ENHANCE SEARCH RELEVANCE VIA DYNAMIC MODEL WEIGHTING USING LEARNING TO RANK

T.J.J.M. REINTJES

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

STUDENT NUMBER

u497970

COMMITTEE

dr. Afra Alishahi
dr. Eva Vanmassenhove

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

January 15th, 2024

WORD COUNT

8589

ACKNOWLEDGMENTS

I would like to thank Dr. Afra Ashahi for her valuable guidance during this research. Her insights and suggestions were essential in shaping both the direction and the outcome of this study.
Special thanks go to Rabobank for providing the case study that forms the foundation of this research. The data and real-world context supplied by Rabobank were essential in bringing societal relevance to this thesis. The support and computational resources provided by the bank were beneficial.
Within Rabobank, I want to express my gratitude to Martijn Di Bucchianico, Lars Hanegraaf, and Ward Jongelie for their support, contributions, and critical reviews. They were crucial in enhancing the quality of this study and a constant source of motivation.

ENHANCE SEARCH RELEVANCE VIA DYNAMIC MODEL WEIGHTING USING LEARNING TO RANK

T.J.J.M. REINTJES

Abstract

The challenge of obtaining accurate search results persists in Information Retrieval Systems (IRSs), prompting this study to enhance their precision. Therefore, this study explores the integration of Learning to Rank, specifically the LambdaMART algorithm, into Dynamic Model Weighting (DMW) to enhance the relevance ranking of cross-industry news articles in custom IRSs. Addressing the limitations of fixed-weighting models in IRSs, we propose a dynamic model using feature importances as proportional weights. Two datasets of news articles with relevance scores on a 0–100 scale, based on sentiment, subject, and credit risk scores, are employed. One dataset includes expert-annotated labels, and the other dataset includes user-annotated labels to evaluate the performance of the dynamic model and the real-world representation of the expert set.

While the integration of LambdaMART and DMW yields marginal improvements in accuracy (from 0.76 to 0.77) and F1 score (from 0.67 to 0.69), further analysis reveals significant shifts in the relevance score distribution. These shifts, unnoticed by standard metrics due to the study's reliance on binary classification, are attributed to the changes in the weights assigned to each portfolio. Density distributions display these distribution shifts, highlighting the nuanced impact of the dynamic model on data distribution. Basic statistics, such as mean and median, show minimal change, and the standard deviation remains stable, masking the distributional shifts caused by the adjusted weights on individual scores.

The study concludes that while DMW integrated with LambdaMART shows potential for improving IRSs, the results emphasize the need for future research using multi-level relevance labels to evaluate the model's performance in depth. This approach aims to enhance user experience by achieving more accurate ranking within the relevant scored search results.

1 DATA SOURCE, ETHICS, CODE, AND TECHNOLOGY STATEMENT

The used dataset is the property of Rabobank Cooperation, and no new data is collected from human or animal subjects. The data is acquired with the explicit consent of the data owner throughout internal usage, and all figures and images created by the author are original, using draw.io and Python, unless otherwise stated. Any known biases within the dataset are discussed in the context of representativeness and inclusivity. The code used for analysis is a combination of original work and adapted segments from recognized open-source libraries, and all software and tools used are listed with their respective version numbers in the attached GitHub repository[1]. Furthermore, this study has been conducted with the assistance of Thesaurus[2] as a paraphrasing tool. Spell checking and grammar were performed using both LanguageTool[3] and OpenAI[4]. The document was typeset using LaTeX, which also facilitated reference management.

2 INTRODUCTION

2.1 *Problem Statement*

Every day, billions of individuals and organizations rely on Information Retrieval Systems (IRSs) to access and process vast amounts of data relevant to their diverse and complex query requests (Jansen et al., 2008, Tekli, 2022). Despite significant advancements in IRSs, a continual challenge remains: achieving high relevance in search results on the first attempt, a phenomenon known as search relevance (Büttcher et al., 2010). The challenge in achieving optimal search relevance is multifaceted and is exacerbated by the growing volumes of data, the complexity of different contexts, and the diversity of user intents in search query results (J.-T. Huang et al., 2020, Tekli, 2022). Thus, the issue is not merely a technological challenge but a crucial factor in how effectively IRSs meet user needs and expectations and decrease inaccurate results (Ceri et al., 2013, Zhu, 2023). The search relevance strongly depends on the user scenario and is therefore a relative concept. We define a user scenario as the situation where the relevance of the results from the same query can be interpreted differently depending on the user and user interests. These dependencies may include, for instance, those related to timing or the user's chosen topic.

[1] https://github.com/TJJMReintjes/LTR_in_DMW_Thesis_TiU
[2] https://www.thesaurus.com/
[3] https://languagetool.org
[4] https://chat.openai.com

Common IRSs, such as Google, tackle these user-scenario challenges by focusing on achieving an average and general user experience, based on generalizing how to handle search queries and including the behavior of users on the internet, such as clicks. However, this is not the case for custom IRSs; search engines within companies, internal systems, or within certain websites. Rather, these IRSs have the purpose to be adapted for specific tasks or use-cases (Vicente-López et al., 2014).

Custom IRSs can potentially enhance adaptability by integrating multiple advanced Natural Language Processing (NLP) models, each with distinct strengths and weaknesses in various contexts (Zhu, 2023). The objective is to precisely optimize user experiences, and this is manifested in the utilization of different NLP models in custom IRSs. These models can be employed in either a pipeline architecture, where data flows sequentially through a series of models, or in parallel ensemble architectures, where multiple models operate simultaneously. In this study, we focus on ensemble architectures, as they are more common due to their better performance compared to individual models (Feng et al., 2018). Additionally, they raise important questions about determining appropriate weights for each NLP model to contribute to relevance scoring in relation to a user scenario. This highlights the pressing need for advanced weighting methods within custom IRSs (Gupta and Bhatia, 2021).

Fortunately, Dynamic Model Weighting (DMW) emerges as a promising solution to this challenge. DMW dynamically adjusts model weights based on various factors, including feature importances (Feng et al., 2018). In the context of ensemble architectures, DMW can enhance model collaboration by adapting the weights assigned to each model based on their individual contributions and relevance to the current task. To extract feature importances related to a relevance problem, Learning to Rank (LTR) can be employed (Shi et al., 2020). LTR algorithms specialize in ranking problems, aiming to discern complex relations between a search query or topic and true relevance, ensuring the proper ranking of search results. In this study, our focus is on an LTR algorithm named LambdaMART. As one of the state-of-the-art LTR algorithms, it offers a flexible solution for a weighting system by capturing relevance patterns (Casalegno, 2022).

Considering the above, this research will question whether combining DMW with LambdaMART will yield a potential breakthrough in the field of NLP and custom IRSs. See Section 4 for a more in-depth explanation of the literature.

2.2 *Scientific and Societal Impact*

This exploratory study is crucial for pushing the boundaries of search relevance optimization and addressing the ever-evolving demands of users seeking accurate, relevant results. We aim to explore the development of a dynamic system using DMW that can match diverse user expectations, a capability increasingly important in an era of information overload (Tekli, 2022).

Furthermore, as custom IRRs are utilized, for instance within companies, an improved search relevance is likely to enhance decision-making by facilitating finer strategic planning or financial adjustments. Consequently, this reduces the risk of decreased productivity and potentially missing opportunities, ultimately minimizing the likelihood of inaccurate decision-making (Gul and Al-Faryan, 2023).

