Simple Yet Effective Pseudo Relevance Feedback with Rocchio's Technique and Text Classification

by

Yuqi Liu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Data Science

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

With the continuous growth of the Internet and the availability of large-scale collections, assisting users in locating the information they need becomes a necessity. Generally, an information retrieval system will process an input query and provide a list of ranked results. However, this process could be challenging due to the "vocabulary mismatch" issue between input queries and passages. A well-known technique to address this issue is called "query expansion", which reformulates the given query by selecting and adding more relevant terms. Relevance feedback, as a form of query expansion, collects users' opinions on candidate passages and expands query terms from relevant ones. Pseudo relevance feedback assumes that the top documents in initial retrieval are relevant and rebuilds queries without any user interactions.

In this thesis, we will discuss two implementations of pseudo relevance feedback: decades-old Rocchio's Technique and more recent text classification. As the reader might notice, both techniques are not "novel" anymore, e.g., the emergence of Rocchio can even be dated back to the 1960s. They are both proposed and studied before the neural age, where texts are still mostly stored as bag-of-words representations. Today, transformers have been shown to advance information retrieval, and searching with transformer-based dense representations outperforms traditional bag-of-words searching on many challenging and complex ranking tasks.

This motivates us to ask the following three research questions:

• RQ1: Given strong baselines, large labelled datasets, and the emergence of transformers today, does pseudo relevance feedback with Rocchio's Technique still perform effectively with both sparse and dense representations?

• RQ2: Given strong baselines, large labelled datasets, and the emergence of transformers today, does pseudo relevance feedback via text classification still perform effectively with both sparse and dense representations?

• RQ3: Does applying pseudo relevance feedback with text classification on top of Rocchio's Technique results in further improvements?

To answer RQ1, we have implemented Rocchio's Technique with sparse representations based on the Anserini and Pyserini toolkits. Building in a previous implementation of Rocchio's Technique with dense representations in the Pyserini toolkit, we can easily evaluate and compare the impact of Rocchio's Technique on effectiveness with both sparse and dense representations. By applying Rocchio's Technique to MS MARCO Passage and Document TREC Deep Learning topics, we can achieve about a 0.03-0.04 increase in average precision. It's no surprise that Rocchio's Technique outperforms the BM25 baseline, but it's

impressive to find that it is competitive or even superior to RM3, a more common strong baseline, under most circumstances. Hence, we propose to switch to Rocchio's Technique as a more robust and general baseline in future studies.

To our knowledge, pseudo relevance feedback via text classification using both positive and negative labels is not well-studied before our work. To answer RQ2, we have verified the effectiveness of pseudo relevance feedback via text classification with both sparse and dense representations. Three classifiers (LR, SVM, KNN) are trained, and all enhance effectiveness. We also observe that pseudo relevance feedback via text classification with dense representations yields greater improvement than sparse ones. However, when we compare text classification to Rocchio's Technique, we find that Rocchio's Technique is superior to pseudo relevance feedback via text classification under all circumstances.

In RQ3, the success of pseudo relevance feedback via text classification on BM25 + RM3 across four newswire collections in our previous paper motivates us to study the impact of pseudo relevance feedback via text classification on top of another query expansion result, Rocchio's Technique. However, unlike RM3, we could not observe much difference in the two evaluation metrics after applying pseudo relevance feedback via text classification on top of Rocchio's Technique.

This work aims to explore some simple yet effective techniques which might be ignored in light of deep learning transformers. Instead of pursuing "more", we are aiming to find out something "less". We demonstrate the robustness and effectiveness of some "out-of-date" methods in the age of neural networks.

Acknowledgements

First and foremost, I would like to thank my advisor, Professor Jimmy Lin, for giving me the opportunity to work on exciting research projects and to collaborate with so many inspiring researchers. His continuing guidance and support helped me in the writing of this thesis and all other research. I feel really lucky and proud to be his student. Pseudo relevance feedback via text classification is the first research project that led me into the world of IR, and I'm so happy to extend the study on this topic and turn it into this thesis.

I would like to thank the readers of my thesis, Professor Pascal Poupart and Xi He, for reviewing my work.

I would like to thank Xiao Han, Hang Li, Jack Lin, and Xueguang Ma for their discussions and insights that contributed greatly to this thesis.

I would also like to thank my collaborators from the Data Systems Group, Chengcheng Hu, Xinyu Zhang, and Ronak Pradeep, for the projects done outside the scope of this thesis.

Last but not least, I would like to thank my parents for their continuous care and love for all years.

Dedication

This is dedicated to my parents for their unconditional love and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet plays an essential role in our lives. The advent of the Internet has brought not only more interactions among global users but also enormous and continuous growth of text data. Given the availability of large-scale text data, it's very difficult for people to accurately and efficiently locate the information they need. In other words, people need an intelligent information retrieval system that consumes an input query and returns relevant passages in a timely manner.

Retrieval can be very challenging due to the well-known "vocabulary mismatch" problem [9], where search users might express concepts that differ from those in relevant passages. The problem is compounded by a combination of synonymy (multiple words have nearly the same meaning) and polysemy (one word has different meanings). Synonymy can cause a failure in retrieving relevant documents, and polysemy can lead to a retrieval of non-relevant ones; both will reduce the quality of the retrieval results.

Query expansion has long been considered one of the most promising and common approaches in information retrieval to combat the "vocabulary mismatch" issue [4, 49, 22]. It can build new queries by selecting and adding more terms to better capture the users' intents behind the search, thus producing the desired results.

Various studies have been conducted to find the best expansion terms. The most popular and well-studied ones are the following two ways: co-occurrence and relevance feedback. In this thesis, we will mainly focus on the relevance feedback approach, where the information retrieval system expands queries with terms from relevant passages indicated by users.

Pseudo relevance feedback (PRF), also known as blind relevance feedback, assumes the relevance of top k initial retrieval results and performs a second-round retrieval under this

assumption. This approach is commonly used to boost retrieval effectiveness without any inclusion of extended user interaction. As one type of relevance feedback, it could also effectively produce a revised query and alleviate the "vocabulary mismatch" problem in information retrieval tasks.

As the reader notices, pseudo relevance feedback is not that novel and could even be considered a bit "old-fashioned". In fact, one implementation of pseudo relevance feedback, Rocchio's Technique, can date back to the 1960s [40] when Rocchio reported the advance of expanding queries with relevance feedback. Instead of simply relying on the statistics of corpus data, Rocchio took advantage of the relevance judgements of documents provided by users in the initial retrieval process. By combining the given query and feedback, the retrieval effectiveness of the expanded queries was generally superior to the initial ones. Rocchio's Technique immediately brought the attention of researcher Ide [18], and further verified by Salton and Buckley in 1990 [42], and Harman in 1992 [13].

Pseudo relevance feedback via text classification is a more recent implementation of pseudo relevance feedback. A thread of experiments was performed by combining the relevance feedback with classification, e.g., Cormack and Mojdeh applied linear classification on TREC 2019 topic [6], Xu and Akella introduced a Bayesian linear regression algorithm with relevance feedback [50]. Nevertheless, to our knowledge, none of the experiments was done with pseudo judgements. In our previous experiment [12], we presented a technique of training document classifiers from the pseudo labels of the initial ranked lists and reported its success across four newswire collections. In this thesis, our experiment regarding pseudo relevance feedback via text classification can be considered an extension of our previous work on different collections and configurations.

While both techniques were proposed and studied before the domination of neural networks in information retrieval, all experiments were done with bag-of-words document representations, describing the occurrence of words in the document. Without a doubt, pseudo relevance feedback was once a common and promising technique in many information retrieval tasks. However, what will be its role in this neural age? What has been changed in this neural age?

Although the idea of neural networks was proposed in 1943 [31] and the first trainable neural network was demonstrated in 1957 [41], a significant breakthrough did not appear till the computational speeds and GPU got well-established in the 2000s. After that, transformers achieved unprecedented success on multiple information retrieval tasks, and there was a noticeable improvement in how we represent the document text. Instead of a simple representation of the occurrence, like bag-of-words, transformer-based document representation is generated by the corresponding transformer model, e.g., BERT. Many studies

[20, 5, 24] indicated dense retrieval with transformer-based representations is superior to traditional bag-of-words retrieval.

From here, three research questions have been motivated.

1.1 RQ1

Given strong baselines, large labelled datasets, and the emergence of transformers today, does pseudo relevance feedback with Rocchio's Technique still perform effectively with both sparse and dense representations?

1.2 RQ2

Given strong baselines, large labelled datasets, and the emergence of transformers today, does pseudo relevance feedback via text classification still perform effectively with both sparse and dense representations?

1.3 RQ3

Does applying pseudo relevance feedback with text classification on top of Rocchio's Technique results in further improvements?

1.4 Contributions

The contributions of this thesis are summarized below.

- Answer "YES" to RQ1.

We implement Rocchio's Technique with sparse representations as a second-round retrieval in the Anserini environment; thus, it is directly comparable with other query expansion results. Results of Rocchio's Technique significantly improve the initial BM25 on all collections and conditions we experiment with. Besides, compared to other query expansion methods like RM3, Rocchio's Technique shows slightly better or equally competitive results in all cases. Note that compared to the vector-prf technique proposed in Li et al.

[25], we have made two major differences: sparse document representations and pseudo negative labels. Instead of using transformer-based document representations based on a deep learning retriever, we utilize sparse BM25-based document representations. With this implementation, Rocchio's Technique could be applied to more generalized indexes and collections, maintaining about the same level of effectiveness enhancement as they are applied on dense representations. Additionally, unlike the original paper, which simply exclude the information of non-relevant documents, we include the bottom n non-relevant documents in the query expansion step. However, we notice that negative feedback results in a minimal improvement in most of the experiments, so the default Rocchio's Technique we show in this thesis are still the ones without negative feedback.

