Turkish Journal of Electrical Engineering & Computer Sciences
http://journals.tubitak.gov.tr/elektrik/
Research Article
Turk J Elec Eng & Comp Sci
(2022) 30: 1914 – 1930

doi:10.3906/elk-1300-0632.3913

Learning term weights by overfitting pairwise ranking loss

Ömer ŞAHİN*, **İlyas ÇİÇEKLİ**, **Gönenç ERCAN**
Department of Computer Engineering, Faculty of Engineering, Hacettepe University, Ankara, Turkey

Received: 03.12.2021 • **Accepted/Published Online:** 08.05.2022 • **Final Version:** 22.07.2022

Abstract: A search engine strikes a balance between effectiveness and efficiency to retrieve the best documents in a scalable way. Recent deep learning-based ranker methods are proving to be effective and improving the state-of-the-art in relevancy metrics. However, as opposed to index-based retrieval methods, neural rankers like bidirectional encoder representations from transformers (BERT) do not scale to large datasets. In this article, we propose a query term weighting method that can be used with a standard inverted index without modifying it. Query term weights are learned using relevant and irrelevant document pairs for each query, using a pairwise ranking loss. The learned weights prove to be more effective than term recall which is a probabilistic relevance feedback, previously used for the task. We further show that these weights can be predicted with a BERT regression model and improve the performance of both a BM25 based index and an index already optimized with a term weighting function.

Key words: Information retrieval, passage ranking, term weighting, pairwise ranking optimization

1. Introduction

A typical search engine retrieves the "best" matching documents to a user query. Its effectiveness is measured through how accurately it ranks documents against the query. On the other hand, the retrieval algorithm must scale to billions of documents and thus must be efficient. Recent research [1–4] shows that contextualized word embeddings like bidirectional encoder representations from transformers (BERT) [5] are effective in retrieval and outperform the long-standing baseline BM25 [6] significantly. Unfortunately, processing all documents with a ranker like BERT is not efficient and scalable. Recent studies try to address this problem by striking a balance between effectiveness and efficiency using multistage retrieval systems [7, 8].

As an alternative solution, it is possible to process only the query with BERT and transfer information via term weights to an efficient index-based retrieval method like BM25. This allows the retrieval system to scale to large datasets but still be able to capture the importance of the terms using rich semantic information captured in BERT embeddings. Furthermore, a readily available standard inverted index-based retrieval system like Lucene[1] [9] can be used without any modifications.

Frequency-based ranking algorithms such as BM25 use corpus statistics to determine the importance weights for terms. A term appearing in a large portion of the documents is deemed to be less important, while a less frequent term is assigned a large weight. Inverse document frequency (IDF) is an effective weighting scheme that incorporates this to BM25. BM25 can be further improved when some form of relevance feedback is available, i.e. when relevant documents to the query are known. The proportion of relevant documents where

*Correspondence: sahin.omer@outlook.com
[1]Apache Lucene search engine https://lucene.apache.org/

a query term appears is used as the weighting factor. While this improves the retrieval effectiveness, the term recall values are specific to the query and do not generalize to a diverse set of queries.

To overcome this problem, Dai and Callan proposed a method to use BERT for predicting the term recall values for an unseen query [10]. They report significant improvements over BM25 baseline using predicted term recall values. The sum of BM25 score of each query term for a document gives the final ranking score of the document. Estimated weights measuring query term importance are used as coefficients in the document similarity method.

Term recall-based weights are effective in optimizing recall at high cutoff values, as terms appearing in all relevant documents are boosted. A term appearing in all relevant documents will be assigned a high weight value, even though it is also commonly used in irrelevant documents. A stop word appearing in all documents will be assigned a high weight. In this article, we investigate the use of pairwise ranking loss to learn the term weights per query as a replacement for term recall values. This can potentially adjust term weights to better distinguish relevant documents from irrelevant ones. Furthermore, we continue to show that these learned weights can be predicted by the BERT model and yield performance improvements over term recall-based weights.

To increase effectiveness and maintain efficiency simultaneously, most of the approaches in the literature do index-time term weighting, which manipulates documents before the index. Considering the size and accessibility of the indices, reindexing might not be feasible or possible. In other respects, term weights can also be applied in search time by preprocessing the query with a deep learning model. In such a case, term weighting for query terms is an effective and efficient way to fetch relevant documents for the query. For this purpose, we focus on query term weighting that aims to estimate optimum weights for query terms by optimizing a pairwise ranking loss to achieve higher scores for relevant documents than irrelevant ones and propose a BERT regression model to predict desired target term weights. We actualize full-ranking on a large collection of passages by the query term weighting in an efficient and effective way. Thanks to the minimal overhead for query term weighting, more complex reranking models can be performed on the first-stage ranking results of the proposed method.

There is a well-known statistical relevance feedback method called term recall and various term weighting schemes that use term recall in the literature. Most term weighting models focus on document term weighting, and there are fewer query term weighting models. Furthermore, there is no study to use query and document term weighting together. There is a work area for new relevance feedback and evaluating the combination of query and document term weighting approaches in the literature.

We propose a new relevance feedback method that learns optimum term weights for a query to retrieve relevant documents in the top ranks. The pairwise loss-based optimization method tries to find the best coefficients for query terms that boost term contribution in BM25. When boosting terms according to importance fetching relevant documents, it must not tend to favor irrelevant documents. The proposed relevance feedback method called pairwise term weight optimization generalizes over hard instances of relevant and irrelevant pairs to choose relevant documents instead of irrelevant ones. The optimization highlights terms that express the query and relevant document and reduce the effect of the generic terms that are insignificant for relevant documents.

Pairwise term weight optimization runs supervised, which means it requires labeled data for the query. To boost query term weights in the search time, we propose BERT-based regression model that takes term and the query as input, and the model estimates target term weight which can be term recall or pairwise optimized ones. The proposed BERT regression model allows to phrase weighting like term weighting as distinct from

the term estimation frameworks in the literature thanks to the term-query input design. The BERT regression model is trained with the proposed relevance feedback as offline, and with the minor addition of inference time for query term weight estimation in search time, better retrieval and ranking performances are obtained.

In the following section, related studies in the literature on ranking methods are presented, with a focus on methods estimating term weights. Following this, in Section 3, we introduce how relevance information of documents can be used to learn the target term weights. Section 4 introduces the BERT-based term weight prediction method. Consecutively, experimental setup and results are presented, comparing the performance of the proposed method to baseline query term weighting methods.