2.2.1 *Use-case Rabobank*

For this research, a real-life use case will be employed to demonstrate the societal relevance of this study and to conduct the evaluation of a proposed architecture. At Rabobank, professionals in various roles rely on timely and relevant information from news articles to make critical decisions. However, conventional IRSs such as Google News are unable to deliver news linked to the requirements of bankers' portfolios. Therefore, Rabobank employs its own custom application, NewsBird, to aggregate news subscriptions into a single application via an API and deliver them processed and customized to the bankers.

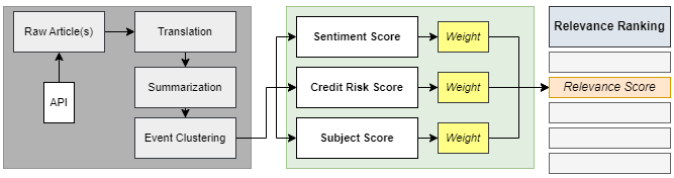

Figure 1: Architecture Rabobank's Application NewsBird

In this use case, each banker requests articles via the API based on their portfolio, which contains a vast number of queries based on the content of their portfolio. A portfolio consists of a search query and optional keywords or topics, where the search result, i.e., article, needs to match. As seen in Figure 1, NewsBird pre-processes the raw articles by cleaning the raw data, followed by summarizing, translating, and clustering the articles. This pipeline is connected to an architecture of ensemble NLP models; FinBERT for sentiment analysis and spaCy for both credit risk and subject relation analysis (Honnibal and Montani, 2017, A. H. Huang et al.,

2022).

For the user experience, it is desirable to rank the articles by relevance, from the most relevant to the least relevant. Therefore, NewsBird employs a relevance score, which can be sorted in various ways, including descending, to obtain this ranking. This score is aggregated using the three models and three fixed model weights, determined by experts, resulting in a relevance score independent of the IRS's user (Figure 1). However, following feedback, several users often experience inaccurate results; articles that are scored as relevant but are irrelevant. This also validates the need for a dynamic system, as discussed in Section 2.1.

3 RESEARCH QUESTION(S)

The study's baseline model is the current ensemble architecture with fixed weights (Figure 1), which can be compared to a dynamic model that includes dynamic weights using LambdaMART's feature importances. The evaluation is based on two separate test sets: one with expert-based annotations and one with user-based annotations. Additionally, we assess whether the expert-based annotations accurately represent the annotations of users. For further elaboration on architectural design and evaluation methodology, see Section 5.

Based on the current background, we define the following research question:

How can the integration of Learning to Rank (LTR) in Dynamic Model Weighting enhance the search relevance of cross-industry news articles?

To answer this research question, the study can be split into two subquestions:

1. *What are the feature importances, identified by the LambdaMART algorithm, that can be used in Dynamic Model weighting for relevance ranking?*

First, we aim to obtain all the feature importances per portfolio (user) of the model scores in the ensemble architecture. This is achieved through the LambdaMART algorithm, using the true relevance labels of both sets as a reference point. Subsequently, feature importances can be employed as proportional weights in the dynamic model by averaging the weights per portfolio, resulting in a custom relevance score per article per portfolio. The hypothesis is that weights change per portfolio, i.e., per user.

2. *Does Dynamic Model Weighting, using extracted features, enhance relevance ranking compared to a fixed-weighting approach?*

Secondly, we aim to evaluate if the computation of the new relevance scores using the dynamic weights in the dynamic model leads to better results, using the performance of the baseline model as a reference. Both models are tested for both expert- and user-based annotated test sets.

3.1 *Main Findings*

The results showed that the dynamic model with LambdaMART's feature importances resulted in minor improvements in relevance scoring compared to a baseline model, with an accuracy increase from 0.76 to 0.77 and an F1 score increase from 0.67 to 0.69. Despite these improvements not being significant, the research does identify a notable shift in the distribution of relevance scores. This shift, not fully captured by the standard performance metrics, indicates an underlying change in the computation of relevance scores. The study's reliance on binary classification oversimplifies the complex concept of relevance. Future research would benefit from a multi-level labeling system to facilitate a more nuanced and precise assessment of the model's performance. This approach may enhance the user experience by providing a more accurate ranking within already relevant scored articles.

In summary, the integration of LambdaMART in DMW could have potential benefits. However, enhancing relevance in IRSs with binary labels is limited by the complex concept of relevance.

4 RELATED WORK

4.1 *Definition of Relevance*

Fundamentally, the definition of relevance in information retrieval (IR) is intrinsically subjective and heavily dependent on the user's needs and context. As noted by Saračević (2007), relevance is a user-centered concept rather than an inherent property of information. This underscores its user-dependent definition. What is considered relevant to one user in a specific context may not be relevant to another. Moreover, according to Mizzaro (1998), relevance is shaped by the situational and cognitive needs of the user rather than a binary concept.

Furthermore, in the case of news, following Joachims et al. (1996), relevance goes beyond topical alignment and includes a high dependency

on time and source credibility. In extension of this, Kanhabua and Anand (2016) noted that IR must account for the temporal aspects of relevance, as information that is relevant at one time may become obsolete or less pertinent at a later date. This is particularly evident in fast-evolving fields like news, finance, or medical research.

Thus, the concept of relevance in IRSs is both subjective and time-dependent, necessitating a nuanced understanding of user needs, context, and the temporal nature of information.

4.2 *Advanced IRSs*

Fortunately, IRSs have undergone significant transformations with the advent of this challenge and big data. Early works, including those by Jansen et al. (2008) and Ceri et al. (2013), establish the basis by describing the progression from simple text retrieval techniques to complex systems that can handle large datasets. This development highlights the growing complexity of IRSs and lays the groundwork for the difficulties and advancements that the field will encounter in the future.

4.3 *Dynamic Model Weighting*

As mentioned before, this study specifically focuses on IRSs with ensemble architectures. The field of machine learning, particularly in IR, has increasingly relied on ensemble learning architectures, known for outperforming individual models (Feng et al., 2018, Jain, 2023). The impact of model weighting in these architectures is significant, as proven by customized weighting systems and algorithms that optimize weight distribution among models using model features. For instance, Troussas et al. (2020) achieved promising results by combining multiple models using a customized voting system. This led to the exploration of Dynamic Model Weighting (DMW), where research by LiKewen et al. (2015) exemplifies the adaptability of ensemble models to different contexts through dynamic weight adjustment, enhancing predicting accuracy.

4.4 *Feature Importances and LambdaMART Algorithm*

The complex systems in IR often use several NLP techniques. According to Dogra et al. (2022), during the last decade, there have been significant efforts in text document analysis where feature importance analysis has been performed. Features importances can be used for customized systems,

as described in Wongthongtham et al. (2018), where several advanced text features were used to qualify the quality of the search results.

To improve the performance of IRSs, Learning to Rank (LTR) was developed (Trotman, 2005), which has the ability to learn from data to improve ranking accuracy. The early models of LTR are often evaluated in depth, as shown in studies by Shi et al. (2020), resulting in extensive evaluation of LTR algorithms: RankNet, LambdaMART, and LightGBM, that greatly improve ranking processes. According to extensive research by Hu et al. (2019) and Shi et al. (2020), LambdaMART's dominance is evident, showing it is a state-of-the-art algorithm in a variety of ranking scenarios.

LambdaMART can be traced back to two key algorithms: LambdaRank and Multiple Additive Regression Trees (MART). LambdaRank, introduced by C. Burges et al. (2005), is a method that extends traditional gradient boosting techniques to ranking problems by using a novel gradient computation method based on pairwise differences in scores and relevance-sensitive loss functions (C. Burges et al., 2005). MART, a form of gradient boosting involving decision trees, had already established itself as a powerful approach for classification and regression tasks. The integration of these two methods led to the development of LambdaMART, effectively combining the ranking-focused gradient approach of LambdaRank with the robust tree-based learning of MART (C. J. Burges, 2010).