• Answer "YES" to RQ2.

We have followed the standard retrieval-and-rerank pipeline. A text classifier is trained from top k relevant documents and tail n non-relevant documents of the initial ranked list. The experiment is previously conducted on a range of standard newswire collections in our work [12]. So the focus of this thesis switches to the MS MARCO Passage/Document collections. The impressive gains on bag-of-words BM25 as well as transformer-based baselines on TREC 2019 and TREC 2020 topics verify the robustness and effectiveness of our technique. Besides, it's impressive to view that transformers serve as an aid in the selection of query expansion terms. Classification with transformer-based document representations generally surpass the ones with traditional BM25 document representations.

• Answer "NOT MUCH DIFFERENCE" to RQ3.

For RQ3, incorporating the two techniques doesn't make much difference in terms of effectiveness. The previous two experiments show that Rocchio's Technique is more effective than pseudo relevance feedback via text classification. As pseudo relevance feedback via text classification could be solely applied on top of any candidate list, it would be interesting to see whether we could obtain additional gains by applying classification on top of Rocchio's results. Unfortunately, the combination of Rocchio's Technique with text classification doesn't make much difference in effectiveness. For some of the experiments, the inclusion of classification even ends with a negative impact.

1.5 Thesis Organization

The thesis is organized as follows:

• Chapter 2 reviews the fundamental background and existing works of BM25, relevance feedback, query expansion, and dense retrieval in detail.

• Chapter 3 introduces the architecture of pseudo relevance feedback implementations built with Anserini/Pyserini toolkits for this experiment, including Rocchio's Technique and text classification.

• Chapter 4 introduces datasets, baselines and the evaluation metrics, then describes the experimental setups and implementations.

• Chapter 5 presents experimental results and analysis.

• Chapter 6 concludes this thesis and discusses potential directions for future work.

Chapter 2

Background and Related Work

2.1 Bag-of-words and BM25

The concept of the vector space model in information retrieval is not proposed till 1975 by Salton et al. [44]. In this work, Salton mentions the possibility of viewing queries and documents as vectors in various dimensions of spaces. The vector values could be computed by several term weighting schemes; one of the best-known schemes is tf-idf. As these queries and documents are term-based, they are often referred to as "bag-of-words" (BOW) representations. The documents relevance rankings could be generated by calculating the similarity between documents and queries representations and sorting in descending order.

The term weighting schemes play a vital role in the ranking effectiveness of vector space models. To achieve a better term weighting scheme, many researchers perform a bunch of experiments during the 1980s-1990s [43]. One of the most promising term weighting schemes is OKAPI BM25 [39], where BM stands for "Best Match". The scheme is based on the bag-of-words representations and the assumption that every term in the document is independent of each other. OKAPI BM25 still provides a relatively strong baseline for ranking tasks in research and industry today. However, as it depends on exact matches between query terms and document terms, it unavoidably suffers from the well-known "vocabulary issue". That is when relevance feedback comes into place.

	Team	Run	Type	nDCG@10	P@5	AP
Round 3: 40 topics						
(3a)	covidex	r3.t5_lr	feedback	0.7740	0.8600	0.3333
(3b)	BioinformaticsUA	BioInfo-run1	feedback	0.7715	0.8650	0.3188
(3c)	SFDC	SFDC-fus12-enc23-tf3	automatic	0.6867	0.7800	0.3160
(3d)	covidex	r3.duot5	automatic	0.6626	0.7700	0.2676
(3e)	anserini	r3.fusion2	automatic	0.6100	0.7150	0.2641
(3f)	anserini	r3.fusion1	automatic	0.5359	0.6100	0.2293
	Team	**Run**	Type	nDCG@20	P@20	AP
Round 4: 45 topics						
(4a)	unique_ptr	UPrrf38rrf3-r4	feedback	0.7843	0.8211	0.4681
(4b)	covidex	covidex.r4.duot5.lr	feedback	0.7745	0.7967	0.3846
(4c)	covidex	covidex.r4.d2q.duot5	automatic	0.7219	0.7267	0.3122
(4d)	anserini	r4.fusion2	automatic	0.6089	0.6589	0.3088
(4e)	anserini	r4.fusion1	automatic	0.5244	0.5611	0.2666
Round 5: 50 topics						
(5a)	unique_ptr	UPrrf93-wt-r5	feedback	0.8496	0.8760	0.4718
(5b)	covidex	covidex.r5.2s.lr	feedback	0.8311	0.8460	0.3922
(5c)	covidex	covidex.r5.2s	automatic	0.7457	0.7610	0.3212
(5d)	anserini	r5.fusion2	automatic	0.6007	0.6440	0.2734
(5e)	anserini	r5.fusion1	automatic	0.5313	0.5840	0.2314

Table 2.1: Select TREC-COVID results. Explicit feedback submissions are under team "covidex". Note that the metrics used in round 3 are different from the other two

2.2 Relevance feedback

Relevance feedback is one of the most popular techniques for information retrieval that can extract feedback from the initial results and improve the final results. Three types of relevance feedback are differentiated by user involvement: explicit feedback, implicit feedback, and pseudo relevance feedback.

Explicit feedback is usually gained from the user, who manually judges the relevance level of documents. A good example of explicit feedback could be found in the TREC COVID challenge with COVID-19 Open Research Dataset (CORD-19) [47]. It organizes five evaluation rounds, where each round includes all topics from earlier rounds and some

additional new topics. After each round, the sponsor National Institute for Standards and Technology (NIST) provides manually annotated judgments from that round, which could be considered feedback for further model evolution.

Here the explicit relevance feedback via classification proposed by Han and Liu et al. [11] is proved to be highly effective. More detailed results can be found in Table 1 under the team name "covidex". Here "feedback" type represents that the submission uses the feedback provided by NIST while "automatic" submissions don't. In the table, we could see in Round 3 that explicit feedback via classification Row (3a) reports the best score over all runs. In rounds 4 and 5, the "covidex" runs are also the second-best submissions Row (4b, 5b). From here, we can demonstrate that explicit relevant feedback via classification could lead to a consistent improvement even on the state-of-arts.

Implicit feedback usually comes from users' behaviours, e.g., browsing time and scrolling actions. This data is usually obtained without the notice of the user, so compared to the explicit, this method consumes less human effort. Because of this advantage, implicit feedback is widely used in real-world recommendation systems, described in [16] and [37]

In this study, we mostly focus on pseudo relevance feedback (PRF), which completely eliminates user interactions compared to the other two feedback methods. It has been found to boost effectiveness on many TREC ad hoc tasks. PRF automates the work of gathering users' opinions to improve second-round retrieval results. PRF assumes that top k documents of the initial list returned by the initial query are relevant and tail n documents are non-relevant. Two implementations could be applied to this feedback: training a classifier or expanding the original query with representations of these pseudo relevance labels. More details of these two pipelines can be found in Sections 3.1 and 3.2.

The drawbacks of PRF are also quite obvious. If the quality of the initial list is relatively low, e.g., If the majority of the first k documents are not relevant to the query, the effectiveness of PRF will be very limited. Even worse, if none of the selected top documents are relevant at all, PRF could even reduce the effectiveness. Besides, it's worth noting that with reranking, we shuffle the initial list by bringing more relevant documents into a higher rank and dragging non-relevant documents into a lower rank. Thus, if relevant documents do not exist in the initial list, the PRF reranker won't help at all in this case.

2.3 Rocchio's Technique

As an implementation of relevance feedback, the Rocchio's Technique could also be divided into three categories: manual, automatic, and interactive. Rocchio's Technique, a classic

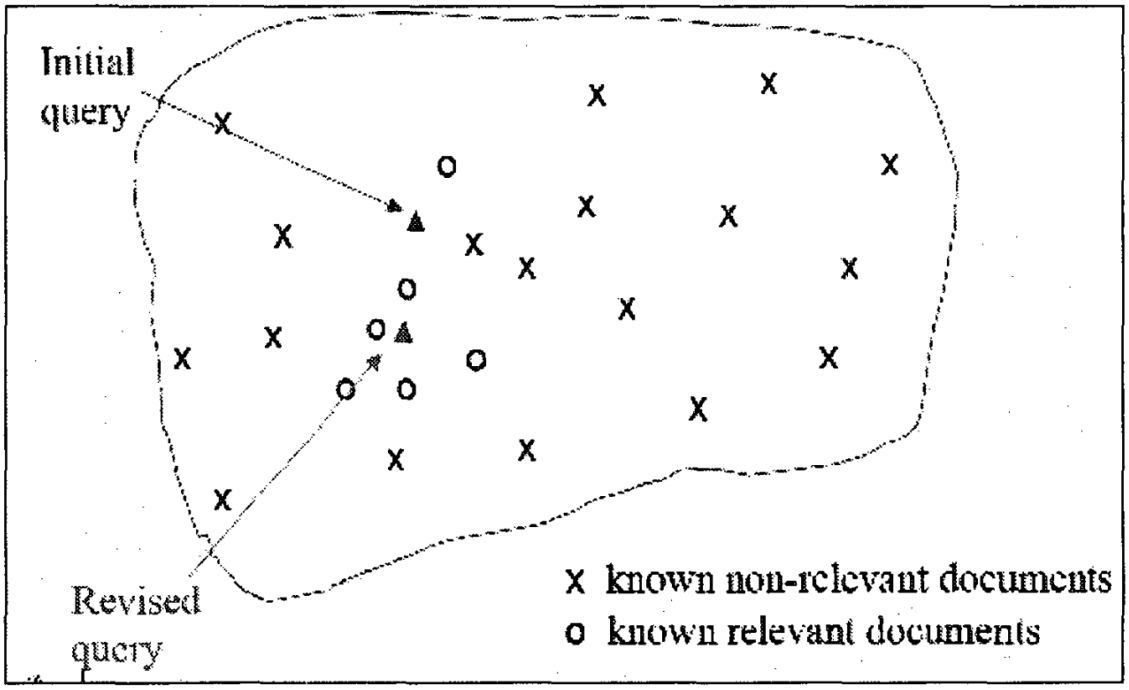

Figure 2.1: Model illustration for Rocchio's Technique. The Diagram is taken from Katta et al. [21]

query expansion model, is developed by Cornell University in the 1970s from the SMART Information Retrieval System [2]. Rocchio's Technique takes advantage of the feedback from the initial ranking and builds a revised query based on that. So, the revised query vectors are closer to the relevant document vectors and away from the non-relevant ones. The Rocchio's Technique model illustration generated by Katta [21] perfectly displays this refinement in Figure 2.1. Since the revised query is computed by the distance to the relevance labels, we could easily associate it with KNN classification, which also relies on the distance features.

