Ranking Fusion Functions in Neural Ranking Models

The Impact of Ranking Fusion Function on Neural Ranking Models with Fast Forward Indexes

Gayeon Jee[1]

Supervisor(s): Avishek Anand[1]**, Jurek Leonhardt**[1]

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 24, 2024

Name of the student: Gayeon Jee (G.Jee@tudelft.nl)
Final project course: CSE3000 Research Project
Thesis committee: Avishek Anand, Jurek Leonhardt, Alan Hanjalic

Abstract

The research explores the impact of rank fusion functions within the retrieve-and-rerank framework with Fast-Forward Indexes. Using the BM25 sparse model for retrieval and TCT-ColBERT dense model for semantic score computation, various rank fusion functions are experimented for the interpolation stage. The interpolated rank in relation to the semantic and lexical ranks is explored. Parametric approach allows to easily adjust the influence of sparse and dense models on the final rank. On the other hand, non-parametric approach lacks flexibility and maintains an equal weight of sparse and dense scores by default. Moreover, the ranking effectiveness and latency are measured to further evaluate each function. Due to flexibility of parametric functions, convex rank fusion function and its normalized variants yield the best trade-off in latency and ranking effectiveness followed by reciprocal rank fusion. On the contrary, non-parametric functions, namely Inverse Square Rank Reciprocal, combMNZ, and Condorcet Fuse, generally performs worse.

1 Introduction

Information Retrieval (IR) is about gathering relevant documents given a query from a collection. Nowadays, the available sets of data continuously, rapidly grows, and the extensive volume of the source databases cause difficulty in efficient retrievals. There are various techniques produced for addressing this issue. Nevertheless, there are limitations in the existing architectures in aspects such as performance, speed, and required computational resources. One of the retrieval techniques is retrieve-and-rerank which first selects candidate documents based on lexical term matching. Then, it reranks them using a dense retrieval method that encodes documents and queries in a vector space where distances between points refer to relevance. [7] proposes the Fast-Forward Index framework which aims for efficient reranking by using a dual-encoder dense retriever with an optimized index. Ultimately, the final relevance rank is decided by interpolating the sparse and dense scores via a rank fusion function.

Rank fusion function controls the influence semantic and lexical scores have in the final reranking step. For some domains, semantic scores yield a more informative, optimal relevance assessment while for some it is the other way around. The ratio of each score reflected on the final rank is manipulated by the rank fusion functions. Thus, using the correct function boosts the performance of the neural ranking model.

In this research, the impact of various rank fusion functions are observed. Impact is broken down into three categories: rank, ranking effectiveness, and latency. For clarification, latency refers to how quickly the interpolated result is computed. These elements are explored with distinct test datasets and normalization technique for applicable datasets. The research is formalized under the following three subquestions:

1. How does the rankings change in relation to semantic and lexical scores using different rank fusion functions?
2. How does using different rank fusion functions impact the ranking effectiveness in different domains?
3. How does using different rank fusion functions impact the latency in different domains?

RQ 1 is answered by observing a heatmap that maps the lexical and semantic scores on the axes and the final rank after interpolation as the hue. It is discovered that the parametric functions - convex rank fusion and reciprocal rank fusion - are heavily affected by the input parameters. On the other hand, the non-parametric functions take into account both scores equally by default. Several experiments are computed for RQ 2 which evaluates the accuracy of the output ranked list via the document and query relevance. The results prove that parametric score-based functions most effectively computes the rank in general. Finally, the latency is measured for RQ 3 and all functions have a similar speed except for Condorcet Fuse which is substantially slower than other methods.

The literature first delves into the background behind information retrieval, introducing the various model types as well as the basic pipeline of Fast-Forward indexes framework in section 2. Then, it moves onto the methodology in 3 which explains the variables explored in the research. In section 4 and 5, the actual implementations and environments of the experimental runs their results are elaborated. The ethical aspects and reproducibility of the research is discussed in 6. Finally, the paper is concluded by stating the conclusions of the research questions as well as the future works in 7.

2 Background

The basic IR models are introduced in 2.1 as a background information to the Fast-Forward indexes pipeline and rank fusion function elaborated in section 2.2.

2.1 Basic Information Retrieval Models

Nowadays, there are various approaches proposed for efficient, effective information retrievals. **Lexical or sparse retrieval** simply conducts term matching on the queries and the documents and BM25 is a classical example of lexical models [12]. These models are quick, but the performance is limited as contextual information is not captured.

Semantic or dense retrieval is another common technique that complements lexical retrieval [6]. Queries and documents are encoded in a common vector space where closer representations having higher relevance. Thus, different words are in proximity if they have similar meanings unlike lexical models. Although dense retrieval generally performs better than lexical, it has disadvantages in terms of speed and index maintenance.

Hybrid retrieval [8] and **retrieve-and-rerank** [11] are approaches which combine lexical and dense models. Hybrid retrieval selects candidates using each model respectively and ranks them based on the interpolated score calculated using rank fusion functions. Since it performs dense retrieval, it inherits its disadvantages. On the other hand, retrieve-and-rerank uses only lexical retrieval for candidate collection. Then, it computes the dense scores solely on the selected documents. The Fast-Forward Index is built upon retrieve-and-rerank.