2. Related work

Terms have different contributions to the relevance score. The term frequencies do not involve any contextual information to retrieve a document. A relevance feedback metric can be defined when knowing the relevance between queries and documents in the collection.

A probabilistic relevance feedback is defined by [11]. Assume that, for each term x_t in the query, there is constant p_t that is probability estimation. Set of user judgment relevant documents is defined as $R = \{d : R_{d,q} = 1\}$. When relevant and irrelevant document sets are big enough, the relevance feedback p_t for term t is shown as follows:

$$p_t = \frac{|VR_t|}{|VR|} \tag{1}$$

where VR_t is the set of relevant documents that contain term x_t, and VR is the set of all relevant documents. This probabilistic relevance feedback is similar to term recall.

Term weighting is a way to achieve better retrieval performance by increasing the importance of terms that are central and more representative of the document. Naturally, an accurate term weighting method benefits from rich semantic knowledge. Term weighting can be applied over document terms and query terms. Due to their importance, there are several term weighting methods for query and document terms in the literature.

DeepTR [12] is a query term weighting method that boosts the score of the term in the ranking function. DeepTR tries to estimate term recall [11] relevance feedback by using the distance between the term and the query as a feature vector. Terms are represented in the feature space by Word2Vec [13] word embeddings. The query is the average of the word vectors that compose the query itself.

DeepCT [10] is a contextual term weighting framework for document and query terms. The contextual knowledge about text extracted with BERT [5] language model and a regression layer tries to predict target relevance feedback which is term recall [11]. The pretrained BERT language model is fine-tuned for the task that estimates term recall values with an additional regression layer.

TextRank [14] is a graph-based keyword extraction model. TextRank uses the PageRank [15] algorithm to find the most important terms in the document by using their cooccurrence matrix. Relevance feedback of the terms is defined by the PageRank algorithm in Text Rank. Most cooccurrence terms in the documents are the most important terms to represent the documents, and the terms that cooccurrence with most important terms is important as well in regard to the PageRank algorithm.

Document expansion models change term frequencies by adding terms and reduce term mismatching between query and document by adding new terms to the document. Doc2Query [16] and its follow-up work DocTTTTTQuery [17] models expand documents by estimating queries that address to the document. Both

models use sequence-to-sequence [18] neural network architecture that takes documents as input sequence and predicts queries as output sequence. Doc2Query model is trained end-to-end to estimate possible queries for documents. DocTTTTTQuery model is fine-tuned for query estimation task from text-to-text transfer transformer [19] model trained as a generative language model for different tasks with rich text data.

A more recent research, DeepImpact [20] combines document expansion by DocTTTTTQuery [17] and term weighting together. The enriched documents with the help of document expansion by DocTTTTTQuery reduce word mismatching, and the contextual document term weighting helps to get better ranking and retrieving performance.

3. Utilizing relevance feedback

The ranking is the task of finding relatively higher scores for relevant documents than irrelevant ones. In this way, relevant documents are placed at the top of the output result list. Relevance feedback information identifies if the presented documents are relevant or not. While this information is not usually available, methods using pseudo-relevance feedback exist that utilize user interactions to adjust the retrieval methods. Term recall is such a method that uses the term weights to adjust the retrieval function like BM25 to retrieve relevant documents.

The term weighted BM25 score of a query Q for document D, is calculated as follows:

$$BM25_{tw}(D,Q) = \sum_{i=1}^{n} w_i \cdot IDF(q_i) \cdot TF(D,q_i) \tag{2}$$

where q_i is the ith term of the query, w_i is the weight of q_i, $TF(D,q_i)$ is the term frequency of q_i in the document D, and $IDF(q_i)$ is the inverse document frequency for q_i. There is additional coefficient w_i for term weight unlike BM25 [6]. Weight coefficient of query terms is 1 in BM25.

When ranking documents for the given query, each term has a different effect on fetching relevant documents. Some terms are more meaningful to understand the keywords of the query, while others serve a more grammatical purpose. Term recall is a probabilistic weight function for setting w_i weights. It uses only relevant documents to weighting the terms and without using the information in other irrelevant documents in the corpus.

3.1. Term recall-based term weighting

The importance of the query term in the given query to retrieval success is determined by the term recall value. Term recall is the ratio of the number of relevant documents that contain the term over the total number of all relevant documents. Thus, term recall shows the usage frequency of the query term in the relevant documents and contribution to fetch relevant ones. Term recall equation is as follows:

$$TermRecall = \frac{|R_{q,t}|}{|R_q|} \tag{3}$$

where $R_{q,t}$ is set of relevant documents that contain the term of query, and R_q is set of relevant documents for the query. DeepCT-Query [10] uses the term recall weights as the target metric and predicts the term recall values for unseen queries using BERT. While this weighting scheme rewards terms appearing in relevant documents, it does not penalize terms with low discriminative value. A stop word appearing in all documents, both relevant and irrelevant will receive the maximum weight under this framework.

3.2. Pairwise ranking loss-based term weighting

We propose an optimization function that learns optimal weights for query terms to achieve a higher weighted BM25 score for relevant documents than irrelevant ones. For each query, a set of weights for terms are optimized by pairwise ranking loss. The set of weights of the query terms is unique and independent from the other queries. Optimization is applied over the BM25 scores of each term in the query. *TF* and *IDF* values of the terms are included in the optimization of term weights in a query by comparing relevant and irrelevant document pairs. Term recall only uses whether a document contains the term or not and term recall only knows the relevant documents instead of relevant-irrelevant pairs. The proposed relevance feedback method also uses irrelevant documents besides *TF-IDF*.

Let $BM25(D, q_i)$ be BM25 value of the ith term of the query for document D. In this case, BM25 score of document D with weighted terms equals $w_1 \cdot BM25(D, q_1) + w_2 \cdot BM25(D, q_2) + \cdots + w_n \cdot BM25(D, q_n)$. In this equation, n is the number of terms in the query, and it does not change for different documents. For this equation, we try to optimize term weights to obtain a higher final BM25 score for relevant documents and bring forward the most important query terms for retrieval performance.

Sorting is based on pairwise comparisons between the documents. The documents which are relevant to the query should have a higher rank compared to the irrelevant documents. To find term weights that lift the score of relevant documents, an optimization based on pairwise ranking loss is designed.