Regarding LambdaMART's abilities, it can be used for both binary and multi-level labels and also has the flexibility to use both numerical and categorical features, which may be crucial if different types of data need to be integrated (Li, 2015).

4.5 *Shortcomings and Combined Application*

Using LambdaMART for dynamic model weighting in IRSs does come with certain shortcomings. LambdaMART, being an advanced machine learning algorithm, can be computationally intensive, especially with large datasets and numerous features. The performance of LambdaMART heavily depends on the quality and representativeness of the training data. Inaccuracies or small biases in the data can lead to poor model performance. The complex nature of the model, which involves multiple decision trees, can make it challenging to interpret how decisions are made, which is a common issue in many ensemble methods (Li, 2015).

Despite these shortcomings, there remains an interesting research gap in the combined application of LTR and DMW, particularly in the context of custom IRSs. This study aims to bridge this gap by integrating LambdaMART into a DMW system, leveraging LTR's strengths in feature identification for ranking and enhancing adaptability through DMW. This

integrated approach could set new benchmarks in IR optimization and relevance determination, leading to significant developments in IRSs.

5 METHODOLOGY

5.1 *Overview*

In response to the research gap identified in Section 4, this study developed an experiment to investigate whether integrating feature importances, extracted by LambdaMART from ensemble models, into a dynamic architecture (model) could improve search relevance. In this case, improved search relevance can be achieved by a more accurately computed relevance score related to a specific portfolio. Initially, we established a baseline model that calculates relevance scores using fixed weights for each portfolio. The effectiveness of this model can be assessed using two test datasets with true labels, one with expert annotations and one with user annotations. Both datasets are binary labeled, resulting in the task of this study being a binary classification study.

Next, we extracted the feature importances of the scores of three models relative to the true label by training the LambdaMART algorithm. To enhance the reliability of LambdaMART, a validation set was utilized for parameter tuning of the model.[5] The extracted feature importances can be used as proportional weights in the dynamic model with DMW. This involves averaging the importance of each feature across portfolios and normalizing these averages to ensure their total sums up to one.

These calculated proportional weights were subsequently applied as dynamic weights in a dynamic model architecture derived from the baseline model. As the aim of the study was to adapt the computed relevance score for each portfolio, the performance of this dynamic model was again evaluated using labels from expert annotations and user annotations, providing a comprehensive measure of its performance.

5.2 *Implementation and Code Reusability*

A significant contribution of this study is the development within Rabobank's codebase. This specific codebase is not permitted to be publicly shared. Therefore, the code of the proposed framework in this study has been made generic, modular, and is available on GitHub[6] for public access, including

[5] Note: To train and optimize the model, only the training and validation sets from the split of the expert dataset are used. To evaluate the computed relevance score in both the baseline and dynamic model, both test sets are used.

[6] https://github.com/TJJMReintjes/LTR_in_DMW_Thesis_TiU

a mock dataset. This allows easy adaptation and extension to various use cases beyond the scope of this research.

5.3 *Dataset Description*

The dataset (Set 1) used for training, validation, and testing consists of 459.390 unique articles in .csv (or .json) format, labeled by experts on binary relevance (non-relevant and relevant). It includes fifty-five fields, such as portfolio identifiers (`portfolio_ids`), model scores (features), and a computed relevance score. The structure is as follows:

- **portfolio_id:** Identifier of the portfolio owner; relates to a group of users, i.e., department (alias: Partitionkey).
- **search_id:** Identifier for a unique search result with the used query (alias: Rowkey).
- **original_query:** Original query from which the article is retrieved.
- **query_id:** Unique identifier of the original query.
- **department_id:** Identifier of the department or user group within the bank where the portfolio is located.
- **topic:** Topic of the article.
- **sentiment_score:** Computed using the NLP model FinBERT (A. H. Huang et al., 2022), ranging between -1 and 1.
- **credit_risk_score:** Computed credit risk score, based on cosine similarity of keywords and content of the article.
- **subject_score:** Computed score on how much the query matches the title and content of the article using cosine similarity.
- **relevance_score:** Computed using sentiment, credit risk, and subject score with fixed weighting of 0.5, 0.3, and 0.2, respectively.
- **true_label:** Annotations retrieved from experts or users (depending on the set).

In Set 1, the true relevance labels are annotated by experts using batches of articles, employing various techniques, including statistics, sampling, and using a threshold for a minimum relevance score.

Via the portfolio identifiers (`portfolio_id`), we can cluster the search results per portfolio, which relates to a specific user group or department. The total relevance scores per article are computed by aggregating sentiment, credit risk, and subject-relation scores, resulting in a 0-100 scale. These separate model scores are computed according to Section 5.5. We assume that, following experts of Rabobank analytics department, the computed relevance score may be considered relevant to a user when the score is greater than or equal to 40 percent.

To maintain the structure of the data regarding `portfolio_id` and `query_id`, a nuanced approach to dataset splitting is needed. A conventional random split would not suffice, as not every `portfolio_id` contains an equal number of queries, which would result in splits within a portfolio's query data. Therefore, to avoid articles from the same query being split across training and validation sets, we use a custom `GroupShuffleSplit`[7] method.

The dataset is first divided into two segments: a larger segment, accounting for approximately 90% of the data, and a smaller segment for the remaining 10%. This first division aimed to create a combined training and validation set, and a separate test set, ensuring that no results of a query were split up between these sets. The larger segment was further subdivided using `GroupShuffleSplit`, allocating approximately 22.2% of its records to the validation set to perform parameter tuning and obtain a reliable evaluation of the model's performance. The remaining data is used for the training set. This two-step process resulted in an approximate distribution of 69.9% for training, 19.8% for validation, and 9.3% for testing.

This methodology, while deviating slightly from the traditional 70-20-10 split, was crucial to uphold the study's integrity. It also ensured that the model was evaluated against completely unseen `query_id` of a certain `portfolio_id` in the test set, a critical aspect to test extracted feature importances of a specific `portfolio_id` on results of an unseen `query_id`. This data structure is clarified in Figure 2 below.

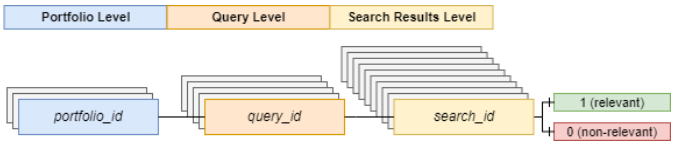

Figure 2: Sample of the Dataset's Hierarchical Structure

[7] Sklearn.model_selection.GroupShuffleSplit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GroupShuffleSplit.html

Furthermore, we use a separate test-set (set 2) of 30789 unique articles, which are labeled by users, retrieved from feedback during use of the application. This separate test set contains overlapping `portfolio_ids` as in set 1, which is necessary for fair evaluation. We use this separate set to evaluate if the train data is representative of real-life scenarios, as it might not fully capture variability of real-users, and thus to rule out any biases. This approach helps in ensuring that the model remains reliable and accurate when deployed in real-world scenarios.

The label distribution of all sets, showed in Figure 3, suggest that the dataset is relatively balanced, resp. ca. 53.4% vs. 46.6% for training. The validation and test set are slightly different distributed, because of the difference of the amount of articles among portfolios, resp. 52.1% vs. 47.9% and 52.9% vs. 47.1%. Utilizing a balanced dataset may prevent the model from becoming biased towards the majority class, enabling fair evaluation. Nonetheless, conducting an experiment on an imbalanced dataset may assess the model's ability to handle real-world scenarios where relevant articles could be significantly outnumbered by non-relevant ones. To tackle that, the user test set (Figure 4) is a bit more imbalanced by having more non-relevant than relevant articles, resp. ca. 59.1% vs. 40.9%. Thus, we can evaluate if the train data is decent enough to still preform well compared to a slightly imbalanced user test set.