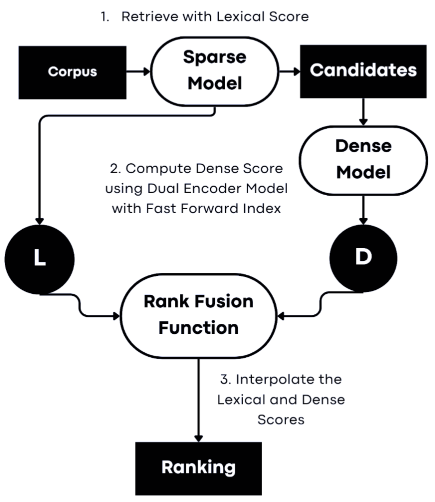

1. Retrieve with Lexical Score
Corpus
Sparse Model
Candidates
Dense Model
2. Compute Dense Score using Dual Encoder Model with Fast Forward Index
L
D
Rank Fusion Function
3. Interpolate the Lexical and Dense Scores
Ranking

Figure 1: Fast-Forward Index Pipeline

2.2 Rank Fusion Functions in Fast-Forward Index

The foundation of the Fast-Forward indexes framework is a retrieve-and-rerank model with dual encoder dense models using Fast-Forward Indexes [7]. It employs techniques such as sequential coalescing, dynamic token dropping, and early stopping to optimize performance.

Figure 1 overviews the pipeline of the Fast-Forward index model. It follows the retrieve-and-rerank procedure, and at the end, rank fusion function combines the lexical and semantic scores to get the final ranking. The focus of this research is on the last procedure, rank fusion function.

Rank fusion functions computes the final score using the lexical and dense score information as the input. Its purpose is to precisely combine the output of lexical and semantic models. The two are complementary and there is information captured by only one of the models. Rank fusion functions allow to utilize both of their advantages, delivering a more precise relevance analysis. However, there is no one perfect balance for fusing the lexical and semantic score that works for all settings. Therefore, experimentation with various rank fusion functions in distinct settings is essential for developing an optimal model.

3 Methodology

The impact of the rank fusion functions are observed via experiments. This section motivates the choice behind each of the components in the experiment. 3.1 elaborates on the retrieval model choice. Subsequently, dataset choice is explained in 3.2 and lastly the rank fusion functions in 3.3.

3.1 Retrieval Model

For first stage retrieval using a sparse model, BM25 [12] is selected as it uses a basic term matching approach. It utilizes a variation of the TF-IDF. TF, or Term Frequency, examines the number of occurrences of a term in the document also taking into account the length of the document. The score computation is in favor of shorter documents. IDF, or Inverse Document Frequency, observes the number of times a word appears over all the documents, yielding higher score for documents with rare terms.

TCT-ColBERT [6] pre-trained with MS MARCO dataset [10] is selected for second stage dense retrieval. It is a BERT-based [3] model which deeply analyzes every query-passage pair.

3.2 Dataset

The main focus of the dataset selection are diversity and presence of a dev set. A dev set is used to validate the parameters of the fusion functions which is an essential step to performance optimization. The following datasets are selected from the TREC [2] and BEIR [13] benchmarks (Table 1).

In-domain datasets refer to the corpus in the same domain as the model is trained on. Since the version of TCT-ColBERT used in this research used MS MARCO for training, TREC DL'19 MS MARCO Passage and TREC DL '20 MS MARCO Passage are in-domain datasets. Note that MS MARCO passage v1 is used instead of v2 due to the limitation in memory.

The rest are classified as out-of-domain. However, the datasets without a dev set like Arguana, CQADupStack, SCIDOCS, and Scifact are tested in zero-shot fashion as the model has no past experience in their domain.

3.3 Rank Fusion Function

Rank fusion functions can be categorized into a score-based function and rank-based function. The former directly inputs the lexical and semantic scores while the latter feeds the ranks, pruning away details of the exact scores.

Score-based Fusion

In the score-based functions, convex rank fusion [1] is tested:

$$f_{CONVEX}(q,d) = \alpha f_{SEM}(q,d) + (1-\alpha) f_{LEX}(q,d)$$

$f_{SEM}(q,d)$ and $f_{LEX}(q,d)$ correspond to the semantic score and lexical score respectively for the query q and document d pair. α is a hyperparameter that should be adjusted on the validation set.

The range of lexical and semantic scores are different, so different normalization functions are explored: identity (no normalization), min-max normalization, z-score normalization [1].

$$\phi_{MM}(f_o(q,d)) = \frac{f_o(q,d) - m_q}{M_q - m_q}$$

$f_o(q,d)$ is the score output from retrieval model o given query q and document o. M_q and m_q represent maximum and minimum score for query q.

$$\phi_Z(f_o(q,d)) = \frac{f_o(q,d) - \mu_o}{\sigma_o}$$

μ_o is equal to the mean of scores from retrieval model o and σ_o the standard deviation.

Dataset	Domain	Task	Dev Set Availability	Test Set			Query	Document
				#Query	#Corpus	Avg. D/Q		
TREC DL '19 MS MARCO Passage [2]	Misc	Passage-Retreival	Yes	43	9260	1.1	5.96	55.98
TREC DL '20 MS MARCO Passage [2]	Misc	Passage-Retreival	Yes	200	11386	1.1	5.96	55.98
BEIR NFCorpus	Bio-Medical	Bio-Medical IR	Yes	323	2622	38.2	3.3	232.26
BEIR FiQA-2018	Finance	Question Answering (QA)	Yes	648	57638	2.6	10.77	132.32
BEIR QUORA	Quora	Duplicate-Question Retrieval	Yes	10000	522931	1.6	9.53	11.44
BEIR DBPedia	Wikipedia	Entity-Retrieval	Yes	400	4635922	38.2	5.39	49.68
BEIR FEVER	Wikipedia	Fact Checking	Yes	6666	5416568	1.2	8.13	84.76
BEIR ArguAna	Misc	Argument Retrieval	No	1406	8674	1	192.98	166.8
BEIR CQADupStack English [4]	Misc	Argument Retrieval	No	1570	40221	1.4	8.59	129.09
BEIR SCIDOCS	StackEx.	Duplicate-Question Retrieval	No	1000	25657	4.9	9.38	176.19
BEIR Scifact	Scientific	Fact Checking	No	300	5183	1.1	12.37	213.63

Table 1: Domain, task, dev set availability, and corpus statistics of the selected datasets.