The algorithm for Rocchio's Technique can be found below:

$$Q_m = \alpha Q_0 + \beta \frac{1}{|D_r|} \sum_{d_j \in D_r} d_j - \gamma \frac{1}{|D_{nr}|} \sum_{d_j \in D_{nr}} d_j \tag{2.1}$$

Here Q_m represents the new query while Q_0 is the original one. α, β, γ are the weights manually assigned to the original query Q_0, positive document representations, and negative document representations. D_r is the set of relevant documents, while D_{nr} is the set of the non-relevant documents. These documents are either annotated by human or assumed by pseudo relevance feedback.

As Rocchio's Technique largely reduces the demand for user interaction and training collection size, many researchers incorporate the Rocchio's Technique with more cutting-edge transformers to further optimize its effectiveness. For example, Miao incorporates proximity information into Rocchio's Technique [32]; Li et.al [26] integrates Rocchio's Technique directly with emergent deep language models [26].

As a simple query expansion technique, Rocchio's Technique perfectly matches the objective of our work: simple yet effective. Compared to the complicated parameter optimization process of the neural networks, the parameters for Rocchio's Technique are much more explainable and easily tuned.

2.4 Dense Retrieval with Bi-encoder

Though relevance feedback is a promising approach to resolve the "vocabulary mismatch" problem, it's still performing an exact match between the updated query and document representations. To fundamentally alleviate the "vocabulary mismatch" issue, dense representations are proposed to directly capture the semantic meaning of queries and documents with the aid of transformers. The technique of ranking relevant documents by their semantic match to queries is called dense retrieval.

The idea of dense representation at first is not proposed to solve the ranking task; instead, this concept named word embedding is used to tackle the word analogy task [33]. Many researchers extend the idea of word embedding to sentence embedding with various encoder models. One of the most simple yet efficient classes of encoder model, Bi-encoder, is introduced by Humeau et al. [17] for the sentences similarity task. Bi-encoder generates one representation for each of the two inputs given, where their similarity is computed by the inner product or cosine similarity between the two representations. On the other hand, Cross-encoder concatenates the two inputs and sends the concatenated input to transformers, producing a probability of similarity in the range of (0,1). Transformer-based representation via Bi-encoder model isn't applied to information retrieval till late 2010s by Lee et al. [24] and Guu et al. [10]. A nice overview of the development of Bi-encoder in ranking is offered by Lin et al. [28]. This is considered a breakthrough in information

retrieval, and many studies are conducted to verify the superiority of Bi-encoder dense retrieval to traditional sparse retrieval [24, 20, 5].

Pulling all the backgrounds together: given strong baselines and dense representations provided by Bi-encoder dense retrieval, we wonder would the two implementations of PRF still perform well on a number of modern test collections.

Chapter 3

Retrieval Architecture

3.1 Architecture for Rocchio's Technique

Our architecture for Rocchio's Technique is motivated by the vector-based pseudo relevance feedback architecture proposed by Li et al. [26]. In this architecture, each query representation q is firstly aggregated with top k feedback documents from the initial ranked list, and then used to perform a second-round retrieval to obtain the final ranked list.

Figure 3.1 depicts the detailed steps of our architecture for Rocchio's Technique. As shown in the first step ❶, the original query q is firstly encoded into a sparse/dense query representation by a sparse BM25-based or dense transformer-based encoder. Besides the dense transformer like ANCE mentioned in Li et al. [26], we also conducted query encoding with two other stronger dense retrievers: TCT ColBERT V2 HN+ [52] and DistilBERT KD TASB [15]. In the next stage, the query representation is sent to the corresponding retriever for first-round retrieval ❷, generating the initial ranked list H ❸. The top k document representations of the initial ranked list are used as pseudo relevance judgements to update the query representation via Rocchio's Technique ❹ ❺. More details about the background, algorithm and implementation of Rocchio's Technique are stated in Section 2.3. In our experiments with negative relevance feedback, we expand queries with both top k document representations and bottom n document representations in steps ❹ ❺. In the end, the updated query representation is sent back to the retriever again for subsequent retrieval ❻ to produce the final ranked list H' ❼. To investigate the improvement brought by Rocchio's Technique, we assess the statistical significance of metric differences between initial ranked list H and final ranked list H' using a paired two-tailed t-test.

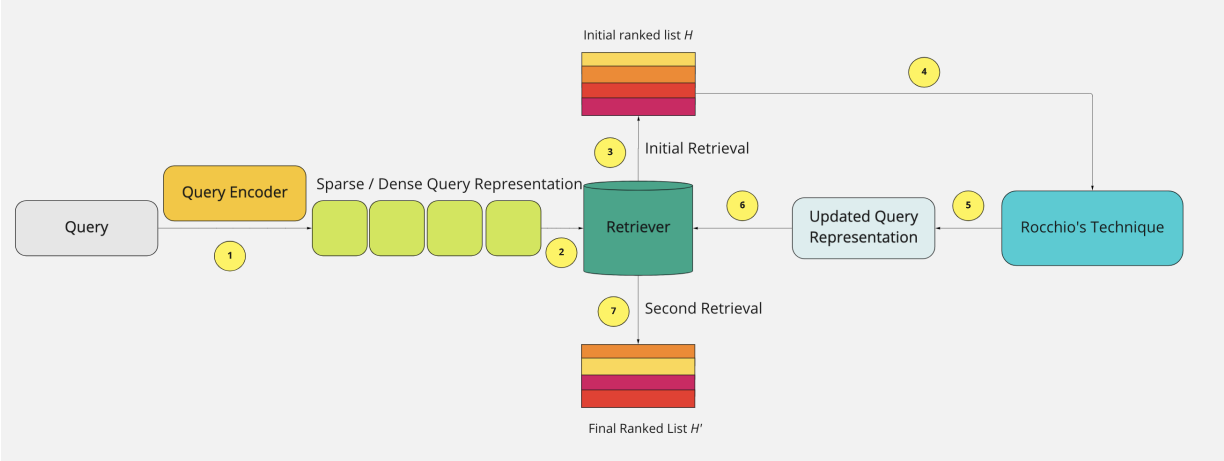

Figure 3.1: Architecture for Rocchio's Technique.

3.2 Architecture for PRF via text classification

In this pseudo relevance feedback implementation, we adopt the retrieval-and-rerank architecture proposed by Nogueira and Cho [36]. The retrieval-and-rerank architecture, a commonly used approach in information retrieval, produces an initial list and reranks it with a more computationally-intensive method. In order to lower latency, the retrieval-and-rerank architecture only sends promising candidates to the second reranking stage, which could be viewed as a trade-off between efficiency and effectiveness. As long as the candidates are of good quality, we can reduce computational costs without sacrificing much effectiveness.

Figure 3.2 describes the detailed steps of our proposed architecture for pseudo relevance feedback via text classification. Similar to the first retrieval round in Rocchio's Technique, queries are encoded into sparse/dense representations in step ❶ and used to generate an initial ranked list H for each query in step ❷,❸. By pseudo relevance feedback, we assume the top k hits of this ranked list are relevant while the last n hits are non-relevant. These positive and negative pseudo document representations are then fed into the classifiers in the next stage ❹, yielding the reranked list H' ❺. It's worth noting that trained classifiers are applied to all 1000 hits of the initial ranked list H, shuffling their ranks to build the reranked list H'. In the last step ❻, we rerank the base run H using new document scores

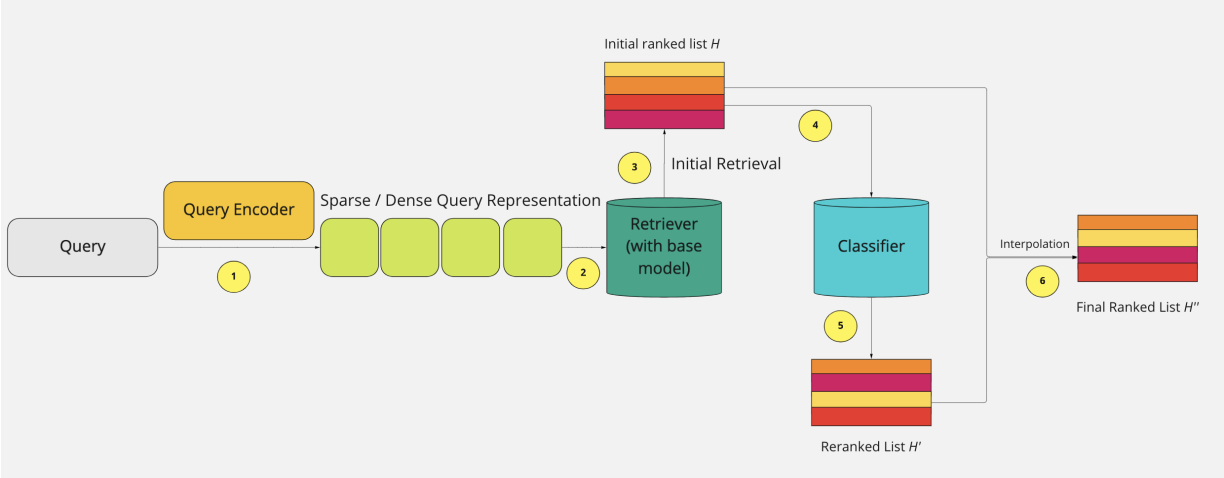

Figure 3.2: Architecture for PRF via text classification.

and generate the final reranked list H'' for our final evaluation. The new document scores here are comprised of the interpolation of the normalized scores from the initial ranked list H and the normalized classifier scores from the reranked list H'.