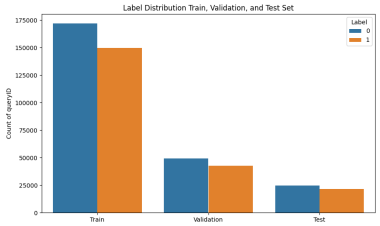

Figure 3: Label Distribution Expert Annotated Sets (non-relevant (0) and relevant (1))

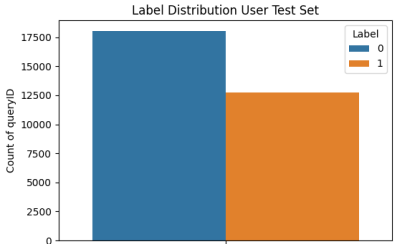

Figure 4: Label Distribution User Annotated Test Set (non-relevant (0) and relevant (1))

5.4 *Preprocessing*

The dataset is obtained from a pipeline that includes the necessary preprocessing steps to ensure a proper dataset.

CLEANING Eliminating extracted articles containing empty fields, which are important in future steps, is crucial to prevent any potential issues.

Additionally, special characters and HTML coding tags are removed to obtain clean text data.

TRANSLATE The translation is necessary because the credit risk model in the ensemble architecture requires English content, aligning with the language preference of the end-user. Since both datasets are multilingual, the articles' contents are sent to a translation API[8], which converts the content into English.

SUMMARIZE The translated articles are processed through several SpaCy models to create a summary by identifying key sentences and phrases (Honnibal and Montani, 2017). This is achieved through techniques like sentence tokenization and word frequency analysis.

CLUSTERING The application of clustering is crucial for removing duplicate articles about the same topic from different news vendors. As a result, the final set exclusively consists of articles that appear once, covering unique topics. This refined set is then utilized within the ensemble architecture to calculate the three model scores for relevance scoring.

5.5 *Computation of Model Scores* [9]

SENTIMENT SCORE For sentiment analysis, FinBERT is used, which is an NLP model specifically designed for financial sentiment analysis (A. H. Huang et al., 2022). Thus, the model assesses the sentiment (positive or negative) of the article from a financial perspective. The sentiment score is on a scale from -1 to 1.

CREDIT RISK SCORE The credit risk score is determined through a custom calculation, relying on a predetermined list of specific keywords accompanied by associated weights relevant to finance and risk. Developed by Rabobank's risk management department, this list is designed to assess potential impacts on credit risk using cosine similarity. The resulting credit risk score is presented on a scale ranging from 0 to 30.

SUBJECT SCORE The subject score is calculated by combining the frequency of the used query in the article's title and finding the direct cosine

[8] Azure Cognitive Functions: https://www.microsoft.com/en-us/translator/business/translator-api/

[9] Note: it is assumed that the methods used to calculate these scores are accurately defined for their application in this study. Detailed discussions about the accuracy of these score computations are not included in this study, as the models are not publicly shared.

similarity of the query to the article's title. The subject score is on a scale from 0 to 1.

RELEVANCE SCORE The total relevance score is computed by combining the sentiment, credit risk, and subject scores, applying weights to each of them. In the current baseline model, the fixed weights are rounded numbers: 0.50 for sentiment, 0.30 for credit risk, and 0.20 for subject. The relevance score is on a scale from 0 to 100.

5.6 *Explorative Data Analysis (EDA)*

We conduct an EDA to obtain an overview of the distributions of the model scores utilized for each article, as shown in Figure 7 in Section 6. This is done to gain insights into the model scores we use with LambdaMART. We want to determine if the scores are well-differentiated and do not contain any outliers or skewness that might affect model performance. Since we are interested in feature importances, it is essential to understand how changes in feature values influence the model's predictions. Constant features have no discriminatory power; they do not contribute to differentiating between different outcomes. Therefore, variability in scores allows the model to discern patterns and make distinctions between different scenarios.

5.7 *Baseline Model*

The baseline model is the current model with fixed weights as structured in Figure 1. This serves as the reference model to which the dynamic model will be evaluated. To obtain performance insights of the baseline model, we assess the current fixed weighted computed relevance score using the test set with expert labels, representing 9.3% of the total set. This is followed by evaluation on the user-labeled test set. Then, we compute a confusion matrix for both and determine the corresponding precision, recall, and F1 score for reliable evaluation.

5.8 *Dynamic Model: LambdaMART Algorithm*

As mentioned in Section 4, LambdaMART is derived from the LambdaRank framework, being a variant of Multiple Additive Regression Trees (MART). It compares the relevance degree for every pair of samples in a query group, in our use case, a user's portfolio, and calculates a gradient for each pair. LambdaMART is typically used for ranking with graded relevance, but its

application also extends to scenarios that diverge from these traditional ranking studies, such as extracting only feature importances based on binary true labels per group (in our case, per portfolio). The primary goal is to differentiate between relevant and non-relevant articles, rather than to establish a precise ranking order. This unique application of LambdaMART, although different from its conventional use, effectively leverages its robust framework for feature importance analysis ("Learning to rank — XGBoost 2.1.0-DeV documentation", n.d.). Thus, it utilizes features such as sentiment, credit risk, and subject score to rank the articles, and post-training, it provides insights into the feature importances, indicating which features predominantly influence the binary relevance classification.

For the specific implementation of LambdaMART, the choice between LightGBM and XGBoost as a gradient boosting framework is essential. LightGBM, developed by Microsoft[10], is chosen for its efficiency with large datasets, facilitated by a histogram-based algorithm that optimizes memory usage and computational speed (Ke, 2017). This efficiency is beneficial given the extensive dataset size. Furthermore, LightGBM is compatible with standard Python data structures, such as Pandas DataFrames, which simplifies the data handling process. This contrasts with XGBoost, which necessitates data conversion to its proprietary DMatrix format, adding complexity to data preprocessing (Chen and Guestrin, 2016). In terms of performance and scalability, both frameworks exhibit high levels of accuracy and robustness. However, LightGBM is more preferred as it demonstrates faster training times. This performance advantage, coupled with its straightforward methods for feature importance analysis, makes LightGBM particularly suitable for our study's objectives.

According to "Learning to rank — XGBoost 2.1.0-DeV documentation" (n.d.) and Casalegno (2022), LambdaMART's performance depends on various metrics such as Normalized Discounted Cumulative Gain (NDCG, rank:ndcg) and Mean Average Precision (MAP, rank:map). Despite NDCG being the standard metric in multi-level relevance scenarios, its application in binary cases remains viable.

To clarify, each portfolio consists of multiple queries yielding daily search results of varying quantities. The adaptation of LambdaMART in this context, facilitated by the LightGBM framework, aims to differentiate between relevant and non-relevant articles within a query group of a portfolio rather than establishing a precise ranking order. This structure was visualized in Section 5.3 in Figure 2, and the corresponding methodology is elaborated in Appendix 10.

[10] https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRanker.html

The iterative construction of decision trees in LambdaMART using LightGBM is oriented towards maximizing NDCG. The 'lambdas', serving as gradients in the boosting process, are recalculated to optimize the NDCG score, reflecting the potential change in NDCG resulting from a swap in the ranking of any two items. The calculation of these lambdas takes into account the binary nature of the relevance labels, focusing on maximizing the accuracy of correctly identifying relevant and non-relevant (Wu et al., 2010), as seen in Equation 1 below.

$$NDCG = 1 - \frac{DCG}{IDCG} \tag{1}$$

where:

- $NDCG$ represents the Normalized Discounted Cumulative Gain.
- DCG Discounted Cumulative Gain, measures the gain from the item based on its position in the result list.
- $IDCG$ Ideal Discounted Cumulative Gain, is the maximum possible DCG.