Rank-based Fusion

For score-based, following functions are chosen: reciprocal rank fusion (RRF) [1], CombMNZ [5], Inverse Square Rank Fusion [9], and Condorcet-Fuse [14].

$$f_{RRF}(q,d) = \frac{1}{\alpha + r_{LEX}(q,d)} + \frac{1}{\beta + r_{SEM}(q,d)}$$

$r_{LEX}(q,d)$ and $r_{SEM}(q,d)$ refer to the lexical and semantic ranking of the document d for query q. RRF is parametric function with two parameters, α and β, that should be validated.

$$\begin{aligned} CombMNZ(q,d) = 2 \times [&(L - r_{LEX}(q,d) + 1) \\ &+(L - r_{SEM}(q,d) + 1)] \end{aligned} \quad (1)$$

CombMNZ is analogous to a simple addition of the rankings. As higher ranks should yield a higher score which the documents are sorted on, it subtracts the rank from the number of candidates L and adds 1.

$$f_{ISR}(q,d) = 2 \times (\frac{1}{r_{SEM}(q,d)^2} + \frac{1}{r_{LEX}(q,d)^2})$$

ISR employs the sum of the squared reciprocal as the interpolated score. It is analogous to a non-parametric approach of RRF.

Note that the original CombMNZ and ISR computations generalize for multi set results and leverages the number of times the document appears as a candidate for each list. This is not applicable in the scope of retrieve-and-rerank as the dense model only reranks the candidates retrieved by the lexical model. Thus, both functions have been simplified to this particular setting.

Condorcet Fuse is an interpolation method motivated from a voting rule which accounts for the pairwise preference relation between all the documents per query (Algorithm 1).

Algorithm 1 Algorithm for Condorcet Fuse score computation for all documents in a candidate list L for query q

```
procedure CONDORCETFUSE($L, q$)
    $i \leftarrow 0$
    while $i < |L|$ do
        $j \leftarrow 0$
        $n \leftarrow$ normalized convex fusion of $L[i]$
        $score_i \leftarrow score_i + n$
        while $j < i$ do
            $l_i \leftarrow r_{LEX}(L[i])$                ▷ $r$ refers to rank
            $s_i \leftarrow r_{SEM}(L[i])$
            $l_j \leftarrow r_{LEX}(L[j])$
            $s_j \leftarrow r_{SEM}(L[j])$
            if $l_i > l_j \wedge s_i > s_j$ then
                $score_i \leftarrow score_i + 1$
            end if
            if $l_j > l_i \wedge s_j > s_i$ then
                $score_j \leftarrow score_j + 1$
            end if
        end while
    end while
end procedure
```

Suppose there are two documents a and b. A document gains a score if and only if a is preferred over b in both lexical and semantic profile. The normalized convex fusion score is added for each outer loop iteration to mitigate ties. As it uses a parametric function, the corresponding parameter is validated.

4 Experimental Setup

This section informs about the environment, tools used, and procedure of experiments. Section 4.1 explains the library versions, models, and datasets used. Subsequently, the methods of ranking effectiveness and latency experiments are elaborated in 4.2 and 4.3.

4.1 Setup and Tools

Specs and Tools Ranking effectiveness experiment uses Intel Xeon Gold 6248R CPU and NVidia Tesla A100 GPU, and latency experiment uses Intel Core i7-1165G7 CPU. The implementation is written in Python 3.11 with PyTerrier[1]

[1]https://pypi.org/project/python-terrier/

framework version 0.10.0, and FAST-FORWARD INDEXES[2] library version 0.2.0.

Models For the sparse model BM25, the PYTERRIER implementation is used. The BM25 score on a dataset is obtained via the `BatchRetrieve` function. TCT-colBERT from the Fast-Forward indexes[3] is used which is modified from PYSERINI[4] toolkit TCT-colBERT encoder[5] pre-trained on MS MARCO. The output representation has 768 features.

Dataset	Parameter
MS MARCO	'irds:msmarco-passage/dev/small' 'irds:msmarco-passage/trec-dl-2019' 'irds:msmarco-passage/trec-dl-2020'
NFCorpus	'irds:beir/nfcorpus/dev' 'irds:beir/nfcorpus/test'
FiQA-2018	'irds:beir/fiqa/dev' 'irds:beir/fiqa/test'
QUORA	'irds:beir/quora/dev' 'irds:beir/quora/test'
DBPedia	'irds:beir/dbpedia-entity/dev' 'irds:beir/dbpedia-entity/test'
FEVER	'irds:beir/fever/dev' 'irds:beir/fever/test'
Arguana	'irds:beir/arguana'
CQADupStack English	'irds:beir/cqadupstack/english'
SCIDOCS	'irds:beir/scidocs'
Scifact	'irds:beir/scifact/test'

Table 2: Parameters used for dataset access

Dataset Indexing All the datasets are accessed via the PYTERRIER `get_dataset` function with the parameters in 2. Datasets with multiple lines have a dev set. The first line is the command used for validation and the later lines for evaluation. Note that for MS MARCO, 3000 queries were sample using the pandas `sample` function with `random_state` 42 due to the massive corpus size.

Then, sparse and dense indexes of the retreived datasets must be built in prior to the experiment runs. As indexing is expensive, either a pre-built index is used if available or it is constructed once and reloaded. PYTERRIER provides a pre-built sparse index for MS MARCO. For other datasets, the sparse indexes are created and saved in the format of a dictionary. The Fast-Forward library is used to manufacture the semantic indexes.