The equation for new document score could be found below:

$$\text{Document}_{score} = (1 - \alpha) \cdot \text{Retrieval}_{score} + \alpha \cdot \text{Classification}_{score} \tag{3.1}$$

3.3 Architecture for PRF with a combination of text classification and Rocchio's Technique

Multi-stage ranking architecture has been greatly exploited and validated in recent studies. One application is to apply sequence-to-sequence models to a multi-stage architecture proposed by Pradeep et al. [38], whose result is at or comparable to the state-of-art in various TREC ranking tasks. In our experiment, we consider Rocchio's Technique the first-round ranking stage and pseudo relevance feedback via text classification as the second-round ranking stage.

While we are still following most of the steps in Figure 3.2, the query encoding and first retrieval steps are omitted. Instead of training classifiers with the initial ranked list produced by the base retriever, we feed the final ranked list of Rocchio's Technique to the classifiers. With all following steps remaining the same, the trained classifier for each topic is applied to all documents in the base run for that topic. In this case, the base run is the final ranked list from Rocchio's Technique.

Chapter 4

Experiment Setup

4.1 Data

For Rocchio's Technique, we perform experiments on two MS MARCO collections: Passage and Document [35]. MS MARCO Passage, a widely used training collection for IR tasks, was released by Microsoft in 2018 to support research on passage ranking and question answering tasks. It contains 503k queries in the training set, where each query is associated with at least one relevant passage from the 8.8 million passages corpus.

Additionally, our experiment use two evaluation topic sets: TREC 2019 and TREC 2020 Deep Learning Track [8, 7]. The official development set contains 6980 queries, with every single query associated with one human-annotated relevance label on average, which is not suitable for evaluating PRF. Instead, TREC 2019 and TREC 2020 Deep Learning Tracks have more than one positive judgment selected during the pooling and judging process.

The goal of the MS MARCO Document task, similar to the MS MARCO Passage task, is to rank the relevant documents from the 3.2 million document corpus. However, given the length of documents, we apply the segmentation technique to divide each document into several smaller chunks and treat each as a single unit in the subsequent retrieval and reranking stages. Specifically, a sliding window strategy is applied in the segmentation, where a 3-sentence sliding window with a 1-sentence stride is applied for each document. The final aggregation uses the maxP technique, where each document score is decided by the highest score of all segmented chunks for that document.

Our technique has two query/document representations: sparse BM25-based representation and dense transformer-based representation. With BM25-based representation, we

	Passage				Document		
	Train	Dev	DL19	DL20	Dev	DL19	DL20
# of documents	8.8 million				3.2 million		
# of queries	502,939	6980	43	200	5193	43	200

Table 4.1: MS MARCO Dataset Statistics

perform both full and segmented retrieval. On the other hand, only segmented retrieval is conducted on experiments with transformer-based representation to satisfy the length limitations of transformer.

To answer the second research question, in previous work, we conduct PRF via text classification on four newswire collections: Robust04, Robust05, Core17 and Core18 [11] and confirm its outstanding performance with the sparse query/document representations. In this study, we extend the collections to MS MARCO Passage/Document and the representations to both BM25-based sparse and transformer-based dense ones.

4.2 Environment

All three experiments are conducted with Anserini [51], a Lucene-based information retrieval toolkit, and a Python toolkit Pyserini [27]. Pyserini supports retrieval for sparse representations integrated from Anserini toolkit and dense representations integrated from Facebook's Faiss library [19]. All sparse/dense representations are converted into unit vectors with L2 normalization before feeding into a pseudo relevance feedback implementation.

To answer RQ2, we train three classifiers with both sparse and dense representations: linear regression (LR), support vector machines (SVM) with linear kernel, and k nearest neighbours (KNN). All classifiers are imported from the scikit-learn package [3]. The reason for us to explore k nearest neighbours with dense representations is to compare its results to Rocchio's Technique. Like the KNN classifier, Rocchio's Technique also groups by the distance (score) of document representations; thus, it could be considered a variant KNN.

In all experiments, the initial retrieval produces a standard top 1000 hits per topic. No matter how the document representations are created (sparse or dense), the top k and in some cases bottom n documents from the base run are used for further text classification/ query expansion. Even if the first initial retrieval takes advantage of the segmentation

technique, we only apply PRF via text classification on the full document list aggregated by the maxP technique. Given that all training labels for classifiers come from our first retrieval, our technique does not produce any new documents other than the initially retrieved candidates. It's worth noting that all effectiveness improvements come from shuffling the list alone.

4.3 Baseline

All initial base candidate lists for Rocchio's Technique are produced by the retrieval results of base models: BM25 or transformers. Besides, two configurations are presented for BM25 on MS MARCO Passage, BM25 with default parameters ($k1$=0.9, b=0.4) and BM25 w/ doc2query-T5. The following three transformers are used for encoding and retrieving in the MS MARCO Passage ranking task: ANCE, TCT-ColBERT V2 HN+, and DistilBERT KD TASB.

• ANCE: Approximate Nearest-neighbour Negative Contrastive Learning (ANCE) [48], a first-stage dense retriever that takes advantage of global negative in the corpus to update the ANN index.

• TCT-ColBERT V2 HN+: A modification of TCT-ColBERT V1 by utilizing a bi-encoder teacher-to-student setup and efficiently adding hard negatives to knowledge distillation [29].

• DistilBERT KD TASB : An improvement on BERT with knowledge distillation model [15] which proposes the usage of Balanced Topic Aware Sampling (TAS-Balanced) to compose dense retrieval training batches.

As we are only utilizing some existing bag-of-words and transformers baselines in Pyserini, for some models e.g., DistilBERT KD TASB, we currently don't have that implemented for MS MARCO Document ranking.

To answer RQ2, we have also employed PRF via text classification on the same base models we utilize in RQ1. Besides, PRF via text classification is also applied to BM25 + RM3 condition to test its robustness upon a stronger baseline. For RQ3, we perform a cross combination between sparse/dense Rocchio's Technique with sparse/dense PRF via text classification. The baseline for this experiment is the final ranked list of Rocchio's Technique.

4.4 Parameter Tuning

4.4.1 Rocchio's Technique

By the algorithm of Rocchio's Technique, we are supposed to compute its score by the BM25 weights of the document representations. Because Anserini uses Lucene boosts for the second-round query, which means it handles BM25 weighting internally, we end up choosing Boolean weighting when aggregating multiple doc vectors. If one of the terms appears in any of the doc vectors, it's considered 1; otherwise, 0.

We also examine the number of relevant documents and terms used for query expansion and the interpolation weight assigned to original query/positive judgments/negative judgments. Empirically, including all terms in documents could reduce the accuracy as many unimportant terms are added to the query, which can hurt the accuracy of search keywords. Additionally, since there's no guarantee that all chosen top documents are relevant, and bottom documents are non-relevant, including too many documents might also reduce the retrieval performance.

The number of the terms used for each document is limited to 256 in Li et al. [25] in order to align with the length limitations of transformers. But here, we adopt a much smaller term and document size to better compare with other query expansions technique like RM3. Specifically, we set feedback term size to 10 and feedback document size to 10 across all experiments without further tuning.

For interpolation weight, we considered three following settings for our implementation:

• Parameters from Li et al. [25] where dense retriever is integrated with Rocchio's Technique. Note that negative non-relevant labels are not utilized here. (α=0.4, β=0.6, γ=0)

• Parameters from the default RM3 reranker [23]. Note that negative non-relevant labels are not utilized here. (α=0.5, β=0.5, γ=0)

• Parameters from the Manning et al. textbook [45] (α=1, β=0.75, γ=0.15)

We end up choosing the parameters from Manning et al. textbook [23] as our default parameter in all following experiments. By comparing the results with negative labels (γ=0.15) and ones without negative labels (γ=0) on the MS MARCO collection, we note that the improvement brought by negative is minimal and not stable at all. Hence, we only report the results without negative labels in all main MS MARCO experiments.

4.4.2 PRF via text classification

For pseudo relevance feedback via text classification, there are only three parameters: k, the number of the top relevant documents used for classifier training; n, the number of the bottom non-relevant documents used for classifier training; α, the interpolation weight of initial BM25 scores. Note that the parameter tuning of PRF is much simpler than a deep learning transformer like BERT, which gives an advantage to researchers who have less experience with parameter optimization. The hyperparameter for k is set to 10 (a common value in pseudo relevance feedback), and n is set to 100 (based on the 10:1 ratio for negative: positive labels) to avoid overfitting. The interpolation weight for PRF via text classification is tuned from 0.0 to 1.0 within 0.1 increment step via 10-fold cross-validation on MS MARCO Passage development set; this ends up as α equals 0.5.