For an input query $Q = \{q_1, q_2, \cdots, q_n\}$, relevant documents are paired with irrelevant ones to form a training instance. A training instance is represented with two feature vectors formed of BM25 values for each term in the same order with query terms. Feature vectors can be represented as $D_{Rel} =< x_1, x_2, \cdots, x_n >$ for relevant document and $D_{Irrel} =< y_1, y_2, \cdots, y_n >$ for irrelevant document in the pair. x_i and y_i values are BM25 scores of ith term of the query in respectively relevant and irrelevant document pairs. BM25 score of the relevant document is written like $BM25_{tw}(D_{Rel}, Q) = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n$. In the same way, for irrelevant document BM25, score is $BM25_{tw}(D_{Irrel}, Q) = w_1 \cdot y_1 + w_2 \cdot y_2 + \cdots + w_n \cdot y_n$.

The final objective is to learn term weights that produce high scores for relevant documents while reducing the score of irrelevant ones. The pairwise optimization aims to provide the following condition:

$$BM25_{tw}(D_{Relevant}, Q) \geq BM25_{tw}(D_{Irrelevant}, Q) + \alpha \tag{4}$$

where α is the margin between relevant and irrelevant document pairs, assuring that the difference between a relevant and irrelevant document is at least α [21]. α selected as 1 according to the experimental results and in considering relevance score range of BM25, which is between 0 and $+\infty$.

To find optimal weights using a gradient, Adam [22] optimizer is used. If the weighted BM25 score of the relevant document is greater than the sum of the margin and the irrelevant one, then the loss will be 0, and weights are not updated.

After pairwise ranking loss-based term weighting, some of the optimal weights for query terms can be negative. That means the optimal solution is provided by penalizing unwanted terms. The negative term weights are not applicable by search framework Lucene. Therefore, all weights must be greater or equal to 0. To provide this constraint, there are two ways; the weights are normalized after optimization or force the weights to be positive when searching for an optimal solution.

3.2.1. Min-max normalization (min-max)

Final weights, optimized for the higher BM25 score for relevant documents, are in the range $-\infty$ and ∞. After the optimization, to scale the weights between 0 and 1, $min-max$ normalization is applied as follows:

$$Scaled\ w_i = \frac{w_i - W_{min}}{W_{max} - W_{min}} \tag{5}$$

where w_i is the ith term weight for query, W_{max} is the maximum term weight for the query, and W_{min} is the minimum term weight for the query.

3.2.2. Nonnegative weight constraint (non-neg)

In the training time, the negative weights are projected to another space by assigning to 0 at each iteration after updating weights. In the next iteration, the equation tries to find another solution without negative weights. If a weight tends to negative values, it is prohibited by ignoring weights less than 0 at each iteration. The final solution is in positive space with the help of the nonnegative constraint.

In each iteration, the following operation is applied to optimized weights after updating weights by gradient:

$$\vec{w}_{t+1} = maximum(\vec{w}_t, 0) \tag{6}$$

where $\vec{w}_t$ is the updated weights by optimizer and $\vec{w}_{t+1}$ the new weights for next iteration.

3.2.3. Pair selection

Relevant and irrelevant pair selection starts with the initial BM25 ranking without weighting terms. Relevant instances are known for a query thanks to the labels, and they are picked from the collection. Ranking documents by original (unweighted) queries is called the initial run. Irrelevant instances are selected from the top results that are retrieved by the initial run, after excluding relevant instances for the query. This allows the model to discriminate the relevant documents from irrelevant false positives of the default BM25. There is an example relevant and irrelevant pair for a query in Table 1.

Each relevant document is matched with irrelevant documents from the initial run to construct pairs. Pair generation is based on cross-production of the relevant and irrelevant documents. The number of pairs at the end of the pair generation depends on the number of documents retrieved by the initial run and the number of relevant documents. Most of the queries fetch the desired number of documents from the initial ranking, but some of them may fetch less results if there is less match with the documents.

Each query is optimized independently from the other queries. This method, in a way, overfits weights with respect to relevant and irrelevant documents per each query. In this way, the optimization model predicts the best weights to increase relevant document weights and decrease irrelevant ones when compared to each other. It should be noted that as these weights would be too specific to a query and not generalize to a different query, they would not be useful for a search engine. However, as we proceed to predict these weights for an unseen query using contextualized word embeddings, considering both the semantics of the query and the role of the term in the query, it is applicable to unseen queries. After estimating optimal weights, a contextual model like BERT learns the predicted weights by only using the query terms.

In short, the pairwise term weight optimization method learns specific term weight for each query by overfitting relevant and irrelevant pairs for the query. A language model like BERT aims to understand

contextual information of queries to predict these optimized term weights by learning interactions between terms in the query. In the end, a term weighting model is trained that estimates pairwise term weight for each term in a query.

Table 1. Relevant and irrelevant pair for a query.

Query	What is dynamic resolution?
Relevant	DynamiX is a unique implementation of real-time dynamic resolution technique that is designed to enable a tunable minimum performance level to increase the playability of a game by dynamically changing the render target resolution of objects in real time, without the need of the game developer to design it in advance.
Irrelevant	Imaging the larynx's positions and the vocal folds' vibrations is possible using dynamic MRI. This technique permits measurements of laryngeal structures and glottal parameters in dynamic function with multiplanar high-resolution imaging.

*Query and passages were taken from MS MARCO passage dataset.

3.3. Term weight prediction model

Pairwise term weight optimization model runs in the manner of supervised learning. The model must know relevant and irrelevant pairs for the query with the query itself. For this reason, weight estimation cannot be made for new queries in the search-time. A BERT-based regression model is trained with the weights optimized pairwise, the BERT model uses only the input query to predict term weights of the query.

BERT [5] is a language representation model that is designed to pretrain on unlabeled text corpus by the masked language model (MLM) to learn the context of the text. The architecture of BERT consists of transformer [23] in multilayer bidirectional form. Transformer applies bidirectional self-attention; each token interacts with other tokens in BERT architecture.

BERT has special tokens to represent single sentences and pairs of sentences. Each text sequence starts with the special classification token $[CLS]$, which represents the final hidden state that aggregated through the sequence, and this token is used on the classification tasks. To separate sentences in a pair of text sequences, the special separator token $[SEP]$ is used between the first sentence and the second sentence.