4.2 Ranking Effectiveness Experiment

There are three components to the experiment pipeline: BM25 retrieval, dense score computation, then interpolation. The interpolation transformer is modified to each rank fusion function elaborated in section 3.3. Based on the interpolated score, the documents are reranked and the nDCG@10, RR@10, and MAP@100 scores are evaluated.

[2]https://pypi.org/project/fast-forward-indexes/

[3]https://github.com/mrjleo/fast-forward-indexes

[4]https://github.com/castorini/pyserini

[5]https://huggingface.co/castorini/tct_colbert-msmarco/tree/main

Evaluation Metrics Ranking efficiency is measured via the following metrics: nDCG@10, MAP@100, and RR@10. nDCG@10 is presented as the official metrics for the TREC [2] and BEIR [13]. Its provides an informative evaluation which discounts the degree of relevance based on ranks. However, MAP and RR are selected as supporting metrics for validation as [7] includes them in the performance assessment. Given that they are supporting metrics, decision and discussions will be oriented around the nDCG metric. RR is only computed over the top 10 documents as it only reflects the first relevant document found. On the other hand, MAP@100 conveys the overall score on the amount of relevant documents retrieved and their rank.

Validation Five rank fusion functions are parametric - convex rank fusion and its two normalization variants, reciprocal rank fusion, and Condorcet Fuse which uses the normalized convex rank fusion as a tie breaker. However, Condorcet Fuse is distinct from the other functions as its primary scoring mechanism is not parametric. It only uses convex rank fusion as a tie-breaker so validation is less significant for Condorcet.

All intermediate values from [0.0, 1.0] with step 0.1 are examined as α of convex rank fusion lies in the interval 0.0 to 1.0. If it is 0.0, then the semantic score equals the final score, and lexical score for 1.0. The rank change is illustrated in figure 2. It illustrates the ranks after applying the convex interpolation with varying α on 500 samples of queries from the QUORA dataset. Higher the α, documents with higher lexical score are ranked higher and vice versa.

The parameters considered for the reciprocal rank fusion are the following: (1, 1), (1, 100), (5, 10), (20, 80), (40, 60), (60, 60), (80, 20), (100, 1), (10, 5), (100, 100), (1000, 1000). Larger the parametric value, the effect of the lexical and semantic ranks diminishes to a single point [1]. Thus, the validation focuses on the parameters in the lower range in between 1-100. Low α in combination with the a high β largely reflects the lexical ranking as the effect of semantic ranking decreases and vice versa (Figure 3a and 3b). If α equals β, then both ranks are equally contributed as seen in figure 3c and 3d. Note that the figure has the ranks in the axes, so unlike figure 2, the lower values should be favored.

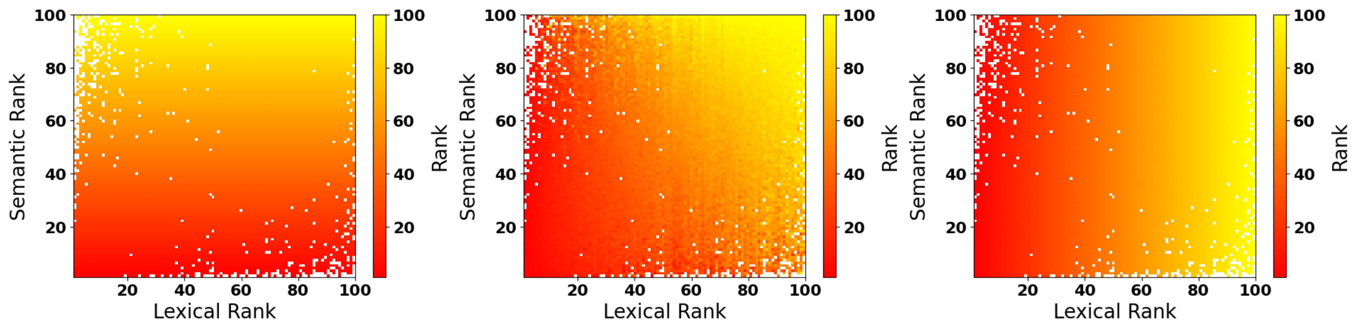
As aforementioned, there are datasets without a dev set. The parameters for these datasets are estimated according the jaccard similarity of domains [13]. The parameters of the nearest datasets are used as described in Table 3.

FiQA	Arguana, CQADupStack English
NFCorpus	SCIDOCS, Scifact

Table 3: Datasets used for validation estimation of zero-shot datasets.

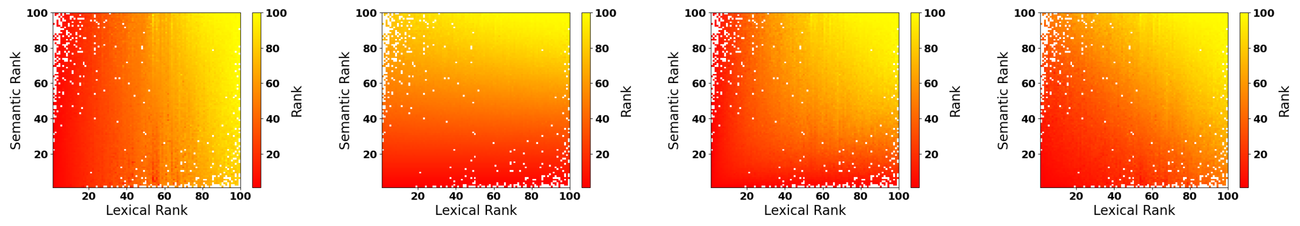
Given this setting, validation is completed via running the PYTERRIER `GridSearch` function once for each metric. In case of disagreements in the result, the parameter is selected based on majority rule but nDCG is prioritized if there is a tie.