Chapter 5

Results and Analysis

We present experimental results for research questions 1, 2, and 3 in this chapter. As a high-level summary, we gain positive answers for both research questions 1 and 2. Both Rocchio's Technique and PRF via text classification technique perform well even with transformer-derived representations. For RQ3, adding PRF via text classification on top of Rocchio's Technique doesn't bring any substantial gains in terms of effectiveness.

5.1 Research Question 1

5.1.1 Rocchio's Technique on Passage ranking

Experimental results for MS MARCO Passage ranking are shown in Table 5.1. We report the evaluation metrics MAP and nDCG at rank cut-off 10 for both TREC topics. Row (1a) serves as a bag-of-words BM25 baseline for further query expansion where "default" refers to the Anserini (system-wide) default setting for MS MARCO Passage, $k1$=0.9, b=0.4. We list Row (1b) as a reference for RM3 query expansion applied to the BM25 base run; Row (1c) is the result of our proposed Rocchio's Technique. Rows (2a–2c) are organized in a similar manner, except that they take advantage of the doc2query-T5 expansion model [30] with BM25 baseline. Thanks to the Pyserini "two-click" reproduction matrix, which our work conributes to, a simple copy and paste can easily reproduce all base runs and expansion results. [1]

[1] https://castorini.github.io/pyserini/2cr/msmarco-v1-passage.html

	Base Model	Expansion	TREC 2019		TREC 2020	
			MAP	nDCG@10	MAP	nDCG@10
(1a)	BM25 (default)	-	0.3013	0.5058	0.2856	0.4796
(1b)	BM25 (default)	RM3	0.3390***	0.5180	0.3019	0.4821
(1c)	BM25 (default)	Rocchio	0.3474***	0.5275	0.3102	0.4893
(2a)	BM25 (default) w/ doc2query-T5	-	0.4034	0.6417	0.4074	0.6187
(2b)	BM25 (default) w/ doc2query-T5	RM3	0.4485***	0.6548	0.4295	0.6172
(2c)	BM25 (default) w/ doc2query-T5	Rocchio	0.4469***	0.6538	0.4246	0.6102
(3a)	ANCE	-	0.3710	0.6452	0.4076	0.6458
(3b)	ANCE	Rocchio	0.4211***	0.6539	0.4315***	0.6471
(4a)	TCT-ColBERT V2 HN+	-	0.4469	0.7204	0.4754	0.6882
(4b)	TCT-ColBERT V2 HN+	Rocchio	0.4883	0.7111	0.4860	0.6804
(5a)	DistillBERT KD TASB	-	0.4590	0.7210	0.4698	0.6854
(5b)	DistillBERT KD TASB	Rocchio	0.4974	0.7231	0.4879	0.7083

Table 5.1: Experimental results on the MS MARCO Passage collection for Rocchio's Technique on TREC 2019 and TREC 2020 Deep Learning Tracks. Superscripts *** on the conditions indicate significant improvements over the baseline, based on paired t-tests ($p < 0.01$).

Besides the BM25 base model, we also report the results for three dense retrievers: ANCE, TCT-ColBERT V2 HN+ and DistilBERT KD TASB. The base results for the three dense retrievers are shown on Rows (3a, 4a, 5a), while ranked results for Rocchio's Technique are shown on Rows (3b, 4b, 5b). In this case, we avoid displaying RM3 expansion because RM3 isn't mathematically reasonable for dense representations. As a typical "sparse" PRF approach, RM3 counts the occurrence of terms in the feedback of document representations; nevertheless, the dense representations are not countable.

Obviously, by comparing the BM25 baselines with our proposed Rocchio's Technique runs, we see that Rocchio's Technique yields a substantial improvement on MAP across both TREC topics, especially TREC 2019. Through a comparison between Rows (1b–1c) to (2b–2c), we can describe that our Rocchio's Technique beats the well-studied RM3 expansion on the BM25 (default) and holds competitive scores on the BM25 (default) w/ doc2query-T5. This is quite impressive that Rocchio's Technique, one of the oldest expansion techniques, could still be considered robust and effective, just like other more popular expansions technique.

When it comes to transformers, we note that transformers help Rocchio in the selection

	Base Model	Expansion	TREC 2019			TREC 2020		
			MAP	nDCG@10	R@100	MAP	nDCG@10	R@100
			doc (segmented)					
(1a)	BM25 (default)	-	0.2449	0.5302	0.3840	0.3586	0.5281	0.5823
(1b)	BM25 (default)	RM3	0.2884***	0.5764	0.4355	0.3774	0.5179	0.6224
(1c)	BM25 (default)	Rocchio	0.2889***	0.5570	0.4415	0.3830	0.5226	0.6291
(1d)	ANCE	-	0.2083	0.6327	0.3162	0.3853	0.6325	0.5440
(1e)	ANCE	Rocchio	0.2351***	0.6592	0.3358	0.3962	0.6279	0.5619
(1f)	TCT-ColBERT V2	-	0.2684	0.6593	0.3854	0.3914	0.6094	0.5964
(1g)	TCT-ColBERT V2	Rocchio	0.2946***	0.6594	0.4241	0.4292	0.6391	0.6253
			doc (full)					
(2a)	BM25 (default)	-	0.2434	0.5176	0.3949	0.3793	0.5286	0.6110
(2b)	BM25 (default)	RM3	0.2774***	0.5170	0.4189	0.4014	0.5225	0.6414
(2c)	BM25 (default)	Rocchio	0.2811***	0.5256	0.4261	0.4089	0.5192	0.6425

Table 5.2: Experimental results on the MS MARCO Document collection for Rocchio's Technique on TREC 2019 and TREC 2020 Deep Learning Tracks. Superscripts *** on the conditions indicate significant improvements over the baseline, based on paired t-tests ($p < 0.01$).

of expansion terms by producing a better ranked results. The application of Rocchio's Technique on the ANCE base model leads to statistically significant ($p<0.01$) gains on both TREC topics. The improvements with other dense retrievers are still impressive but not significant.

In summary, on the MS MARCO Passage TREC topics, starting from both sparse and dense baselines, Rocchio's Technique improves average precision by 0.03-0.05. This means we can gain about 10% higher average precision by simply adding an expansion technique. Even compared to RM3, Rocchio's Technique achieves competitive or even better effectiveness upon the BM25 baseline. In addition, Rocchio's Technique has the advantage of being compatible with deep learning transformers.

5.1.2 Rocchio's Technique on Document ranking

We also conduct experiments on the MS MARCO Document collection to examine the robustness and generality of Rocchio's Technique. Table 5.2 presents more details of results,

where segmented and full document runs are displayed in two separate blocks. For segmented runs, we retrieve the top 10,000 different segments and use the "maxP" technique to aggregate the final 1000 hits. On the other hand, for full document runs, we straightforwardly retrieve results with top 1000 hits. Rows (1a) and (2a) report the "default" BM25 baseline ($k1$=0.9, b=0.4) while Rows (1b–1c) and (2b–2c) display the expansion results. For the segmented condition, as the implementation of DistillBERT KD TASB model is missing for Document ranking, we only report the results with ANCE and TCT-ColBERT V2 HN+ transformers on Rows (1d–1g). According to the official evaluation metrics of the MS MARCO Document, the MAP and recall we report here are both at rank cut-off 100.

On the TREC 2019 Deep Learning Track, we can observe that the MAP of Rocchio's Technique is significantly better ($p < 0.01$) than the original baseline and is higher than RM3 under all circumstances. With the TREC 2020 Deep Learning Track, though the enhancements are no longer significant, it still beats RM3 in MAP and recall@100 across all base models.

For a final comparison, examining the doc (full) and doc (segmented) conditions, it's hard to tell which condition exhibits higher effectiveness than the other. Interestingly, the enhancements in effectiveness results from the query expansion technique are not affected when changing from the full corpus to the segmented one.

To sum up, the great performance of Rocchio's Technique in all conditions compared with the BM25 baselines and strong RM3 expansion results confirm its robustness and effectiveness. BM25 + RM3, one of the most stable and widely used additional baselines, still fails to surpass Rocchio's Technique in MS MARCO Document ranking, a more challenging ranking task.

5.1.3 Negative Feedback

It's worth noting that all Rocchio's Technique results we gain above are built by the top 10 documents as pseudo relevance feedback. We wonder whether including negative feedback can yield more gains; the detailed results are shown in Table 5.3. To investigate the impact of negative feedback on Rocchio's Technique in general, we experiment on two different collections: MS MARCO Passage and MS MARCO Document. The boost between Rows (1) and (2) comes solely from Rocchio's Technique with positive feedback, while the gap between Rows (2) and (3) is caused by the involvement of negative feedback. In a comparison of Row (2) to Row (3), we can note that including negative feedback reduces the effectiveness in most runs and gives minimal improvements in the rest of the experiments. One possible explanation is that the most frequent terms in the tail n

		MSMARCO Passage		MSMARCO Document	
	Condition	TREC2019 AP	TREC2020 AP	TREC2019 AP	TREC2020 AP
(1)	BM25	0.3013	0.2856	0.2434	0.3793
(2)	BM25 + Rocchio	0.3474***	0.3102	0.2811***	0.4089
(3)	BM25 + Rocchio-neg	0.3463***	0.3096	0.2813***	0.4096

Table 5.3: Effectiveness of Rocchio's Technique with/without negative feedback on collections MS MARCO Passage/Document TREC 2019 Deep Learning Track and TREC 2020 Deep Learning Track. Superscripts *** on the Rocchio's Technique conditions indicate significant improvements over the baseline, based on paired t-tests ($p < 0.01$).

documents are still associated with the query topics. Reductions in their weights could have a bad impact on the performance of second-round retrieval. Hence, we decide to consider Rocchio's Technique with only positive feedback as our default setting in this thesis.