The loss function is relevant when evaluating the algorithm's ability to learn an optimal ranking based on the provided features and labels. If ranking performance is not a central concern of the study, and the focus is solely on extracting feature importances, the discussion of the loss function can be omitted from the evaluation.

5.8.1 *Extract Feature Importances*

This study uses a .csv file and converts it to a Pandas DataFrame, as LightGBM is directly compatible. For computational efficiency, Pandas on Spark is employed, as it significantly enhances speed and reduces memory usage. This choice is made due to the memory inefficiency of Pandas DataFrames. (PySpark[11])

Via LightGBM, LambdaMART has a direct functionality to extract feature importances, representing the contribution of each model score (sentiment, credit risk, and subject score) to the true label. Each iteration involves the algorithm selecting splits in the decision trees based on the features that most effectively improve NCDG. Features that are more frequently used in splits that significantly improve the ranking metric are deemed more important.

[11] https://spark.apache.org/docs/latest/api/python/user_guide/pandas_on_spark/index.html

5.8.2 *Architectural design*

To obtain proportional weights per portfolio, the extracted feature importances should be averaged and normalized per `portfolio_id`, to ensure the total sum of averages is equal to 1. After that, the weights can be used in the dynamic architecture as showed in Figure 5.

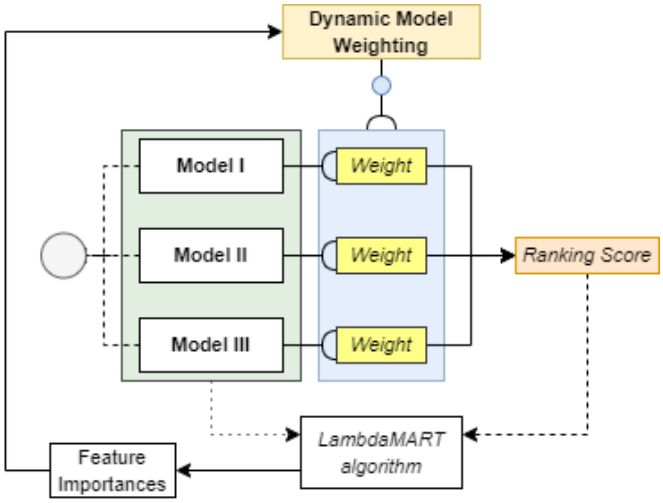

Figure 5: Architecture of the Dynamic Model

5.8.3 *Hyperparameters*

According to Dubey (2022), the performance heavily depends on chosen parameters. For this reason, considering time constraints, a limited GridSearchCV from scikit-learn is employed. This method systematically explores the combinations of parameter values using the validation set to ensure the model's robustness. The range of hyperparameters is chosen based on the default value according documentation[12] and surrounding values, these are defined in Table 1.

Table 1: Grid Search Parameters for LambdaMART

Parameter	**Values**
learning_rate	0.01, 0.05, 0.1
n_estimators	50, 100, 150

After the grid search, the best available hyperparameter are identified and returned. These chosen hyperparameters are then used to train the final LambdaMART model; at vast learning rate and number of estimators. Nguyen et al., 2016

[12] ("LightGBM.LGBMRanker — LightGBM 4.1.0.99 Documentation", n.d.)

5.9 *Evaluation*

As earlier explained, we use a dual testing-framework by predict relevance on both expert- and user-based test set, to evaluate the robustness and possible bias of the dynamic model. For the evaluation of the dynamic model with the user set, we use the computed proportional weights during set 1 corresponding to a `portfolio_id`. As the set 1 and set 2 have overlapping `portfolio_ids`.

Next, we preform an in depth error analysis, this includes a small impact assessment; to inspect whether the dynamic model's results are represented enough via the current metrics or if there are biases with reasoning present. This is essential for understanding of the model for its application.

5.9.1 *Metrics*

We compare the baseline model with the dynamic model as visualized in Figure 6. For binary classification studies, a confusion matrix provides a decent overview of the performance.

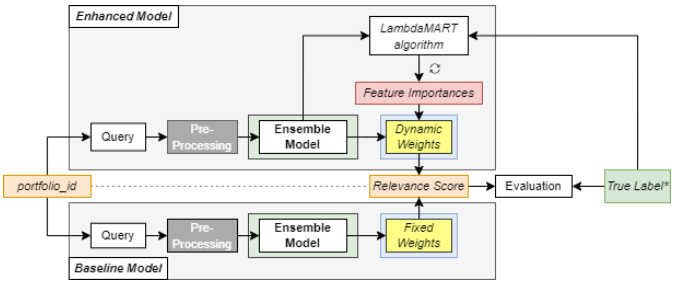

Figure 6: Overview of Methodology Architecture

The confusion matrix in this study is defined as follows:

1. True Positives (TP): Articles correctly identified as relevant.
2. True Negatives (TN): Articles correctly identified as not relevant.
3. False Positives (FP): Articles incorrectly identified as relevant (Type I error).
4. False Negatives (FN): Articles incorrectly identified as not relevant (Type II error).

From the confusion matrix, performance measurement is strengthened and extended by Precision, Recall, and F1 score, which are suitable for tasks with binary relevance. In this case, the true score of an article can

be 0 (non-relevant) or 1 (relevant). Note that, since both datasets are not significantly imbalanced, it is unnecessary to consider a precision-recall curve.

5.10 *Computational specifications*

All programming is done with Python 3.10 using Intel® Core TM i7 CPU @ 4.00GHz, 32 GB RAM for local test runs and for training an external Databricks Server with 2–8 workers and each driver with 14 GB Memory and 4 Cores, which results in 28-112 GB Memory and 8–12 Cores. To handle the big dataset efficient, PySpark[13] is used to load data. See the attached Github repository[14] repository for extended overview and used packages.

6 RESULTS

6.1 *EDA*

The EDA visualizes an overview of the three model scores and the total relevance score, as seen in Figure 7 below.

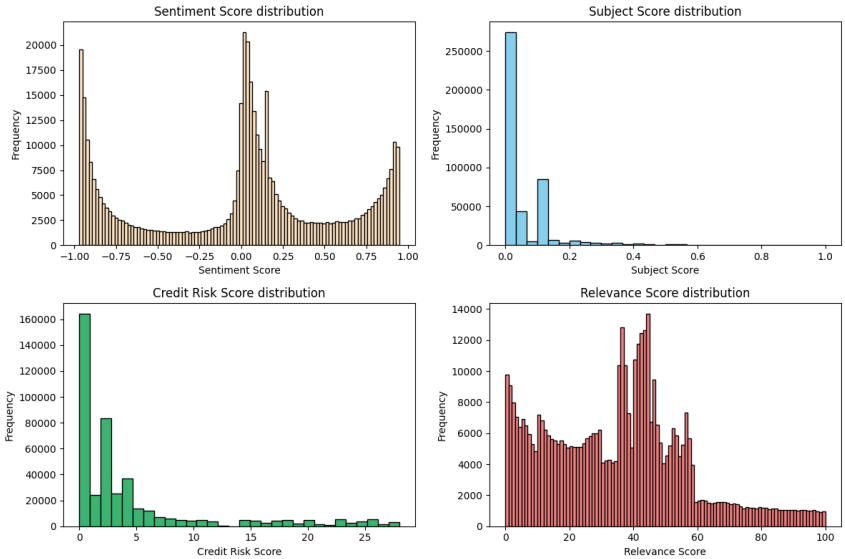

Figure 7: Overview of all feature (score) distributions (set 1)

[13] https://spark.apache.org/docs/latest/api/python/index.html
[14] https://github.com/TJJMReintjes/LTR_in_DMW_Thesis_TiU

The sentiment score exhibits a polarized distribution, with peaks around extremes at -1 and 1, and peaks centering around a neutral sentiment near 0. This polarization is beneficial for feature importance analysis as it reflects a clear distinction in the sentiment's positive and negative aspects, crucial for rating of articles.