Experiment Using the optimal parameter values retrieved in the validation stage, the experiment is conducted using



(a) $\alpha = 0.0$ (b) $\alpha = 0.5$ (c) $\alpha = 1.0$

Figure 2: Rank in relation to lexical and semantic ranks for convex rank fusion with different α using. Lexical rank on the x-axis, semantic on the y-axis, and final rank is the hue.



(a) $\alpha = 1, \beta = 100$ (b) $\alpha = 100, \beta = 1$ (c) $\alpha = 1, \beta = 1$ (d) $\alpha = 1000, \beta = 1000$

Figure 3: Rank in relation to lexical and semantic ranks for reciprocal rank fusion with different α and β. Lexical rank on the x-axis, semantic on the y-axis, and final rank is the hue.

the PyTerrier `Experiment` method and metrics from the framework. For MS MARCO, the relevance level is adjusted as 2 for MAP and RR as MS MARCO has multiple relevance levels but the metrics are binary.

4.3 Latency Experiment

In prior to the experiment runs with latency, the functions get fully loaded into memory. Due to the limitation in computational resource, 100 queries are sampled with random state 42 and 100 candidates are retrieved per query. The time is measured for running an experiment on the interpolation and evaluation metrics computation using pre-computed candidates and index scores. The experiment is conducted using the `timeit` module and obtains results for four runs, each ran for three times. The datasets tested are Arguana and QUORA. The test set is used for QUORA. These are selected based on their size due to the limitation in resource. Finally, the average over of the fastest experimental run is reported as the latency.

5 Experimental Results

This section explains the results of the experiments elaborated in the previous section, answering the three research about ranking change in relation the scores and ranks, ranking effectiveness, and latency. First, section 5.1 observes the patterns of ranks versus the lexical and semantic scores or ranks. Section 5.2 discusses the validation results, followed by the result of ranking effectiveness and latency experiments in 5.3 and 5.4.

5.1 Ranking Change in Relation to the Score and Rank

This section dives deeper into RQ 1, analyzing the rank changes in relation to lexical and semantic ranks by scrutinizing Figures 2, 3, and 4. From the QUORA dataset, 500 queries are randomly selected and and the reranking is applied with a retrieval depth 100. The ranking difference for each fusion function is visualized in a heatmap in which the lexical and semantic ranks are graphed in the axes with the rank after interpolation as the hue. Although convex rank fusion uses the scores as the input, the ranks are used in the axes for the sake of comparison.

Convex Rank Fusion Convex rank fusion is presented in figure 2. When $\alpha = 0.0$, there is a horizontal gradient and as the parametric value increments, the gradient becomes more diagonal, finally becoming vertical as α reaches 1.0. Higher the α, there is more weight to the semantic rank. Thus, the final ranks directly link to the semantic rank and ignores the lexical rank. The same pattern is observed for lower α as lexical rank dominates the interpolated rank.

Reciprocal Rank Fusion The effect of reciprocal rank fusion function is illustrated in figure 3. Lower α in combination with a relatively higher β has a vertical gradient, pruning away the semantic rank information as it diminishes the magnitude of the dense rank term with a large denominator. The same occurs for lexical ranks with higher α and lower β pair. When both values are the same, both ranks are equally accounted for. However, the final ranks are slightly different for

	MS MARCO	FiQA	NFCorpus	QUORA	DBPedia	FEVER
Convex	0.0	0.1	0.2	0.1	0.1	0.0
Convex (Min-Max)	0.2	0.5	0.6	0.5	0.4	0.1
Convex (Z Score)	0.1	0.3	0.4	0.4	0.4	0.1
Reciprocal	(100, 1)*	(1, 1)	(60, 60)	(1, 1)	(80, 20)	(100, 1)
Condorcet Fuse	0.3	0.5	0.1	0.5	0.3	0.2

Table 4: Validation result on dev sets. *First element is α and second element is β.

	TREC '19	TREC '20	FiQA-2018	NFCorpus	QUORA	DBPedia	FEVER
BM25 (No Interpolation)	0.480	0.494	0.253	0.322	0.768	0.274	0.427
Convex	0.679	0.641	0.311	0.335	0.841	0.379	0.663
Convex (Min-Max)	0.683	0.655	0.310	0.336	0.842	0.381	0.670
Convex (Z Score)	0.682	0.652	0.310	0.335	0.842	0.378	0.672
Reciprocal	0.679	0.641	0.298	0.331	0.828	0.356	0.663
Condorcet Fuse	0.629	0.592	0.300	0.329	0.821	0.337	0.582
Inverse Square Reciprocal	0.603	0.592	0.294	0.330	0.824	0.352	0.622
CombMNZ	0.623	0.597	0.300	0.329	0.820	0.337	0.579

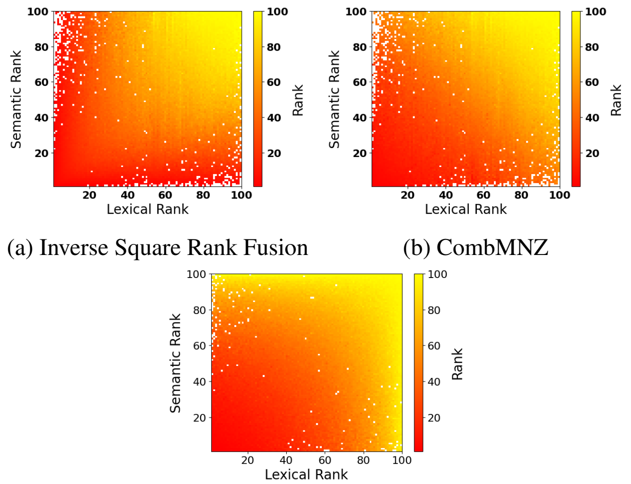
Table 5: nDCG@10 values before and after interpolation with sparse retrieval depth at 100 for datasets with a validation set. Yellow highlight marks the best performing function. Red highlight marks the functions that have statistically significant difference from the convex rank fusion function using p-value 0.05 with Bonferroni correction.

lower parametric values compared to higher. Lower values have a curved gradient in contrast to higher with has a linear gradient (Figure 3c and 3d). Smaller parametric values are likely to be dominated by high rank in one of the list while greater values mitigate them. For instance, suppose document a has rank 1 and 100 given an arbitrary query. On the other hand, there is document b with rank 40 and 40. If $\alpha, \beta = 1$, document a obtains a higher final rank while it is the opposite when $\alpha, \beta = 1000$. This example explains why figure 3d has a linear gradient pattern unlike figure 3c.