5.2 Research Question 2

5.2.1 PRF via text classification on Passage ranking

Since we have observed the great performance of PRF via text classification on collections like Robust04, Robust05, Core17, and Core18 in our previous work [11], the focus of this thesis will be MS MARCO ranking tasks instead. Table 5.4 contains our main results for PRF via text classification on MS MARCO Passage ranking. The results are split into four sections according to their base models. Besides traditional BM25 retrievers, we also include three strong dense retrievers to study the potential of our proposed PRF via text classification with dense retrieval. Pyserini provides the capability to easily encode the query text by a pre-generated deep learning encoder and access the pre-computed document representations stored in indexes.

Rows (b–d) are the results of classifications based on Row (a), where "LR", "SVM" and "KNN" refers to the classifier applied to the document representations. KNN classifier could be fairly compared to Rocchio's Technique as they both utilize pseudo relevance feedback over the initial ranked list and weight based on the distance of their document representations.

	Base Model	Expansion	Classifier	TREC 2019			TREC 2020		
				MAP	nDCG@10	R@1k	MAP	nDCG@10	R@1k
(1a)	BM25	-	-	0.3013	0.5058	0.7501	0.2856	0.4796	0.7863
(1b)	BM25	-	LR	0.3135	0.5180	0.7501	0.2887	0.5023	0.7863
(1c)	BM25	-	SVM	0.3179	0.5244	0.7501	0.2974	0.5085	0.7863
(1d)	BM25	-	KNN	0.3119	0.5105	0.7501	0.2927	0.4969	0.7863
(2a)	ANCE	-	-	0.371	0.5540	0.7554	0.4076	0.5679	0.7764
(2b)	ANCE	-	LR	0.3841	0.5736	0.7554	0.4054	0.5831	0.7764
(2c)	ANCE	-	SVM	0.381	0.5724	0.7554	0.4109	0.5855	0.7764
(2d)	ANCE	-	KNN	0.3969***	0.5746	0.7554	0.4173	0.5897	0.7764
(3a)	TCT-ColBERT V2 HN+	-	-	0.4469	0.6318	0.8261	0.4754	0.6206	0.8429
(3b)	TCT-ColBERT V2 HN+	-	LR	0.4744***	0.6582	0.8261	0.4834	0.6349	0.8429
(3c)	TCT-ColBERT V2 HN+	-	SVM	0.4672***	0.6571	0.8261	0.4755	0.6288	0.8429
(3d)	TCT-ColBERT V2 HN+	-	KNN	0.4681	0.6531	0.8261	0.4741	0.6179	0.8429
(4a)	DistillBERT KD TASB	-	-	0.4590	0.6360	0.8406	0.4698	0.6346	0.8727
(4b)	DistillBERT KD TASB	-	LR	0.4866***	0.6672	0.8406	0.4853	0.6533	0.8727
(4c)	DistillBERT KD TASB	-	SVM	0.4825	0.6643	0.8406	0.4795	0.6492	0.8727
(4d)	DistillBERT KD TASB	-	KNN	0.4997***	0.6660	0.8406	0.4825	0.6553	0.8727

Table 5.4: Experimental results on the MS MARCO Passage collection for PRF via text classification on TREC 2019 and TREC 2020 Deep Learning Tracks. Superscripts *** on the conditions indicate significant improvements over the baseline, based on paired t-tests ($p < 0.01$).

We can observe from Rows (1a–1d) that the PRF via text classification improves MAP under all conditions though the improvement is not considered statistically significant by our t-tests. Meanwhile, the improvement of classification on TREC 2020 is more modest compared to the ones on TREC 2019. Comparing BM25-derived with transformer-derived results, we can easily find out that though dense retrievers provide a relatively high baseline, PRF via text classification appears to exhibit stronger reranking power with them. This conclusion matches the finding demonstrated by Naseri et al. [34] that transformers could serve as an aid in the selection of query expansion terms and even improve the pseudo relevance feedback model. One reasonable explanation is that as dense retrievals are superior to BM25 retrieval, it could provide a better initial list to our reranking classifier. A boost in the quality of the candidate list could largely benefit the performance of text classification reranking.

We also explore the improvement of PRF via text classification on BM25 + RM3 baselines, which could be viewed in Rows (1a–1d) of Table 5.6. As RM3 also utilizes PRF for query expansion, we are not surprised to observe PRF via text classification, as a second-round PRF technique barely squeezes additional effectiveness from the reranked list. This differs from our previous observation [11] where PRF via text classification

	Base Model	Expansion	Classifier	TREC 2019			TREC 2020		
				MAP	nDCG	R@100	MAP	nDCG	R@100
(1a)	BM25	-	-	0.2434	0.5176	0.3949	0.3793	0.5286	0.6110
(1b)	BM25	-	LR	0.2809***	0.5566	0.4315	0.4090	0.5641	0.6469
(1c)	BM25	-	SVM	0.2748***	0.5285	0.4286	0.4054	0.5365	0.6391
(1d)	BM25	-	KNN	0.2539	0.5209	0.4018	0.3781	0.5106	0.6110
(2a)	ANCE	-	-	0.2083	0.6327	0.3162	0.3853	0.5440	0.6325
(2b)	ANCE	-	LR	0.2417***	0.6405	0.3660	0.3826	0.6312	0.5771
(2c)	ANCE	-	SVM	0.2444***	0.6521	0.3625	0.3899	0.6193	0.5746
(2d)	ANCE	-	KNN	0.2315	0.6266	0.3386	0.3936	0.6294	0.5719
(3a)	TCT-ColBERT V2 HN+	-	-	0.2684	0.6593	0.3854	0.3914	0.6094	0.5964
(3b)	TCT-ColBERT V2 HN+	-	LR	0.2992***	0.6716	0.4178	0.4200	0.6184	0.6382
(3c)	TCT-ColBERT V2 HN+	-	SVM	0.2968***	0.6714	0.4158	0.4212	0.6182	0.6407
(3d)	TCT-ColBERT V2 HN+	-	KNN	0.2817	0.6559	0.4058	0.4012	0.6106	0.6245

Table 5.5: Experimental results on the MS MARCO Document collection for PRF via text classification on TREC 2019 and TREC 2020 Deep Learning Tracks. Superscripts *** on the Rocchio conditions indicate significant improvements over the baseline, based on paired t-tests ($p < 0.01$).

still shows consistent improvements on BM25 + RM3 baselines across all four newswire collections. It's unclear which individual classification model is significantly better than the other two. Nevertheless, a further experiment could be conducted to study the performance of ensemble models.

Overall, in this section, we confirm the success of PRF via text classification on both sparse and dense retrieval. We state that PRF via text classification with the aid of transformers generally performs better than the ones with the traditional BM25 retriever. PRF via text classification tends to perform better with a more robust base model, e.g., DistillBERT KD TASB. The performances of the three classifications are close to each other. Note that all improvements come directly from shuffling the ranked list, as PRF via text classification is applied as a reranker on the candidate list.

5.2.2 PRF via text classification on Document ranking

Results for PRF via text classification on MS MARCO Document collection are presented on Table 5.5. Similarly to Table 5.4, we report one baseline and three classification results in each block. As the index and encoder for DistillBERT KD TASB transformer are missing

with the MS MARCO Document collection, we choose to omit this case. Classification is applied to the full document ranked list instead of segmented ones for simplicity. For the three rerankers, we report recall@100 instead of recall@1000, so the reader could still view a change on the recall@100 metric.

As expected, almost all PRF rerankers outperform the initial retrieval results, though the improvement on MAP is only statistically significant on TREC 2019 with the "LR" and "SVM" classifiers. We reach a similar conclusion in terms of nDCG@10 and recall@100 on TREC 2019 and TREC 2020 topics. As the performance of ANCE and TCT-ColBERT V2 HN+ on MSMARCO Document is not substantially greater than the BM25 one, we can't observe an obvious advantage brought from transformers on PRF via text classification.

5.2.3 Comparison of two PRF implementations

Another interesting comparison is PRF via text classification v.s. PRF with Rocchio's Technique. For MS MARCO Passage ranking, we firstly compare the BM25 + text classification in Rows (1b–1d) of Table 5.4 with BM25 + Rocchio in Row (2a) of Table 5.6. To our surprise, the best result of PRF via text classification is still inferior to Rocchio's Technique baseline. The same conclusion could be achieved by comparing the other three dense retrievers with PRF via text classification to their corresponding Rocchio's Technique in Table 5.6. Although PRF via text classification improves with both sparse and dense retrieval, Rocchio's Technique is still preferred.

For MS MARCO Document ranking, by comparing BM25 + classification in Rows (1b–1d) of Table 5.5 with BM25 + Rocchio baseline in Row (2c) of Table 5.7, we can observe that the performance of PRF via text classification and Rocchio's Technique is surprisingly close in terms of MAP and nDCG@10. A similar conclusion could be achieved by comparing the results from two other dense retrievers. Though the recall@100 scores are close for these two PRF implementations, Rocchio's Technique should produce a higher recall@1000 score. As it's performing an end-to-end retrieval rather than a reranking, it should introduce a set of new relevant documents and rule out some non-relevant ones to the top 1000 hits.