To fine-tune the BERT model, a regression task targeting query term weights is used. BERT takes term and query as a pair. The term of the query and query itself are separated with a special token $[SEP]$. This way, the interaction between individual terms and queries is learned. The pooled output of BERT, which is $[CLS]$ token vector, is followed by a fully connected feed-forward layer with a single output value. The output layer has no activation function due to being trained as a regression task. To prevent overfitting and achieve more generalization, there is a dropout layer with 0.2 probability between pooled output and fully connected layer. The model is trained with MSE (mean squared error) by fine-tuning the pre-trained BERT model to estimate the term weights of the input query. The model is optimized with Adam [22] optimizer, and 10% of training steps are used as warm-up steps.

All training pairs have a target value between 0 and 1. For this reason, most of the predicted term weights fall into this range. In the end, negative weights are ignored and set as 0. Nonnegative term weights enable the use of boosting factors available in most search engines.

Thanks to interaction architecture, phrases that are sequential words can be weighted as terms. The first part of the input is the phrase in the query, and the second part is the query, both of them are separated by the

special token. The target weight predicted by the model is for a phrase instead of a single word in this case.

In search-time, the model estimates the weights of each term of the query simultaneously. For each term in the query, the model estimates weights as the coefficient for terms. An example for tokenization of the query and forming input is given with BERT term weight learning model architecture at Figure.

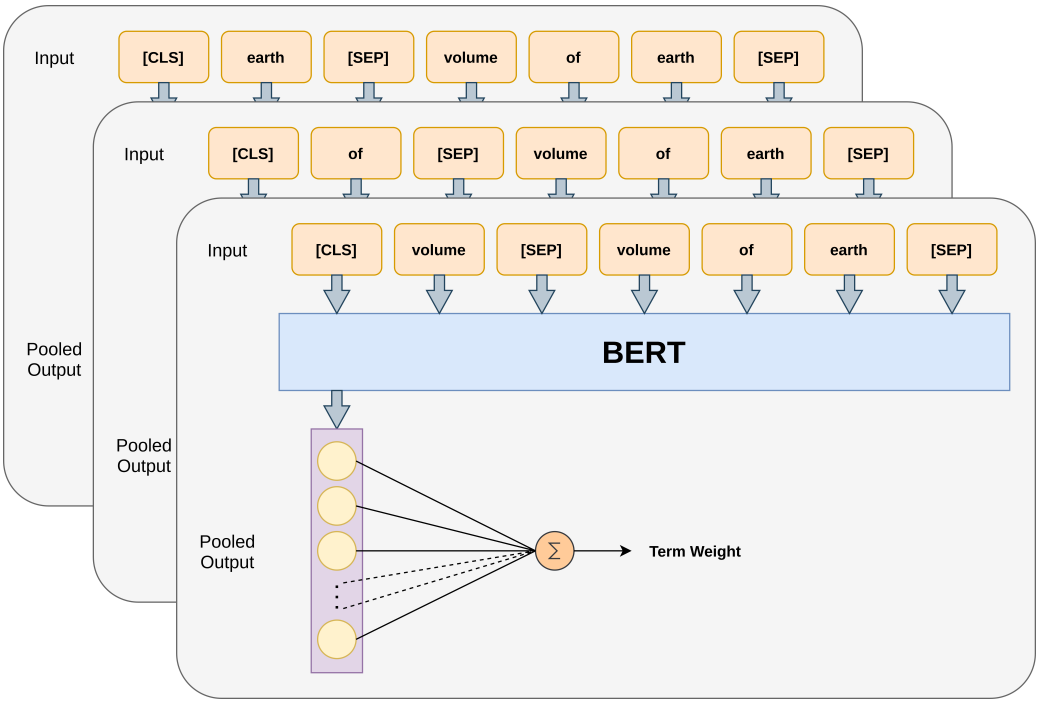

Figure. [BERT Query Term Weighting Model Architecture]
An example input and model architecture for term weight estimation.

The proposed term weighting model and DeepCT [10] framework use the same pretrained BERT model to understand the context of the text to predict target term weight. The main difference between DeepCT and ours is that the proposed model can estimate term weights for phrases with the help of the interaction architecture. DeepCT takes a query as an input and predicts term weights for each token in the query. In other respect, the proposed BERT-based term weight estimation model takes term and query as a pair, and it can predict term weights for tokens or phrases in the same architecture.

4. Experimental setup

4.1. Dataset

MS MARCO[2] [24] dataset is widely used in related work for information retrieval and learning-to-rank tasks. MS MARCO has two tasks which are document and passage ranking. The competition can be evaluated by full-ranking or reranking for both tasks. Document and passage ranking tasks use a large dataset that consists of the same queries and human-supervised labels. Labels are made that indicate relevance between queries and documents by human supervision. In this work, we achieved evaluation scores for the full passage ranking task of MS MARCO and compared proposed and existing methods with this task.

[2]MS MARCO - microsoft.github.io/msmarco

To evaluate query term weighting models, the dataset of MS MARCO passage retrieval task is chosen in this work. The passage ranking dataset has nearly 8,841,823 passages, 502,939 training, and 6980 evaluation queries, evaluation queries are named as dev queries in MS MARCO. The collection of the passages consists of a unique identifier *pid* for each passage and the passage text. Train and evaluation queries have unique identifiers *qid* and query text. The relations between query and passage are given as $pid - qid$ pairs. The relevance between queries and documents is marked as binary, and they can be relevant or irrelevant.

In MS MARCO passage collection, the passages were indexed, and all evaluations for the query term weighting task were run over this index. Document term weighting approaches manipulate original passage collection, and the changed collection was reindexed. The train queries were used in model training and parameter tuning. Parameters of BM25, k_1 and b, were tuned by randomly selected queries from train queries. In model training, part of training queries was selected as the validation set for the model evaluation. Evaluation queries in MS MARCO were used as a test set, all evaluations and comparisons were performed with evaluation queries. The evaluation queries of MS MARCO are used as the test set for evaluated methods in the literature as well.

According to the distribution of the number of passages in MS MARCO collection, most of the passages consist of around 50 terms, and nearly all of them have less than 150 terms. A similar distribution for both train and evaluation query sets, most of them consist of 3 to 10 terms. Another distribution about queries and the number of relevant passages in MS MARCO dataset is that most of the queries in both train and evaluation set refer to a single passage.