Inverse Square Rank Fusion Inverse Square Rank Fusion has a similar approach to reciprocal rank fusion (Figure 4a). Since it takes the square of the reciprocal ranks, if a term has a high rank (i.e. high denominator), its magnitude quickly diminishes. Such characteristic resembles the reciprocal rank fusion with lower, equivalent α and β.

CombMNZ Rank Fusion CombMNZ rank fusion is a simple additive approach on ranks. Thus, there is also a linear pattern (Figure 4b). This function does not leverage on or penalize higher or lower ranks.

Condorcet Fuse Condorcet Fuse is distinct from all other fusion function explored so far as it yields a circular gradient as seen in figure 4c. For a document to earn scores in Condorcet, they have dominate the other documents in both of the ranking profiles. Therefore, it is impossible for the documents further below the rank in either of the two profiles to be in higher ranks. Furthermore, heatmaps with α of 0.5 and 1.0 were additionally computed although not present in the paper. Nevertheless, the variance in α did not affect the general pattern of the graph as it is a supporting scoring metric for tie breaking.



(a) Inverse Square Rank Fusion

(b) CombMNZ

(c) Condorcet Fuse $\alpha = 0.0$

Figure 4: Rank in relation to lexical and semantic ranks for Inverse Square Rank Fusion, CombMNZ rank fusion, and Condorcet Fuse. Lexical rank on the x-axis, semantic on the y-axis, and final rank is the hue.

5.2 Validation Result

Table 4 presents the validation results. The data shows that it is more optimal for the model to grant more weight on dense scores than lexical. For convex rank fusion functions, the alpha value stays in the range 0 to 0.6, only the min-max normalized convex function exceeding 0.5. The same pattern is observed in the reciprocal fusion function. Either higher weight is placed on the dense ranks or two ranks have an equal contributions. This patterns conveys that the semantic scores are generally more informative and effective in terms of document relevance evaluation.

Datasets can be categorized into two distinct types. There are datasets that grant higher weight on the dense score and some place equal weight on the two ranks. MS MARCO and FEVER relies on the dense scores to accurately analyze document relevance as it prefers lower α values for convex and high α and lower β for reciprocal. DBPedia also favors semantic score over lexical although to a less extent than MS MARCO and FEVER. On the other hand, the remaining dataset - FiQA, NFCorpus, and QUORA - place almost equivalent weight on sparse and dense scores for the normalized and rank-based fusion functions which has an equal range for both scoring profiles. Nevertheless, the α values for convex identity and Condorcet Fuse are closer to 0.0 in general. One hypothesis for the convex function without normalization is the variance in semantic scores. Although the convex function in the Condorcet Fuse is also normalized, it has less impact as it is not the primary scoring method.

5.3 Ranking Effectiveness Result

The results from Table 5 and Table 6 explains the ranking effectiveness of each function, answering RQ 2. From the table, it is clear that the convex rank fusion delivers the best ranking effectiveness compared to all other functions. For all datasets, variations of the convex has the optimal nDCG@10. Normalization does not affect the performance significantly in general as the p-value is not below 0.05 except for the FEVER function. Although accuracy decreases for zero-shot datasets, this might be a problem with validation.

Subsequently, the reciprocal rank fusion is the second best rank fusion function. It has the least amount of times in which its difference from the convex function is statistically significant. Thus, it cannot be concluded that the difference in between the two functions did not happen by chance. The other functions - Condorcet Fuse, ISR, and CombMNZ generally result in a lower nDCG@10 compared to all other functions.

Such order among functions suggest that non-parametric functions perform worse than parametric functions when evaluating document relevance (assuming that Condorcet Fuse is non-parametric as its main scoring mechanism is not parametric). While parametric functions must be validated, it offers flexibility to adjust the weights of semantic and lexical scores in return. Moreover, convex rank fusion may be more accurate in relevance assessment than reciprocal as it directly uses the exact score.

5.4 Latency Result

	Arguana	**QUORA**
Convex	171	446
Convex (Min-Max)	190	438
Convex (Z Score)	174	445
Reciprocal	174	446
Condorcet Fuse	42108	54190
Inverse Square Reciprocal	177	437
CombMNZ	172	460

Table 7: Latency measurement in ms for 100 queries with sparse retrieval depth 100.

Rank Fusion Functions Comparing the rank fusion functions, there is no clear difference in between the rank fusion functions except for the Condorcet Fuse. Although the convex rank fusion with min-max interpolation yields relatively high latency in Arguana, it has a lower result in QUORA. As the two results conflict, it is difficult to conclude about its latency. A probable reason behind the variance is the flaky nature of latency experiments. In contrast, Condorcet Fuse is certainly slower than the other interpolation mechanisms. While the other functions simply conduct a computation for each candidate and sorting if applicable, Condorcet Fuse requires pairwise iteration of each candidate group per query to discover the preference relationship in between documents. Thus, its latency is inevitably greater than the other functions.