5.3 Research Question 3

To investigate the collaboration of two PRF implementations, we perform experiments with a cross combination between sparse/dense Rocchio's Technique and sparse/dense

	Base Model	Expansion	Classifier	TREC 2019			TREC 2020		
				MAP	nDCG@10	R@1k	MAP	nDCG@10	R@1k
(1a)	BM25	RM3	-	0.3390	0.5286	0.7998	0.3019	0.5077	0.8217
(1b)	BM25	RM3	sparse LR	0.3361	0.5199	0.7998	0.2864	0.5028	0.8217
(1c)	BM25	RM3	sparse SVM	0.3397	0.5263	0.7998	0.2983	0.5084	0.8217
(1d)	BM25	RM3	sparse KNN	0.3353	0.5241	0.7998	0.2942	0.4980	0.8217
(2a)	BM25	Rocchio	-	0.3474	0.5339	0.8007	0.3102	0.5095	0.8156
(2b)	BM25	Rocchio	sparse LR	0.3436	0.5306	0.8007	0.2982	0.5074	0.8156
(2c)	BM25	Rocchio	sparse SVM	0.3488	0.5337	0.8007	0.3095	0.5129	0.8156
(2d)	BM25	Rocchio	sparse KNN	0.3447	0.5270	0.8007	0.3047	0.4998	0.8156
(3a)	BM25	Rocchio	-	0.3474	0.5339	0.8007	0.3102	0.5095	0.8156
(3b)	BM25	Rocchio	dense LR	0.3682	0.5627	0.8007	0.3242	0.5317	0.8156
(3c)	BM25	Rocchio	dense SVM	0.3600	0.5559	0.8007	0.3229	0.5321	0.8156
(3d)	BM25	Rocchio	dense KNN	0.3637	0.5485	0.8007	0.3282	0.5262	0.8156
(4a)	ANCE	Rocchio	-	0.4211	0.5928	0.7825	0.4315	0.5800	0.7957
(4b)	ANCE	Rocchio	sparse LR	0.424	0.6138	0.7825	0.4385	0.6053	0.7957
(4c)	ANCE	Rocchio	sparse SVM	0.4362	0.6169	0.7825	0.4374	0.6113	0.7957
(4d)	ANCE	Rocchio	sparse KNN	0.4215	0.5926	0.7825	0.4337	0.5996	0.7957
(5a)	ANCE	Rocchio	-	0.4211	0.5928	0.7825	0.4315	0.5800	0.7957
(5b)	ANCE	Rocchio	dense LR	0.4128	0.5976	0.7825	0.4234	0.5855	0.7957
(5c)	ANCE	Rocchio	dense SVM	0.4222	0.6001	0.7825	0.4252	0.5871	0.7957
(5d)	ANCE	Rocchio	dense KNN	0.4293	0.5948	0.7825	0.4372	0.5918	0.7957
(6a)	TCT-ColBERT V2 HN+	Rocchio	-	0.4883	0.6684	0.8694	0.486	0.6254	0.8518
(6b)	TCT-ColBERT V2 HN+	Rocchio	sparse LR	0.4844	0.6702	0.8694	0.4694	0.6277	0.8518
(6c)	TCT-ColBERT V2 HN+	Rocchio	sparse SVM	0.4943	0.6771	0.8694	0.4868	0.646	0.8518
(6d)	TCT-ColBERT V2 HN+	Rocchio	sparse KNN	0.4796	0.6529	0.8694	0.476	0.628	0.8518
(7a)	TCT-ColBERT V2 HN+	Rocchio	-	0.4883	0.6684	0.8694	0.486	0.6254	0.8518
(7b)	TCT-ColBERT V2 HN+	Rocchio	dense LR	0.4841	0.661	0.8694	0.4627	0.6167	0.8518
(7c)	TCT-ColBERT V2 HN+	Rocchio	dense SVM	0.4894	0.6712	0.8694	0.4797	0.6266	0.8518
(7d)	TCT-ColBERT V2 HN+	Rocchio	dense KNN	0.4749	0.6477	0.8694	0.4691	0.6125	0.8518
(8a)	DistillBERT KD TASB	Rocchio	-	0.4974	0.6684	0.8775	0.4879	0.6470	0.8926
(8b)	DistillBERT KD TASB	Rocchio	sparse LR	0.5007	0.6716	0.8775	0.4889	0.6644	0.8926
(8c)	DistillBERT KD TASB	Rocchio	sparse SVM	0.5077	0.6803	0.8775	0.4961	0.6736	0.8926
(8d)	DistillBERT KD TASB	Rocchio	sparse KNN	0.4868	0.6497	0.8775	0.4723	0.6515	0.8926
(9a)	DistillBERT KD TASB	Rocchio	-	0.4974	0.6684	0.8775	0.4879	0.6470	0.8926
(9b)	DistillBERT KD TASB	Rocchio	dense LR	0.5055	0.6714	0.8775	0.4898	0.6569	0.8926
(9c)	DistillBERT KD TASB	Rocchio	dense SVM	0.5081	0.6758	0.8775	0.4909	0.6553	0.8926
(9d)	DistillBERT KD TASB	Rocchio	dense KNN	0.4963	0.6556	0.8775	0.4831	0.6488	0.8926

Table 5.6: Experimental results on the MS MARCO passage collection with Rocchio's Technique and PRF via text classification on TREC 2019 and TREC 2020 Deep Learning Tracks. Superscripts *** indicates the significant improvements over the baseline, based on paired t-tests ($p < 0.01$).

	Base Model	Expansion	Classifier	TREC 2019			TREC 2020		
				MAP	nDCG@10	R@100	MAP	nDCG@10	R@100
(1a)	BM25	RM3	-	0.2774	0.5170	0.4189	0.4014	0.5225	0.6414
(1b)	BM25	RM3	sparse LR	0.2800	0.5318	0.4205	0.3916	0.5342	0.6320
(1c)	BM25	RM3	sparse SVM	0.2779	0.5345	0.4036	0.3932	0.5321	0.6318
(1d)	BM25	RM3	sparse KNN	0.2535	0.5058	0.3921	0.3720***	0.5215	0.5920
(2a)	BM25	Rocchio	-	0.2811	0.5256	0.4261	0.4089	0.5192	0.6425
(2b)	BM25	Rocchio	sparse LR	0.2935	0.5485	0.4327	0.3950	0.5298	0.6369
(2c)	BM25	Rocchio	sparse SVM	0.2836	0.5395	0.4136	0.3978	0.5272	0.6368
(2d)	BM25	Rocchio	sparse KNN	0.2580	0.5182	0.3869	0.3751	0.5121	0.5978
(3a)	BM25	Rocchio	-	0.2811	0.5256	0.4261	0.4089	0.5192	0.6425
(3b)	BM25	Rocchio	dense LR	0.2540***	0.5216	0.4074	0.3625***	0.4968	0.6250
(3c)	BM25	Rocchio	dense SVM	0.2143***	0.4294	0.3737	0.3325***	0.4695	0.5821
(3d)	BM25	Rocchio	dense KNN	0.2044***	0.4617	0.3305	0.3081***	0.4700	0.5442
(4a)	ANCE	Rocchio	-	0.2351	0.6592	0.3358	0.3962	0.6279	0.5619
(4b)	ANCE	Rocchio	sparse LR	0.2590***	0.6624	0.3669	0.3900	0.6266	0.5846
(4c)	ANCE	Rocchio	sparse SVM	0.2586***	0.6649	0.3695	0.3922	0.6248	0.5778
(4d)	ANCE	Rocchio	sparse KNN	0.2516	0.6544	0.3620	0.4035	0.6300	0.5720
(5a)	ANCE	Rocchio	-	0.2351	0.6592	0.3358	0.3962	0.6279	0.5619
(5b)	ANCE	Rocchio	dense LR	0.2030***	0.6409	0.3152	0.3642	0.6145	0.5377
(5c)	ANCE	Rocchio	dense SVM	0.1934***	0.6057	0.2979	0.3163***	0.5738	0.4962
(5d)	ANCE	Rocchio	dense KNN	0.1614***	0.5956	0.2620	0.2947***	0.5474	0.4594
(6a)	TCT-ColBERT V2 HN	Rocchio	-	0.2946	0.6594	0.4241	0.4292	0.6391	0.6253
(6b)	TCT-ColBERT V2 HN	Rocchio	sparse LR	0.3180	0.6720	0.4553	0.4337	0.6294	0.6568
(6c)	TCT-ColBERT V2 HN	Rocchio	sparse SVM	0.3147	0.6748	0.4471	0.4318	0.6408	0.6433
(6d)	TCT-ColBERT V2 HN	Rocchio	sparse KNN	0.2920	0.6566	0.4292	0.4248	0.6372	0.6341
(7a)	TCT-ColBERT V2 HN	Rocchio	-	0.2946	0.6594	0.4241	0.4292	0.6391	0.6253
(7b)	TCT-ColBERT V2 HN	Rocchio	dense LR	0.2689***	0.6554	0.4116	0.4078	0.6230	0.6194
(7c)	TCT-ColBERT V2 HN	Rocchio	dense SVM	0.2363***	0.6044	0.3782	0.3805***	0.5843	0.5836
(7d)	TCT-ColBERT V2 HN	Rocchio	dense KNN	0.1764***	0.5984	0.3002	0.3067***	0.5370	0.5047

Table 5.7: Experimental results on the MS MARCO Document collection with Rocchio's Technique and PRF via text classification on TREC 2019 and TREC 2020 Deep Learning Tracks. Superscripts *** indicates significant improvements over the baseline, based on paired t-tests ($p < 0.01$).