4.2. Search framework

To achieve easy access to documents that contain one of the words in the query, the inverted index method is used. The inverted index is a way to access documents by searching a set of words without requiring investigating all documents in the corpus.

To index documents by words and search easily, the Apache Lucene[3] [9] framework is used in this work. Porter stemmer was applied to words when indexing documents. The same stemming is also applied when extracting term recall values and BM25 calculation for the pairwise model.

Free parameters of BM25, which are k_1 and b, are assigned to the same values for evaluation of all methods. Fine-tuned k_1 and b parameters[4] of BM25 are used as $k_1 = 0.82$ and $b = 0.68$. To keep the option to reranking results of the first stage ranking with the proposed method, free parameters optimized for RECALL@1000 were chosen. By this preference, while increasing ranking performance by query term weighting, we aim to fetch relevant documents as much as possible.

4.3. Compared methods

The ranking by BM25 of evaluation queries without term weighting, in other words, each term weight equals 1. As all methods are built on top of BM25 method, they all improve BM25 score by adding weighting to query terms. Each method runs with the same search framework and BM25 configurations, except term weights.

[3] Apache Lucene - lucene.apache.org

[4] Anserini: BM25 Baseline for MS MARCO Passage Ranking, github.com/castorini/anserini/blob/master/docs/experiments-msmarco-passage.md

4.3.1. DeepTR-BoW

DeepTR model was trained 100 iterations as LASSO optimization [25]. There are more than 1.3 million instances for the model training, the 10% of the instances are used as validation data. Pretrained Word2Vec [13] word embeddings are the same as DeepTR [12]. DeepTR was trained to estimate term recall values of the query terms. This method is referred to as *DeepTR-BoW* in the results.

4.3.2. DeepCT

In DeepCT, the BERT model, which is pretrained with a large uncased corpus of English as masked language modeling (MLM) task, and base configuration, was fine-tuned to estimate term recall values of query terms. DeepCT estimates all term weights for a query or a document in a single prediction. To do this, DeepCT uses the sequential output of BERT corresponding to terms. Each vector for terms in sequential output proceeds to a layer that estimates the weight of the term. The last layer of DeepCT has no activation function in the same way as our BERT regression model. Most of the estimated term weights are in the range between 0 and 1 due to training targets being in the same range, and if a predicted weight is less than 0, then the term is ignored.

DeepCT-Query: DeepCT model was fine-tuned to predict query term weight as term recall value with the same configuration[5] as the configuration of DeepCT for MSMARCO.

DeepCT-Index: To evaluate if the pairwise optimization method can be applied to any other corpus, the weighted documents by DeepCT-Index [10] were indexed by Apache Lucene. The corpus of DeepCT-Index[6] is provided by the authors. Free parameters of BM25 are defined as $k1 = 10$ and $b = 0.9$, which are mentioned by the author.

4.3.3. BERT-based term weight

The proposed BERT-based term-query interaction model uses the same pretrained BERT model, which is pretrained using uncased English corpus as MLM task. The term recall and pairwise term weight values are used as the target value of the model to predict term weight only by looking at the input query. The term-query interaction model was fine-tuned only a single epoch for both target term weighting. When the target term weight is term recall, we will refer to this model as BERT-TermRecall. It is referred to as BERT-Pairwise when the term weights are learned by pairwise ranking loss as described in Section 3.2

5. Results

The pairwise term weight optimization method is proposed as an alternative to the term recall to retrieve documents more accurately. After that, the BERT weight estimation model was trained to predict term recall and pairwise optimized term weights and compared with different methods in the literature to evaluate the success of pairwise optimization and the model. Firstly, the target term weight functions are evaluated and compared to each other to measure the effectiveness improvement that can be attributed to the proposed target weights optimized using pairwise ranking loss. Following this, results showing the BERT-based predicted term weights would be evaluated and compared to each other. Finally, we investigate the use of the proposed query term weighting method with an existing index-based term weighting method.

The retrieval and ranking performances are evaluated with different metrics, and different cuts of the results on MS MARCO [24] passage retrieval dataset. Performances of existing and proposed methods are

[5] DeepCT - github.com/AdeDZY/DeepCT
[6] DeepCT Weighted Documents - boston.lti.cs.cmu.edu/appendices/arXiv2019-DeepCT-Zhuyun-Dai/weighted_documents

evaluated with mean reciprocal rank (MRR) [26], mean average precision (MAP), recall, and normalized discounted cumulative gain (NDCG) [27] metrics. The significance of the proposed methods to baselines is measured with *paired t-test*. The statistical significance threshold is defined as 0.05 to measure significance.

The MAP measures the performance of getting relevant documents at the very beginning of the results. MRR and NDCG measure the ranking performance of the models based on the ranking positions of the relevant documents with the input query. The recall metric measures the performance of catching relevant documents in the top results. With these evaluation metrics, the performance of the models is compared by their retrieval and ranking of relevant documents to the input query.

Additionally, the significance of the improvement by proposed methods to baselines is measured. Paired t-test is a statistical hypothesis test that investigates the difference between two lists of data for the same subject. The similarity of the distribution between paired lists of data is tested with the help of paired t-test. When the calculated $p - value$ is less than the significance threshold, there is a significant difference in the distribution of the results.

5.1. Number of pairs for pairwise term weight optimization

In pairwise optimization, pairs are constructed from at the top of initial results to generate hard negative instances. To reduce the score of confusing irrelevant documents, hard negative samples are required. The relevant and irrelevant documents are paired by cross-product to generate pairs with hard negative instances. After ranking documents with the original queries, relevant documents are excluded from the result, and the rest of them are considered irrelevant documents. In other respects, extreme negative instances may prevent the model from being more generalized for collection. Still, there are far more completely irrelevant documents than confusing ones.

To get a number of documents from the initial run that are enough to generalize the model and achieve good results, retrieval and ranking performances are compared in Table 2. The pairwise query term weight optimization method was evaluated with the min-max normalization. In regard to Table 2, the higher the number of documents acquire higher the retrieval and ranking scores. At the point between 800 and 1000 documents, the ranking performance slightly reduces with respect to MRR@10, MAP@10, and NDCG metrics, but then, retrieval performance is still increasing with respect to RECALL@1000. This spot was chosen to define the number of documents to optimize the model without missing relevant documents with acceptable ranking performance loss. In the next experiments, 1000 documents were selected from the top of the initial results.