Datasets The latency measurement involves interpolating the scores and computing the evaluation metrics as stated in section 4.3. The speed of the evaluation is affected by the volume of the corpus qrels as it requires comparison of the relevance judgements. As QUORA has 15675 qrels in its test set while has 1406. QUORA has about 10 times more qrels in comparison to Arguana which explains why experiment on QUORA takes longer.

6 Responsible Research

This section elaborates on the responsible research feature of the project, explaining how the project complies with the FAIR principle. The responsible research of the Neural Ranking Models project mainly concerns with the reproducibility of the experimental results. Although the research also inherits the common problems of machine learning, such as excessive resource consumption and bias in models used, the elements particular to this project is the main focus.

- ***Findable and Accessible*** - the codebase is uploaded on the GitHub[6] which is publicly available.
- ***Interoperable*** - the codebase will be uploaded which can be cloned and edited. Additionally, the csv files saving the experimental results will be available in the repository to reproduce results of aggregated data. This allows

[6]https://github.com/gjee22/CSE3000-Research-Project-Neural-Ranking-Models

	Arguana	CQADupStack	SCIDOCS	Scifact
BM25 (No Interpolation)	0.342	0.280	0.147	0.672
Convex	0.363	0.319	0.155	0.688
Convex (Min-Max)	0.336	0.318	0.150	0.684
Convex (Z Score)	0.340	0.319	0.157	0.687
Reciprocal	0.357	0.311	0.151	0.668
Condorcet Fuse	0.341	0.306	0.149	0.664
Inverse Square Reciprocal	0.352	0.309	0.149	0.669
CombMNZ	0.344	0.305	0.149	0.666

Table 6: nDCG@10 values before and after interpolation with sparse retrieval depth at 100 for datasets without a validation set (i.e. zero-shot). Yellow highlight marks the best performing function. Red highlight marks the functions that have statistically significant difference from the convex rank fusion function using p-value 0.05 with Bonferroni correction.

to ensure the correctness of the code as the newly obtained result using the codebase can be cross checked with the provided data.

- ***Reusable*** - the codebase contains comments and documents explaining the experimental setup. Further details are overviewed in this paper which is publicly available in the TU Delft repository[7].

Thereby, the experiment is fully reproducible using the publicly available documentation and codebase. The results can be recovered and further analyzed.

7 Conclusions and Future Work

The research is about exploring the impact of ranking fusion functions in the retrieve-and-rerank with Fast-Forward Indexes setup. Impact is formalized into the three different subquestions stated below.

RQ 1: How does the rankings change in relation to semantic and lexical scores using different rank fusion functions?

First, the parametric functions, convex rank fusion and reciprocal rank fusion are heavily affected by the parameter values. As α is closer to 0, the lexical score is ignored and interpolation rank solely depends on the semantic score. Then, the impact of lexical score increases as α gets larger. There are two parametric values α and β for reciprocal rank fusion. The contrast between the them reflects the influence that the lexical and semantic score possess. Higher α leads to lower weight on the lexical as the additive parameter value diminishes the lexical term and vice versa. The same effect appears for β and semantic score.

The non-parametric functions yield a fixed relation in between the final rank and the ranks output by two models. Inverse Square Rank Reciprocal, CombMNZ, and Condorcet Fuse grant equal influence for both ranks in all settings. However, there are distinct characteristics among the three functions. ISR leverages the extreme ranks. Thus, maximum ranks earn an excessive score while it is the opposite for minimum ranks. CombMNZ always grants equal weight on both ranks via simple addition of the ranks. On the other hand, Condorcet Fuse makes it impossible for elements to earn high scores if it ranks lower in any of the semantic and dense score lists.

RQ 2: How does using different rank fusion functions impact the ranking effectiveness in different domains?

Convex fusion function offers the most effective ranking, followed by reciprocal. Lastly, the non-parametric functions performs the worst overall. Parametric functions have advantage over non-parametric functions for datasets evaluated more effectively by dense models. Furthermore, score-based function is better than rank-based as it preserves the score information of each model. Thereby, the distance in between each rank is fully taken into account. Non-parametric rank-based functions lacks these elements and inevitable performs the worst.

RQ 3: How does using different rank fusion functions impact the latency in different domains?

Overall, all the rank fusion functions have a similar latency excluding the Condorcet Fuse. As it is an algorithm based on the voting rule, preference relation is necessary which requires iteration through all possible pairs for the candidates. Furthermore, latency, particularly the metrics computation is affected by the size of the qrels of the datasets.

Future Work The main points of focus for future research are the hyperparameter observations and sparse and dense model extension. Given that parametric rank fusions functions are more effective to use in the framework, it implies that validation is an essential stage to optimize a neural ranking model's performance. Especially for reciprocal rank fusion functions, there is an extensive amount of possible combinations since it accepts two parametric values. In other words, there is a lot of freedom for the weight adjustment compared to a single parameter or no parameter function. In this research, only a limited set of combinations were considered due to the time limit. However, it might be possible for the function to outperform the convex rank fusion via a thorough validation procedure. In addition, [7] experiments with various sparse and dense models to evaluate the Fast-Forward framework's performance. Although only BM25 and TCT-ColBERT were used in this research, extending the research

[7]https://repository.tudelft.nl/

onto a wider range of models can yield distinct results, providing more points of analysis for the rank fusion functions.

References

[1] Sebastian Bruch, Siyu Gai, and Amir Ingber. An analysis of fusion functions for hybrid retrieval. *ACM Transactions on Information Systems*, 42(1):1–35, 2024.

[2] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, E. M. Voorhees, I. Soboroff, and Machinery Assoc Comp. Trec deep learning track: Reusable test collections in the large data regime. In *44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2369–2375, 2021. Craswell, Nick Mitra, Bhaskar Yilmaz, Emine Campos, Daniel Voorhees, Ellen M. Soboroff, Ian Voorhees, Ellen/0000-0002-5658-2308.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, October 01, 2018 2018.