In contrast, the subject score distribution shows a considerable skewness, with almost all scores clustering near zero. It seems that this could indicate potential issues in the score's calculation method, or imply that the articles' titles often do not precisely match the query content. As the subject score is based on the cosine similarity with the query only in the title, and does not take the content of the article in to account, may lead to a less accurate score distribution.

Furthermore, the credit risk score is also skewed, but this is a logical result as it is based on very specific risk-related keywords. Not every article will necessarily correspond to these keywords of credit risk, but it might still be a relevant article. Thus, this score is clearly an additional tool to calculate the relevance score.

Finally, the total relevance score does not exhibit a clear and unambiguous distribution. The distribution appears to be somewhat right-skewed but lacks a distinct pattern. It features numerous peaks around a relevance score of 40, creating a complex and non-uniform distribution. Notably, this relevance score of 40 serves as the threshold for classifying articles as relevant or non-relevant. It is important to highlight that the distribution undergoes significant changes beyond a relevance score of 60. Consequently, it can be concluded that there is thus a small amount of articles that genuinely fall into the relevant category.

6.2 *Baseline Model: Metrics and Confusion Matrices*

In general, the baseline model demonstrates reasonable accuracy and precision across both datasets. However, the variance in recall and F1 scores between expert and user annotations highlights the model's limitations in universally capturing the nuances of relevance in real-life scenarios.

The evaluation overview of the baseline model, presented in Table 2, illustrates the following metrics for both expert-labeled and user-labeled datasets.

Examining the confusion matrices depicted in Figure 8 and 9, we observe a significant difference in model performance between expert and user-labeled data. In the expert-labeled set, false positives (FP) and false negatives (FN) are relatively evenly distributed. However, in the user-

Table 2: Evaluation Overview Baseline model

Metric	Expert Labeled	User Labeled
Accuracy	0.76	0.69
Precision	0.66	0.70
Recall	0.69	0.44
F1 Score	0.67	0.54

labeled set, false negatives are clearly more dominant than false positives. This discrepancy suggests variations in the perception of relevance between experts and general users, indicating a potential deficiency in the computation of relevance scores for real-life scenarios.

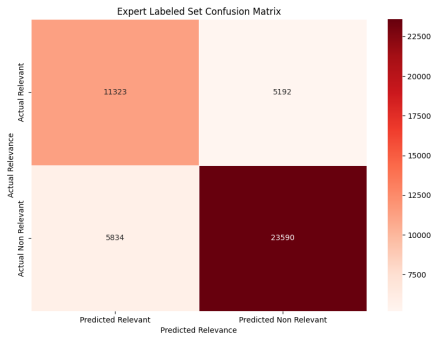

Figure 8: Expert Labeled Confusion Matrix Baseline

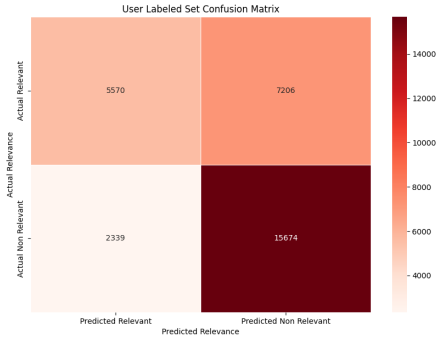

Figure 9: User Labeled Confusion Matrix Baseline

6.3 *Dynamic Model: LambdaMART's Feature Importances*

In this research, to address the first sub-research question (Section 3), the focus is on extracting feature importances rather than achieving the highest possible ranking accuracy. The interpretation of the model's output should align accordingly. The feature importance results will provide insights into which factors (sentiment, credit risk, subject scores) are most crucial in determining the relevance of an article, given the binary labels.

Figure 10 presents a normalized distribution of feature importances per `portfolio_id`. The heatmap reveals a consistent importance of the subject score across portfolios, while the sentiment and credit risk scores exhibit more variability. This observation suggests that while the subject score provides a stable contribution to relevance scoring, the sentiment, and credit risk scores play more dynamic roles, changing in importance for

specific contexts or different portfolios. Despite outliers, it is visible that the sentiment score has the overall highest feature importance, followed by credit risk, and then the subject score.

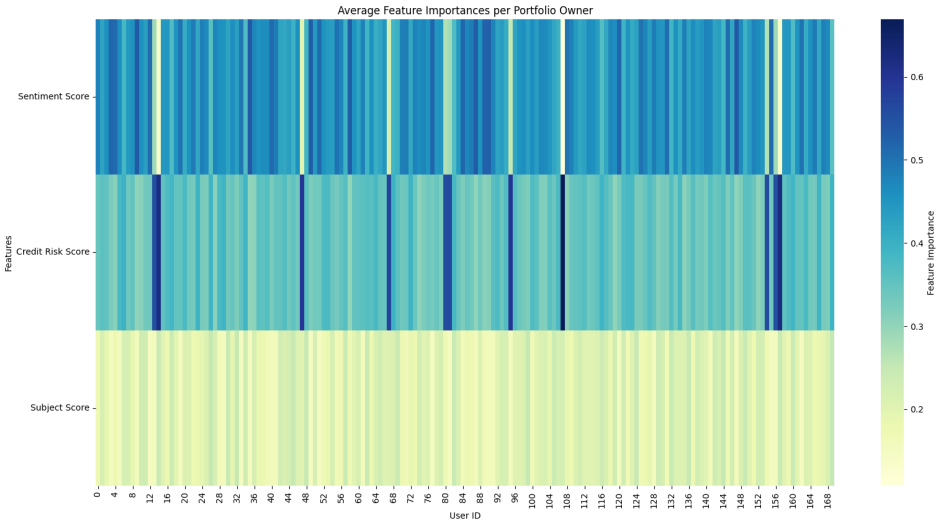

Figure 10: Average Feature Importances per Portfolio

As shown in Table 3 below, the comparison of the normalized averaged feature importances (proportional weights) appears to align reasonably well with the current fixed weights, albeit with some notable differences. This alignment validates the expert-selected fixed weights to some extent but also indicates potential differences in relevance scores per `portfolio_id`. The computed averaged feature importance of sentiment, credit risk, and subject score has a percentage difference of 13.22

Table 3: Average Feature Importances per Model

Feature	**Av. Importance**	**Current Weights**	**% Difference**
Sentiment Score	0.438	0.500	13.22%
Credit Risk Score	0.362	0.300	18.73%
Subject Score	0.201	0.200	0.50%

6.3.1 *Parameter Tuning*

The grid search returned 0.1 as the learning rate and 50 estimators as the best parameters (Appendix A). These values were used in the model to extract feature importances. Unfortunately, due to time limitations, no other hyperparameter combinations were tested, as discussed in Section 7.3.

6.4 *Dynamic Model: Metrics and Confusion Matrices*

To assess the feasibility of utilizing feature importances as dynamic weights, as inquired in the second sub-research question, we examine the results presented in Table 4 below. Minor improvements have been observed, although they are not notably significant. The enhancements are marginal, with an average increase of 0.01 in Accuracy, Precision, and Recall for both expert and user sets. Moreover, there is a slight uptick in the F1 score, amounting to 0.02 and 0.01 for the expert and user-labeled sets, respectively.