Table 2. Number of documents to optimize pairwise model.

#Docs	MRR@10	MAP@10	NDCG@5	NDCG@10	RECALL@1000
20	0.2348	0.2398	0.2513	0.2758	0.7522
100	0.2562	0.2625	0.2750	0.3066	0.8193
200	0.2621	0.2692	0.2800	0.3130	0.8406
400	0.2657	0.2723	0.2841	0.3173	0.8610
500	0.2638	0.2709	0.2825	0.3156	0.8706
600	0.2654	0.2722	0.2834	0.3167	0.8733
800	0.2662	0.2733	0.2845	0.3177	0.8784
1000	0.2633	0.2706	0.2814	0.3150	0.8814

The number of pairs for each query can be different. Most of the queries fetch 1000 documents after initial rank. After relevant queries are excluded from the top results, the rest of the documents number differs. In the end, the number of pairs equals all relevant documents number in collection multiplied by irrelevant documents number in the results.

5.2. Term recall and pairwise term weight optimization

Firstly, BM25 and oracle target query term weights are compared to each other. BM25 is the score of the original evaluation queries. The original queries are ranked with BM25 rank algorithm and the same configuration as other methods. As can be seen in Table 3, the target weighting schemes can improve the baseline BM25 by more than 30% in MRR, MAP, and NDCG metrics with improvements up to 15% for RECALL metrics.

The query term weighting is not applied to the queries in BM25 results; in other words, all weights are considered 1. The oracle scores are shown as the best possible outcome when term recall values or pairwise optimized weights are perfectly predicted. The oracle methods use the relevant and irrelevant document labels, which are not typically available at the search time. The target weights are extracted for test queries by supervised. In this regard, they can be regarded as the upper bounds for the BERT-based weight prediction methods. In evaluations of the oracle, term weighting schemes are applied to the evaluated queries directly. Term recall and pairwise optimized weights are extracted for the evaluation set as supervised.

Table 3. Query term weighting with oracle methods using the document relevance information.

Method	MRR@10	MAP@10	RECALL		NDCG		
			@100	@1000	@5	@10	@20
BM25	0.1875	0.1958	0.6700	0.8575	0.2024	0.2342	0.2578
Oracle-TermRecall	0.2582	0.2661	0.7743	0.9234	0.2781	0.3145	0.3393
Oracle-PairWise (Min-Max)	0.2676*	0.2745*	0.7427	0.8805	0.2850	0.3190	0.3413
Oracle-PairWise (Non-Neg)	0.3089*	0.3164*	0.7877*	0.8999	0.3290*	0.3628*	0.3848*

Comparison of oracles which are term recall and pairwise are given in Table 3. According to oracle results, the pairwise term weight optimization methods (min-max normalization and non-neg constraint) achieve significantly (shown in Table 3 with *) higher scores on ranking relevant passages at the very top results as per MRR@10, MAP@10 RECALL@100, and NDCG metrics than the term recall method, and gets slightly better scores on RECALL@1000 metric. *Paired t-test statistic* of the proposed method and the previous method in the literature (*Oracle-PairWise (Non-Neg)* vs. *Oracle-TermRecall*) for MRR@10 is 23.79 and *p-value* is 2.37×10^{-120}.

Pairwise optimization aims to increase BM25 score of relevant documents against irrelevant ones. In this way, the ranking performance of pairwise optimization is way better than term recall. Conversely, term recall is a statistical metric that gives which term is more important to retrieve relevant documents without knowing irrelevant ones. As a result, term weighting by term recall finds much more relevant documents than pairwise optimization when considering wide range of ranks such as top 1000 documents, but metrics focusing on top ranks are improved with *Oracle-Pairwise* as it is able to discriminate relevancy in more detail as it considers irrelevant documents as well.

The weights of the query terms must be positive numbers due to constraints of the Lucene framework. To ensure that, two methods are proposed and evaluated for pairwise optimization. The first one is the fitting

weights into the $[0-1]$ range. This causes at least one weight to equal to 0, and at least one weight to equal to 1. The other method is applied to the model when optimizing weights. The 0 assignment to the negative weights was used as a layer constraint in the second method. At the end of each iteration, negative weights are ignored and assigned to 0. This kind of projection allows us to make sure the weights are always greater or equal to 0.

According to Table 3 nonnegative constraints methods achieve significantly higher performance than the min-max normalization method. The min-max normalization can result in loss of information due to fitting the optimized weights into a fixed range. Therefore, the min-max normalized pairwise optimization had less success on the ranking and retrieval performance.

5.3. Term weight estimation

Oracle results show the possibly best results if predicted weights of the query as term recall or pairwise optimization, which are extracted using the relevant and irrelevant document labels. In real life, there is no way to find exact values of term recall or pairwise term weights due to unique and unknown input queries. A way to overcome these unknown query inputs is by creating a generalized model that learns term weights of queries by supervised and predicts term weights of new queries in the search-time.

To this end, models try to estimate term weight as close as possible to the ground truth to achieve scores as good as oracles. DeepTR-BoW and DeepCT-Query are the query term weighting approaches that use term recall as term weight in the literature for the document ranking task. The proposed BERT-based term-query interaction model is evaluated with term recall and pairwise as the target of the query term weighting task.

When the target value changes to pairwise optimized ones, the query-term interaction model acquires significantly (shown in Table 4 with *) higher scores for ranking and retrieving relevant documents for the input query than DeepCT-Query model. The results indicate that target term weights learned by pairwise ranking loss independently for each query can also be predicted by a BERT-based regression model. The proposed relevance feedback achieves the best results compared to both DeepCT-Query and BERT-TermRecall in all metrics. It is interesting to note that, although Oracle TermRecall achieves high recall values compared to Pairwise oracle weights (Table 3), when considering the BERT, predicted weights recall of BERT-Pairwise is significantly better than BERT-TermRecall. Paired t-test statistic of the proposed method and the previous method in the literature (BERT-Pairwise (Min-Max) vs. DeepCT-Query) for MRR@10 is 6.97 and p-value is 3.57×10^{-12}.

Table 4. Query term weighting using estimated term weights.