PRF via text classifications. To distinguish the document representations we utilized for training classifiers, we introduce two new configurations on the "Classifier" column: "dense classifier" and "sparse classifier". The results for MS MARCO Passage Ranking are shown on Table 5.6 while the MS MARCO Document Ranking ones are shown on Table 5.7.

When comparing the results of sparse classifiers to the dense classifiers in Table 5.6, we can note that "hybrid models" generally perform better than the models with only sparse representations or dense representations. Here, the "hybrid model" represents two different configurations: Rocchio's Technique (BM25-derived) + text classification (transformer-derived) or Rocchio's Technique (transformer-derived) + text classification (BM25-derived).

Except for "hybrid models", we observe that PRF via text classification either hurts MAP or marginally improves it for both TREC 2019 and TREC 2020 Deep Learning Tracks. Besides, by comparing the three classifiers, we can observe that the "SVM" classifiers generate more stable and effective results than the other two in general.

Now we shift our focus to MS MARCO Document ranking on Table 5.7. We can see that the sparse classifiers are superior to the dense classifiers under all circumstances. Another interesting observation is that KNN seems to perform the worst in almost all conditions. We can explain it from two perspectives. Firstly, due to the similarity of KNN and Rocchio's Technique we discussed above, KNN can barely learn additional ranking features beyond what has already been learned by Rocchio's Technique. Besides, as the default KNN only has five neighbours, the model is too simple to resolve complicated MS MARCO Document ranking tasks.

Overall, on MS MARCO Passage ranking, the combination of Rocchio's Technique with PRF via text classification doesn't make much difference in effectiveness. In the more challenging MS MARCO Document ranking, applying PRF via text classification on top of Rocchio's Technique even decreases the performance in the majority of cases.

Chapter 6

Conclusion and Future Work

Over the last three decades, deep learning transformers have been successfully applied to many information retrieval tasks, while the number of publications regarding deep learning is growing yearly. On the other hand, to pursue an improvement in effectiveness, researchers choose to build more and more complicated neural networks with fancier loss functions and deeper architectures. However, more recent papers start to question the necessity of complex networks and the reliability of empirical gains. [46, 1, 14]. In this spirit, we wonder whether some simple and old-fashioned techniques are still valuable for further exploration in this neural age?

Motivated by this thought, we tackle the decades-old pseudo relevance feedback problem and raise the following three research questions. In RQ1, we questioned whether Rocchio's Technique still performs well with both sparse and dense representations in modern information retrieval tasks. If so, how well is it compared to other query expansion approaches? While in the RQ2, we shifted our focus to the performance of another more recent technique, PRF via text classification. After we confirm the success of both two implementations of pseudo relevance feedback, we work on a combination of them.

To answer RQ1, we conducted experiments by applying Rocchio's Technique on top of both BM25, and BM25 w/ doc2query-T5 base runs. It turns out that with Rocchio's Technique, we can achieve up to a 0.05 increase across all TREC 2019 and TREC 2020 topics. Due to its competitive score compared to RM3 expansion and its capability of integrating PRF judgments with deep learning transformers, we propose switching RM3 into Rocchio's Technique as a stronger baseline.

The answer to the RQ2 is also yes. We observe large gains in effectiveness across all conditions by reranking the initial candidate list using classifiers trained with sparse

and dense document representations. Note that this is a lightweight reranking technique that could be easily applied to any candidate list without the usage of topic and corpus information.

Our experiments for RQ3 demonstrate the PRF via text classification doesn't make much difference on top of Rocchio's Technique baseline. That could be explained as Rocchio's Technique perform so well, and PRF via text classification fail to capture additional ranking information.

All in all, these answers verify the robustness and effectiveness of two "old-fashioned" techniques in the neural age. Doubtless, neural networks manage to capture complicated features and patterns much better than traditional techniques; nevertheless, some "out-of-date" techniques should not be ignored or forgotten.

Due to the limited time and resources, there are a few potential directions we hope to explore in future works. A first possible direction is to expand Rocchio's Technique and PRF via text classification with more collections, baselines, and different configurations, e.g., different transformers and different PRF depth. Additionally, as we apply PRF via classification on top of Rocchio's Technique to answer RQ3, we wonder what if we change the order of these two techniques? It would be interesting to explore whether the conclusion still holds if we apply Rocchio's Technique on top of PRF via classification instead.

References

[1] M. Baker. 1,500 scientists lift the lid on reproducibility. In *Nature*, page 452–454, 2016.

[2] Salton G. Buckley, C. Optimization of relevance feedback weights. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–357, 1995.

[3] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[4] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)*, 44(1):1–50, 2012.

[5] Wei-Cheng Chang, Felix X Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. Pre-training tasks for embedding-based large-scale retrieval. *arXiv preprint arXiv:2002.03932*, 2020.

[6] Gordon V. Cormack and Mona Mojdeh. Machine learning for information retrieval: trec 2009 web, relevance feedback and legal tracks. In *Proceedings of the Eighteenth Text REtrieval Conference (TREC 2009)*, Gaithersburg, Maryland, 2009.

[7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the trec 2020 deep learning track. In *Proceedings of the Twenty-Nineteenth Text REtrieval Conference (TREC 2020)*, 2020.

[8] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen Voorhees. Overview of the trec 2019 deep learning track. In *Proceedings of the Twenty-Eighteenth Text REtrieval Conference (TREC 2019)*, 2019.

[9] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.

[10] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR, 2020.

[11] Xiao Han, Yuqi Liu, and Jimmy Lin. The simplest thing that can possibly work: (pseudo-)relevance feedback via text classification. In Faegheh Hasibi, Yi Fang, and Akiko Aizawa, editors, *ICTIR '21: The 2021 ACM SIGIR International Conference on the Theory of Information Retrieval, Virtual Event, Canada, July 11, 2021*, pages 123–129. ACM, 2021.

[12] Xiao Han, Yuqi Liu, and Jimmy Lin. The simplest thing that can possibly work:(pseudo-) relevance feedback via text classification. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 123–129, 2021.

[13] Donna Harman. Relevance feedback revisited. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 1–10, 1992.

[14] Islam R. Bachman P. Pineau J. Precup D. Meger D. Henderson, P. Deep reinforcement learning that matters. In *In Proceedings of Thirthy-Second AAAI Conference On Artificial Intelligence (AAAI)*, 2018.

[15] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 113–122, 2021.

[16] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, pages 263–272. IEEE, 2008.

[17] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*, 2019.

[18] Eleanor Ide. New experiments in relevance feedback. *The SMART retrieval system: Experiments in automatic document processing*, pages 337–354, 1971.

[19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[20] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics.

[21] Deepthi Katta. *A study of relevance feedback in vector space model*. PhD thesis, University of Nevada, Las Vegas, 2009.

[22] Saar Kuzi, Anna Shtok, and Oren Kurland. Query expansion using word embeddings. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 1929–1932, 2016.

[23] Victor Lavrenko and W Bruce Croft. Relevance-based language models. 51(2):260–267, 2017.

[24] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*, 2019.

[25] Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and G. Zuccon. Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls. *ArXiv*, abs/2108.11044, 2021.

[26] Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls. *arXiv preprint arXiv:2108.11044*, 2021.

[27] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2356–2362, 2021.

[28] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. Pretrained transformers for text ranking: Bert and beyond. *Synthesis Lectures on Human Language Technologies*, 14(4):1–325, 2021.

[29] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173, 2021.

[30] Xueguang Ma, Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. Document expansions and learned sparse lexical representations for ms marco v1 and v2. 2022.

[31] Pitts W. McCulloch, W.S. A logical calculus of the ideas immanent in nervous activity. In *Bulletin of Mathematical Biophysics 5*, page 115–133, 1943.

[32] Jun Miao, Jimmy Xiangji Huang, and Zheng Ye. Proximity-based rocchio's model for pseudo relevance. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, page 535–544, New York, NY, USA, 2012. Association for Computing Machinery.

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[34] Shahrzad Naseri, Jeffrey Dalton, Andrew Yates, and James Allan. Ceqe: Contextualized embeddings for query expansion. In *European Conference on Information Retrieval*, pages 467–482. Springer, 2021.

[35] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. *MS MARCO: A human generated machine reading comprehension dataset*. 2016.

[36] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

[37] Douglas W. Oard, Jinmook Kim, et al. Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, volume 83, pages 81–83. AAAI, 1998.

[38] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. *arXiv preprint arXiv:2101.05667*, 2021.

[39] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

[40] Joseph John Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System—Experiments in Automatic Document Processing*, pages 313–323, Englewood Cliffs, New Jersey, 1971. Prentice-Hall.

[41] F. Rosenblatt. The perceptron: A perceiving and recognizing automaton. In *Report 85-60-1, Cornell Aeronautical Laboratory*, 1957.

[42] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American society for information science*, 41(4):288–297, 1990.

[43] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[44] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[45] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[46] Snoek J. Wiltschko A. Rahimi A. Sculley, D. Winner's curse? on pace, progress, and empirical rigor. In *Proceedings of the 6th International Conference on Learning Representations, Workshop Track*, 2018.

[47] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Nancy Xin Ru Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. CORD-19: The COVID-19 open research dataset. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online, July 2020. Association for Computational Linguistics.

[48] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.

[49] Jinxi Xu and W Bruce Croft. Quary expansion using local and global document analysis. In *Acm sigir forum*, volume 51, pages 168–175. ACM New York, NY, USA, 2017.

[50] Zuobing Xu and Ram Akella. A bayesian logistic regression model for active relevance feedback. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008)*, pages 227–234, Singapore, 2008.

[51] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 1253–1256, 2017.

[52] Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. A robustly optimized bert pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, 2021.