Table 4: Dynamic Model Results compared to Baseline Model

Metric	**Expert Labeled**	**Difference**	**User Labeled**	**Difference**
Accuracy	0.77	+0.01	0.69	+0.00
Precision	0.67	+0.01	0.71	+0.01
Recall	0.70	+0.01	0.45	+0.01
F1 score	0.69	+0.02	0.55	+0.01

The dynamic model, which incorporates averaged and normalized feature importances as weights, shows minor improvements compared to the baseline model, as seen in Figure 11 and 12. However, these improvements are not significant, suggesting that while the model adapts to portfolio-specific requirements, the overall impact on classification accuracy is limited. There are fewer false positives and false negatives, but there is not a significant shift in the confusion matrix, unfortunately. This result also holds for the user-test set, in which false negatives remain the dominant factor.

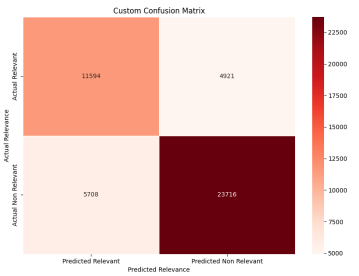

Figure 11: Expert Labeled Confusion Matrix Dynamic Model

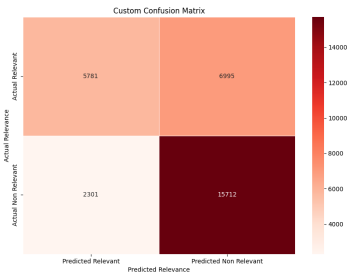

Figure 12: User Labeled Confusion Matrix Dynamic Model

In summary, despite not observing significant changes in the confusion matrices, this does not immediately refer to negligible improvements. Therefore, we need to delve deeper into the error analysis of specific articles with corresponding `portfolio_ids` to obtain more insights into the dynamic weights.

6.5 *Error Analysis of Feature Importances*

The top 5 outliers of Figure 10 are shown in Table 5. As concluded in the beginning, the outliers are related to specific `portfolio_ids`. This is based on an increase in the credit risk score's importance, resulting in a decrease in sentiment score's importance, while the subject score's importance stays constant around 0.200.

Table 5: Feature Importances Top Outliers

Portfolio_id	**Sentiment Score**	**Credit Risk Score**	**Subject Score**
13	0.288	0.554	0.158
14	0.136	0.619	0.245
47	0.199	0.596	0.205
67	0.219	0.573	0.208
80	0.270	0.559	0.171

Looking into the `department_id` of each `portfolio_id` of the top 5 outliers, we found out that they are related to three separate departments; but all related to investments and equity funds. In other words, these are departments that probably rely more on the importance of credit risk scores to relevance than other departments.

6.6 *Error Analysis of the Dynamic Model*

The error analysis begins with a comparative assessment of the relevance scoring against the baseline distribution. A key observation here is that, while the previous overall metrics did not show significant changes, the relevance scoring exhibited noticeable alterations. In Figure 13[15], a density plot is shown to provide a more nuanced view of the distribution changes, showing notable shifting of relevance scores. These shifts are attributed to the changes in the weights assigned to each `portfolio_id`, which directly influenced the overall relevance score calculations. In this context, the utility of confusion matrices is somewhat limited, as they primarily address

[15] Due to the function of a density distribution, it appears to be outside the range of 0–100, but in fact, this is not the case. See Appendix C for the histogram plot (Figure 14)

binary classification. The matrices do not offer insights into the distribution of relevance scores, from which articles are considered relevant if they exceed a threshold of 40.

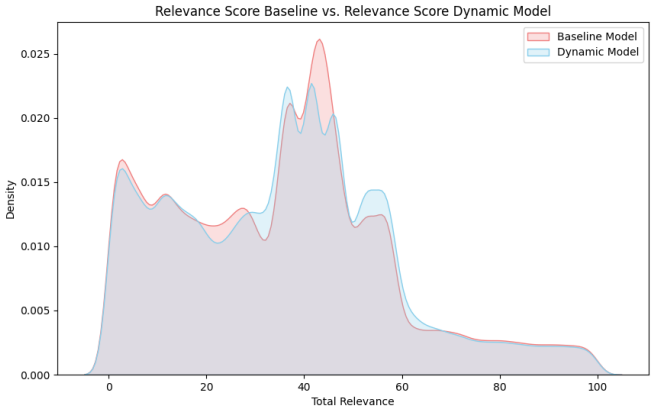

Figure 13: Density Model Comparison

An analysis of the relevance score distribution indicates that the dynamic model's adjustments have not significantly altered the statistical measures—mean, median, and standard deviation, as shown in Table 6. This observation is crucial because it suggests that, while the distribution of scores (depicted in Figure 13 and Appendix C Figure 14) has undergone shifts, these changes have counterbalanced each other. Increases in certain areas of the distribution were compensated by decreases in others, resulting in relatively unchanged central tendency measures (mean and median). Additionally, the standard deviation remains relatively stable despite apparent shifts in the data distribution.

Table 6: Relevance Scores Basic Statistics

	Baseline	**Dynamic**
Mean	35.33	35.97
Median	36.55	36.52
Std Deviation	22.41	22.27

In summary, we do not observe substantial shifts in mean, median, and Std Deviation of the relevance score distribution, while the distribution's shape has significantly changes. These critical observations highlight a possible limitation of the study, which is discussed in Section 7.1.

7 DISCUSSION

The integration of LambdaMART with Dynamic Model Weights (DMW) in custom Information Retrieval Systems (IRSs) provided a nuanced perspective within the existing literature. While DMW and LambdaMART have been noted for their effectiveness in relevance ranking, as discussed in Section 4, our findings suggest only marginal improvements in metric performance. Before exploring the findings, it is essential to explicitly revisit the main research question and sub-questions to guide the discussion.

Main Research Question: *How can the integration of Learning to Rank (LTR) in Dynamic Model Weighting enhance the search relevance of cross-industry news articles?*

The outcomes reveal marginal improvements in metric performance, which can be attributed to several factors influencing the performance. This includes, for example, the complexity of the specific custom IRS architecture, the quality of the dataset including features, and the general challenges of conducting a relevance-related study.

Further discussion of the main research question depends on two sub-questions.

Sub-Question 1: *What are the feature importances, identified by the LambdaMART algorithm, that can be used in Dynamic Model weighting for relevance ranking?*

The computed feature importances rely on features calculated by custom models. Therefore, fundamentally, the reliability of the feature importances as well as the total relevance score can be doubted. Both are computed based on three different (custom) model scores, without any validation conducted to ensure the reliability of these calculations. For this study, we assumed the scores were reliable. As we critically analyze the distributions of all model scores, not every score has a promising distribution. For instance, the subject score distribution was significant skewed, whereas one would expect it to be more similar to the label distribution. As a relevant article would be expected to contain the query in the title, resulting in higher cosine similarity. Contrarily, the current subject score showed a lot of articles between a score of 0 and 0.2, which is skewed and notably low as we are working on a 0 to 1 scale. In addition, as mentioned before, the credit risk score was also quite skewed, but this measure only comes in at credit risk-related articles. Therefore, its skewness is not surprising, as not all articles, by definition, need to include cosine similarity

to specific credit risk keywords.