Method	MRR@10	MAP@10	RECALL		NDCG		
			@100	@1000	@5	@10	@20
BM25	0.1875	0.1958	0.6700	0.8575	0.2024	0.2342	0.2578
DeepTR-BoW	0.1901	0.1989	0.6845	0.8738	0.2053	0.2380	0.2620
DeepCT-Query	0.1915	0.2005	0.6907	0.8821	0.2060	0.2393	0.2631
BERT-TermRecall	0.1930	0.2020	0.6954	0.8837	0.2080	0.2414	0.2657
BERT-Pairwise (Min-Max)	0.2012*	0.2097*	0.7062*	0.8870*	0.2179*	0.2517*	0.2750*
BERT-Pairwise (Non-Neg)	0.2005*	0.2092*	0.7065*	0.8842	0.2167*	0.2509*	0.2746*

As Table 4 shows, DeepCT-Query achieves slightly higher scores than DeepTR-BoW on any metric, both

methods try to predict term recall values of the query terms. DeepCT-Query tries to understand the semantic of each term in the query in considering contextual relation, but DeepTR-BoW simply uses vectors of terms that measure how far from the query with direction. Due to the use of semantic information, DeepCT-Query is slightly better than DeepTR-BoW for retrieval and ranking tasks.

Pairwise optimization with nonnegative constraint model got higher scores on any evaluation metrics when supervised evaluation (Oracle results, Table 3). However, the BERT-based contextual model that tries to estimate pairwise optimized term weight did not succeed in this increase. BERT-based model made very close estimation for both min-max normalization and nonnegative constraint methods of the pairwise term weight optimization model when comparing the evaluation metrics on ranking and retrieval performance.

As a final note, DeepTR-BoW, DeepCT-Query, and BERT-TermRecall methods all try to predict term recall weights using different supervised regression methods. DeepTR-BoW uses Word2Vec for feature representation and LASSO optimization for term weight predictions. DeepCT-Query and BERT-TermRecall use a similar architecture for term weighting based on BERT and the same target value as relevance feedback. Even so, the proposed BERT-TermRecall method achieves the best results compared to these three methods. The proposed BERT-TermRecall and DeepCT-Query achieve similar results to each other due to the similar architecture. In other respects, pairwise term weight optimization BERT-Pairwise, which is the proposed relevance feedback, achieves best retrieval and ranking results over other methods and term recall.

According to the query term weighting example in Table 5, ”acetate definition” is searched with non-weighted and weighted terms by different methods. The keyword of the given query is ”acetate” and the second term ”definition” can be seen as a question term. In the original query, both terms have the same contribution to the relevance score to retrieve documents, which causes the loss of the contribution of the key term. In other respect, DeepTR-BoW completely ignores the contribution of the question term ”definition”, but both terms carry a mean for relevant documents at different rates. For this reason, DeepTR-BoW got the worst result among the methods. DeepCT-Query and proposed BERT-Pairwise (which predict proposed pairwise optimized relevance feedback with BERT-based regression model) can create a weight distribution over terms. DeepCT-Query uses the term recall as relevance feedback, and estimated term wights focus on the only relevant documents. Because of this knowledge, the rank of the irrelevant documents also increases by giving higher term weight for a key term ”acetate” and lowering the contribution of the question term ”definition”. The proposed relevance feedback increases the score of relevant documents when decreasing irrelevant ones. Thanks to the pairwise optimization by relevant and irrelevant document pairs, the optimal term weight distribution can be found, and with help of the contextual knowledge about the query, weights are correctly estimated.

As a result, the proposed relevance feedback method got more accurate than the statistical term recall, with help of the optimization by using relevant and irrelevant document pairs. Thus, it found and ranked the relevant passage in the first rank, as can be seen in Table 5 and MRR@10 metric. According to ranking and MRR@10 results in Table 5, BERT-Pairwise and DeepCT-Query are better than DeepTR-BoW at the ranking task. This result shows that a model that uses contextual knowledge by using a deep language model can estimate term weight much more correctly than the simple regression model which uses word embedding vectors.

5.4. Combining with index term weighting

DeepCT-Index [10] is a document term weighting method. The term frequencies are modified before indexing using the estimated term weights. The query is executed on the modified index. The proposed query term

Table 5. Query term weight example and ranking performances.

Method	Query	Rank	MRR@10
Nonweighted	acetate definition	3	0.333
DeepTR-BoW	acetate^0.999 definition^0.007	8	0.125
DeepCT-Query	acetate^1.004 definition^0.405	2	0.500
BERT-Pairwise (Min-Max)	acetate^0.654 definition^0.316	1	1.000
Relevant passage with the query: *This is a form of the compound we will be discussing in this lesson: acetate. By definition, acetate is a type of anion, salt, or ester derived from the compound acetic acid. Let's break down this definition by looking at the three different forms of acetate. Acetate Has Many Forms... Keep in mind that as we go through each form of acetate, everything originates from the parent molecule acetic acid.*			

*Estimated term weight coefficients represent with the exponential sign (∧).

**Query and passage were taken from MS MARCO passage dataset.

weighting method can also be applied to a reweighted index such as DeepCT-Index. This strategy can combine the effects of both methods and result in superior outcomes.

Query term weights are estimated by the proposed BERT model using the modified TF and IDF values obtained from DeepCT-Index. The result of BM25, DeepCT-Index, and weighted query one is given in Table 6. As can be seen from the results, the proposed model improves DeepCT-Index significantly (shown in Table 6 with *), with improvements in all metrics. This demonstrates that combining the proposed query term weighting method with existing index term weights can improve the results even further. Paired t-test statistic of the proposed method and the previous method in the literature (DeepCT-Index + BERT Pairwise (Min-Max) vs. DeepCT-Index+Query) for MRR@10 is 3.73 and p-value is 1.95×10^{-4}. The BERT-Pairwise model adds minimal overhead to the search engine, taking less than 23 ms when processed with an RTX 2060 GPU.