[4] D. Hoogeveen, Karin M. Verspoor, and Timothy Baldwin. CQADupStack: A benchmark data set for community question-answering research. *Proceedings of the 20th Australasian Document Computing Symposium*, 2015.

[5] Antonio Juárez-González, Manuel Montes, Luis Villaseñor-Pineda, David Pinto, and Manuel Pérez-Coutiño. *Selecting the N-Top Retrieval Result Lists for an Effective Data Fusion*, volume 6008. 2010.

[6] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering, April 01, 2020 2020. EMNLP 2020.

[7] Jurek Leonhardt, Henrik Müller, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. Efficient neural ranking using forward indexes and lightweight encoders. *ACM Trans. Inf. Syst.*, 2023. Just Accepted.

[8] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. Association for Computational Linguistics.

[9] André Mourão, Flávio Martins, and João Magalhães. *Inverse square rank fusion for multimodal search*. 2014.

[10] 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.

[11] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert, January 01, 2019 2019.

[12] Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. *Okapi at TREC-3*. 1994.

[13] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models, April 01, 2021 2021. Accepted at NeurIPS 2021 Dataset and Benchmark Track.

[14] Shengli Wu and Xiaoqin Zeng. *Condorcet Fusion for Blog Opinion Retrieval*. 2012.

A Ranking Effectiveness Complete Result

	MS MARCO Psg TREC DL ’19			MS MARCO Psg TREC DL ’20			FiQA-2018			NFCorpus		
	nDCG@10	RR@10	MAP@100	nDCG@10	RR@10	MAP@100	nDCG@10	RR@10	MAP@100	nDCG@10	RR@10	MAP@100
BM25 (No Interpolation)	0.480	0.640	0.232	0.494	0.615	0.275	0.253	0.310	0.209	0.322	0.534	0.144
Convex Identity	0.679	0.833	0.336	0.641	0.803	0.390	0.311	0.380	0.258	0.335	0.540	0.149
Convex Min Max	0.683	0.828	0.338	0.655	0.805	0.397	0.310	0.382	0.257	0.336	0.541	0.150
Convex Z Score	0.682	0.831	0.338	0.652	0.831	0.396	0.310	0.380	0.259	0.335	0.541	0.150
Reciprocal	0.679	0.833	0.336	0.641	0.803	0.391	0.298	0.364	0.246	0.331	0.551	0.147
Condorcet Fuse	0.629	0.796	0.304	0.592	0.714	0.339	0.300	0.365	0.247	0.329	0.544	0.146
Inverse Square Reciprocal	0.603	0.777	0.304	0.592	0.756	0.353	0.294	0.362	0.244	0.330	0.538	0.149
CombMNZ	0.623	0.809	0.306	0.597	0.728	0.341	0.300	0.365	0.246	0.329	0.549	0.147
	QUORA			**DBPedia**			**FEVER**					
	nDCG@10	RR@10	MAP@100	nDCG@10	RR@10	MAP@100	nDCG@10	RR@10	MAP@100			
BM25 (No Interpolation)	0.768	0.758	0.727	0	0	0	0	0	0			
Convex Identity	0.841	0.834	0.804	0.379	0.702	0.242	0.663	0.649	0.614			
Convex Min Max	0.842	0.836	0.805	0.381	0.703	0.242	0.670	0.657	0.623			
Convex Z Score	0.842	0.835	0.805	0.378	0.699	0.241	0.672	0.659	0.625			
Reciprocal	0.828	0.819	0.789	0.356	0.661	0.229	0.663	0.649	0.614			
Condorcet Fuse	0.821	0.815	0.783	0.337	0.638	0.219	0.582	0.556	0.535			
Inverse Square Reciprocal	0.824	0.813	0.784	0.352	0.665	0.225	0.622	0.590	0.563			
CombMNZ	0.820	0.811	0.780	0.337	0.641	0.220	0.579	0.551	0.529			

Table 8: nDCG@10, RR@10, MAP@100 values before and after interpolation with sparse retrieval depth at 100 for datasets with a validation set

	Arguana			CQADupStack English			SCIDOCS			Scifact		
	nDCG@10	RR@10	MAP@100	nDCG@10	RR@10	MAP@100	nDCG@10	RR@10	MAP@100	nDCG@10	RR@10	MAP@100
BM25 (No Interpolation)	0.342	0.226	0.237	0.280	0.285	0.252	0.147	0.254	0.097	0.672	0.632	0.627
Convex Identity	0.363	0.241	0.252	0.319	0.331	0.289	0.155	0.275	0.104	.688	0.653	0.644
Convex Min Max	0.336	0.220	0.232	0.318	0.331	0.290	0.150	0.266	0.101	0.684	0.650	0.642
Convex Z Score	0.340	0.223	0.235	0.319	0.331	0.290	0.157	0.278	0.105	0.687	0.651	0.642
Reciprocal	0.357	0.232	0.246	0.311	0.322	0.280	0.151	0.269	0.102	0.668	0.635	0.620
Condorcet Fuse	0.341	0.224	0.236	0.306	0.317	0.279	0.149	0.268	0.101	0.664	0.626	0.618
Inverse Square Reciprocal	0.352	0.227	0.241	0.309	0.319	0.278	0.149	0.265	0.100	0.669	0.643	0.625
CombMNZ	0.344	0.226	0.239	0.305	0.315	0.278	0.149	0.271	0.101	0.666	0.631	0.620

Table 9: nDCG@10, RR@10, MAP@100 values before and after interpolation with sparse retrieval depth at 100 for zero-shot datasets