87.2624	-
fb_docs = 3, fb_terms = 10											
Bo1	0.2043	0.2175	0.2183	0.2643	0.3303	0.3510	0.2069	0.2199	0.2207	303.7460	129.2211
KL	0.2034	0.2161	0.2168	0.2628	0.3286	0.3493	0.2053	0.2181	0.2189	360.2182	185.6934
RM3	0.2041	0.2174	0.2181	0.2643	0.3318	0.3513	0.2057	0.2190	0.2197	300.8879	126.3631
Axiomatic	0.2271	0.2387	0.2394	0.2868	0.3479	0.3670	0.2285	0.2402	0.2409	224.2310	49.7061
fb_docs = 10, fb_terms = 40											
Bo1	0.1664	0.1811	0.1819	0.2217	0.2964	0.3171	0.1678	0.1825	0.1833	717.8090	551.0606
KL	0.1663	0.1804	0.1812	0.2234	0.2966	0.3165	0.1677	0.1818	0.1825	987.8933	821.1448
RM3	0.2129	0.2264	0.2271	0.2714	0.3400	0.3589	0.2148	0.2283	0.2290	667.9981	501.2497
Axiomatic	0.2271	0.2387	0.2394	0.2868	0.3479	0.3670	0.2285	0.2402	0.2409	219.2020	52.4536
fb_docs = 50, fb_terms = 50											
Bo1	0.1312	0.1480	0.1489	0.1723	0.2586	0.2842	0.1326	0.1494	0.1503	813.8171	646.2700
KL	0.1429	0.1588	0.1597	0.1870	0.2690	0.2942	0.1444	0.1604	0.1613	1134.0844	966.5372
RM3	0.2088	0.2224	0.2230	0.2643	0.3347	0.3539	0.2101	0.2237	0.2244	867.2030	699.6559
Axiomatic	0.2271	0.2387	0.2394	0.2868	0.3479	0.3670	0.2285	0.2402	0.2409	218.6392	51.0920

Table 8: Experiment results for the dataset “trec-deep-learning-docs”

References

[1] Giambattista Amati. Probability models for information retrieval based on divergence from randomness. 2003.

[2] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. Ms marco: A human generated machine reading comprehension dataset. In *InCoCo@NIPS*, 2016.

[3] Donna Harman. Evaluation issues in information retrieval. *Information Processing and Management*, 28(4):439–440, 1992.

[4] Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and Bruce Croft. Antique: A non-factoid question answering benchmark. In *ECIR*, 2020.

[5] Nasreen Jaleel, James Allan, W. Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark Smucker, and Courtney Wade. Umass at trec 2004: Novelty and hard. 01 2004.

[6] Kalervo J"arvelin and Jaana Kek"al"ainen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[7] Paul Kantor and Ellen Voorhees. The trec-5 confusion track. *Information Retrieval*, 2(2-3):165–176, 2000.

[8] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[9] Christina Lioma, Craig Macdonald, Vassilis Plachouras, Jie Peng, Ben He, and Iadh Ounis. University of glasgow at trec 2006: Experiments in terabyte and enterprise tracks with terrier. In *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland, November 14-17, 2006, Special Publication 500-272*. National Institute of Standards and Technology (NIST), 2006. Lioma, C., Macdonald, C., Plachouras, V., Peng, J., He, B., Ounis, I. (2006). University of Glasgow at TREC 2006: Experiments in Terabyte and Enterprise Tracks with Terrier. In Voorhees, E. M., Buckland, L. P. (Eds.), Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland, November 14-17, 2006, Special Publication 500-272, National Institute of Standards and Technology (NIST).

[10] Sean MacAvaney, Craig Macdonald, Charlie Clarke, Benjamin Piwowarski, and Harry Scells. ir_measures. https://github.com/terrierteam/ir_measures, 2021. A Python library for standardized evaluation of information retrieval metrics.

[11] Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. Simplified data wrangling with ir_datasets. In *SIGIR*, 2021.

[12] Craig Macdonald and Nicola Tonellotto. Declarative experimentation ininformation retrieval using pyterrier. In *Proceedings of ICTIR 2020*, 2020.

[13] Dipasree Pal, Mandar Mitra, and Kalyankumar Datta. Query expansion using term distribution and term association, 2013.

[14] Athanasios Papoulis. *Probability, random variables, and stochastic processes*. McGraw Hill Higher Education, Maidenhead, England, 3 edition, March 1991.

[15] Vassilis Plachouras, Ben He, and Iadh Ounis. University of glasgow at trec 2004: Experiments in web, robust, and terabyte tracks with terrier. 2004.

[16] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 4 2021.

[17] Henning Wachsmuth, Shahbaz Syed, and Benno Stein. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 241–251. Association for Computational Linguistics, 2018.

[18] Peilin Yang and Jimmy Lin. *Reproducing and Generalizing Semantic Term Matching in Axiomatic Information Retrieval*, pages 369–381. 04 2019.