Table 6. Combined effect of query term weighting with index term weighting (DeepCT-Index)

Method	MRR@10	MAP@10	RECALL		NDCG		
			@100	@1000	@5	@10	@20
BM25	0.1875	0.1958	0.6700	0.8575	0.2024	0.2342	0.2578
DeepCT-Index	0.2440	0.2516	0.7550	0.9086	0.2624	0.2979	0.3227
DeepCT-Index+Query	0.2487	0.2564	0.7635	0.9111	0.2676	0.3030	0.3275
DeepCT-Index + BERT Pairwise (Min-Max)	0.2534*	0.2608*	0.7732*	0.9137*	0.2728	0.3091*	0.3335*
DeepCT-Index + BERT Pairwise (Non-Neg)	0.2524*	0.2599*	0.7721*	0.9127*	0.2727*	0.3077*	0.3319*

As a baseline, we compared the combination of DeepCT-Index and DeepCT-Query which was presented by the same authors to our model. Similar to Table 4, proposed pairwise optimized relevance feedback and BERT-based model achieved better retrieval and ranking results than DeepCT-Query that learns term recall as relevance feedback. The clear advantage of the proposed pairwise optimized relevance feedback is obtained in these results as well.

Both pairwise optimization methods got close results similar to the original corpus index. The min-max

normalized pairwise optimized weights are slightly better than the nonnegative constraint optimized one.

6. Conclusions

A text search engine works with a huge amount of corpus, and it keeps text data in a structured database to retrieve as fast as possible when queried by a user request. Search engines use retrieval and ranking algorithms when searching a query to find the best candidates that meet the user request in the vast dataset. To take advantage of a search engine in all possible ways, neural network solutions are applied to documents and queries. While doing this, minimal overhead has to be added to the search cost to ensure a fluent user experience.

The usage of neural networks to improve retrieval and ranking performance focus on two sides of the query searching task. The first one is editing documents in index-time, and the second one is weighting query terms in search-time. The index-time operations are offline, and its cost does not affect the execution time of the search. However, weighting query terms adds an additional operation that estimates term weights to the search-time.

The contribution of this work is the utilization of the well-known pairwise loss function for the term weighting as relevance feedback besides comparing term weighting approaches in the literature. The pairwise relevance feedback method finds optimal weights that boost the contribution of essential terms for relevant documents when throwing back the irrelevant ones. The proposed relevance feedback method outperformed the statistical term recall. To support our work, we applied the pairwise optimized term weights estimated by the BERT-based regression model to queries in search-time with minimal cost. The proposed relevance feedback method increased retrieval and ranking performance significantly compared to term recall approaches.

Pairwise term weight optimization requires lots of pairs for generalization and hard negative instances for an optimal solution. A query can fetch more than a thousand documents, and the initial ranking of documents consists of mostly hard negative instances. To this end, the proposed relevance feedback method is demonstrated on the query term weighting task, and the strategy was evaluated from various perspectives.

The combination of query term expansion models and the proposed query term weighting is considered as future work. Evaluating the proposed relevance feedback method on the document term weighting is an additional task for follow-up works. Additionally, studies can increase the predictability of relevance feedback to catch oracle results.

References

[1] Nogueira R, Cho K. Passage Re-ranking with BERT. CoRR 2019.

[2] Han S, Wang X, Bendersky M, Najork M. Learning-to-Rank with BERT in TF-Ranking. CoRR 2020.

[3] Khattab O, Zaharia M. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval 2020; 39-48. doi: 0.1145/3397271.3401075

[4] Gao L, Dai Z, Callan J. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 3030–3042. doi: 10.18653/v1/2021.naacl-main.241

[5] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) 2019. doi: 10.18653/v1/N19-1423

[6] Robertson S, Zaragoza H. The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval 2009; 3 (4): 333. doi: 10.1561/1500000019

[7] Nogueira R, Yang W, Cho K, Lin J. Multi-Stage Document Ranking with BERT. CoRR 2019.

[8] Guo J, Cai Y, Fan Y, Sun F, Zhang R et al. Semantic Models for the First-stage Retrieval: A Comprehensive Review. ACM Trans. Inf. Syst. 2022; 40 (4): 42. doi: 10.1145/3486250

[9] Białecki A, Muir R, Ingersoll G, Imagination L. Apache Lucene 4. In SIGIR 2012 Workshop on Open Source Information Retrieval 2012; 17.

[10] Dai Z, Callan J. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. In Proceedings of ACM Conference 2019.

[11] Mogotsi IC, Manning CD, Raghavan P, Schütze H. Introduction to Information Retrieval. Information Retrieval 2010; 13 (2): 192-195. doi: 10.1007/s10791-009-9115-y

[12] Zheng G, Callan J. Learning to Reweight Terms with Distributed Representations. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval 2015; 575-584. doi: 10.1145/2766462.2767700

[13] Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. International Conference on Learning Representations 2013.

[14] Mihalcea R, Tarau P. TextRank: Bringing Order into Text. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing 2004; 404-411.

[15] Brin S, Page L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems 1998; 30 (1-7): 107-117.

[16] Nogueira R, Yang W, Lin J, Cho K. Document Expansion by Query Prediction. CoRR 2019.

[17] Nogueira R, Lin J, Epistemic AI. From doc2query to docTTTTTquery. Online Preprint 2019.

[18] Sutskever I, Vinyals O, Le QV. Sequence to Sequence Learning with Neural Networks. Advances in Neural Information Processing Systems 2014.

[19] Raffel C, Shazeer N, Roberts A, Lee K, Narang S et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research 2020; 21 (140): 1-67.

[20] Mallia A, Khattab O, Suel T, Tonellotto N. Learning Passage Impacts for Inverted Indexes. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval 2021; 1723-1727. doi: 10.1145/3404835.3463030

[21] Schroff F, Kalenichenko D, Philbin J. FaceNet: A Unified Embedding for Face Recognition and Clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015; 815-823. doi: 10.1109/CVPR.2015.7298682

[22] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations 2014.

[23] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L et al. Attention is All You Need. Advances in Neural Information Processing Systems 2017.

[24] Nguyen T, Rosenberg M, Song X, Gao J, Tiwary S et al. MS MARCO: A Human Generated Machine Reading Comprehension Dataset. In CoCo@ NIPS 2016.

[25] Tibshirani R. Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society: Series B (Methodological) 1996; 58 (1): 267-288. doi: 10.1111/j.2517-6161.1996.tb02080.x

[26] Craswell N. Mean Reciprocal Rank. Encyclopedia of Database Systems 2009; 1703. doi: 10.1007/978-0-387-39940-9_488

[27] Järvelin K. Kekäläinen J. Cumulated Gain-Based Evaluation of IR Techniques. ACM Transactions on Information Systems 2002; 20 (4): doi: 10.1145/582415.582418