Sub-Question 2: *Does Dynamic Model Weighting, using extracted features, enhance relevance ranking compared to a fixed-weighting approach?*

As mentioned, we observed some changes in the relevance score distribution. It is noteworthy that basic statistics of central tendency, such as the mean and median, exhibit minimal change. The standard deviation also remains relatively stable despite apparent shifts in the data distribution. Such a pattern is often observed in large datasets where minor changes do not significantly impact the overall distribution's central tendency. The relatively unchanged standard deviation is probably attributable to the fact that we adjust individual scores. This may suggest how hard it is to make significant improvements in large datasets with minor tweaks to the model.

Consequently, while the mean and median provide a deceptive sense of stability, both histogram and density plot reveal shifted data. If we delve into the relevance scores greater than 40, unfortunately, we cannot obtain any insights if this leads to better ranking of articles based on relevance. This is due to the study's reliance on binary classification, which does not allow for an examination of whether the shift led to a better ranking order within already relevant articles. Thus, it is important to note that accuracy, precision, recall, and F1 score could be misleading in binary classification, as they do not fully capture the nuances of shifted data. A multilevel labeling system might have provided a more accurate representation of improvements, especially considering that LambdaMART is better suited for multilevel data, as discussed in Section 5.

Referring back to the main research question, there are some remaining discussions. Firstly, the minor performance improvement may result from two distinct processes in custom IRSs: the retrieval process from the API and the calculation of the total relevance score. These are critical areas of concern. False positive results may arise due to noise in the dataset retrieved from the API, i.e., the API already returning too many irrelevant articles.

Another important aspect of the study was the comparison of the two different test sets; one annotated by experts and the other reflecting real-life situations by users. This was essential to validate if the model accurately represents real-world data usage. It is observed that the performance on both sets was almost equal despite the different label distribution. The primary difference observed between the expert and user sets was the balance between FN and FP. In contrast to the expert set, where FP and FN

were evenly distributed, the user set showed a marked predominance of FN. This refers to the current relevance scores excluding articles that are apparently relevant for users.

However, the reliance on expert-annotated data and user-annotated data may be discussed for training further models. For example, retrieving user annotations through a feedback loop may be crucial in refining the model. Nevertheless, it needs to be considered critically, as incorrect feedback provided by users results in a high risk of bias. The system will end up focusing on one-sided articles, potentially omitting crucial and important content. This is also due to relevance's dependency on subjectivity per user, discussed in Section 7.1 and Section 7.3.

7.1 *Limitations*

The primary limitation of this research lies in the inherent user-dependent and highly subjective nature of relevance in IR. The subjectivity, combined with the reliance on true labels in relevance ranking studies, raises questions about the validity of these labels. Is this truly the correct label? Who determines that a result can be considered relevant, as individuals may perceive something as more relevant than others? It can be doubted if there is a systemic difference between experts and users, or if it is just inter-person variability of interpretation.

Additionally, the variability in the dataset, especially in response to major news events, can significantly impact the dataset and potentially skew results. Consequently, relevance appears to be a fluid concept that varies across users and changes over time. This variability makes it challenging to measure improvements in relevance models solely through quantitative means, necessitating an interpretative approach to evaluate relevance and model performance.

Furthermore, this study relied on evaluating binary labels, preventing a deeper exploration into whether the dynamic model results in a more effective ranking order. It determines the accuracy of the relevance score in distinguishing between relevant and non-relevant articles according to a true label but does not assess the range of relevancy a relevant article has for a user. In other words, it lacks a multi-level relevance assessment. For example, while we can observe shifts in the distribution of relevance scores, we cannot evaluate whether one relevant article is correctly scored higher or lower than another.

7.2 *Possible Implications*

The research contributes to the fields of Natural Language Processing (NLP) and custom Information Retrieval Systems (IRSs) by demonstrating the potential of combining Dynamic Model Weights (DMW) with advanced algorithms like LambdaMART. This approach addresses the challenge of adapting custom IRSs to the needs and contexts of diverse user groups (portfolios), crucial in an era marked by information overload, as mentioned in Section 4. On a societal level, this research emphasizes the importance of accuracy in IR, which is essential for effective decision-making in various industries, particularly in fields like banking, where informed decision-making is crucial.

7.3 *Future Research*

Given the time constraints of the study, it becomes essential to consider alternative boosting algorithms, conduct more extensive parameter tuning, and validate the quality of the features used. These three aspects may improve the fundament of this research. Additionally, implementing only user-based annotation sets to train the model is another promising direction, although the high potential risk of bias and human error needs consideration.

The temporal aspect of relevance suggests the usefulness of incorporating training data from different time periods, with a particular focus on periods marked by significant news events. Researchers can significantly improve the real-world applicability of IRSs by understanding how these systems adapt to and reflect constantly changing information. Furthermore, integrating various methodologies, such as simpler models combined with DMW, could result in a better balance between complexity, interpretability, and efficiency. Enriching the model with a wider range of features, such as topics, could enhance accuracy and decrease the risk of bias resulting from human labeling errors and discussions over what qualifies as relevant.

Another critical area of development is the use of specific exclusion mechanisms for search results with unique characteristics. This tailored approach would improve the accuracy and customization of IR, enabling IRSs to identify and exclude content that may be broadly relevant but does not meet the specific needs or preferences of individual users or contexts.

8 CONCLUSION

The integration of LambdaMART with DMW in custom IRSs provides a nuanced insight into the complex interplay of challenges and opportunities

within the domain of IR. This exploration, particularly focused on the relevance ranking of cross-industry news articles, has illuminated several critical facets and inherent challenges in this field.

While the study did not reveal statistically significant improvements over the baseline model, it showcased the potential of incorporating Learning to Rank techniques like LambdaMART into DMW systems. This integration, especially in adapting to the specific requirements of different portfolios, illustrates the dynamic model's capacity to meet user group-specific demands. However, the modest improvements in relevance scoring underscore the complexity of deploying advanced algorithms in practical scenarios, coupled with the intricate definition of relevance.

Central to this study is the relativity of relevance in IR, a theme strongly related to the research questions outlined in the paper. The observed variability and time-dependency of news data significantly affect the model's ability to predict relevance consistently.

A critical limitation of this research, aligned with the sub-research questions, is its reliance on binary classification for relevance ranking. Future research, therefore, should pivot towards exploring multilevel labeling systems, offering a more granular understanding of relevance that a binary system might overlook. Additionally, the study raises questions about the potential biases inherent in expert-annotated data versus real-life user feedback, questioning whether IRSs would then align more closely with actual user needs.

In conclusion, this research contributes to the evolving literature on IR, highlighting the need for continuous innovation and a nuanced understanding of relevance in the field of continuously changing information. It emphasizes the importance of adaptability and precision in IRS, especially custom IRSs, to meet the diverse and dynamic information needs of users in different domains.

REFERENCES

Burges, C. J. (2010). *From RankNet to LambdaRank to LambdaMART: An overview*.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *Conference: Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*. https://doi.org/10.1145/1102351.1102363

Büttcher, S., Clarke, C. L. A., & Cormack, G. V. (2010). *Information retrieval: Implementing and evaluating search engines*. http://ci.nii.ac.jp/ncid/BB03736550

Casalegno, F. (2022). Learning to Rank: A complete guide to ranking using machine learning. https://towardsdatascience.com/learning-to-rank-a-complete-guide-to-ranking-using-machine-learning-4c9688d370d4

Ceri, S., Bozzon, A., Brambilla, M., Della Valle, E., Fraternali, P., & Quarteroni, S. (2013). *An introduction to information retrieval*. https://doi.org/10.1007/978-3-642-39314-3\{_}1

Chen, T., & Guestrin, C. (2016). XGBoost. *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939785

Dogra, V., Verma, S., Kavita, K., Chatterjee, P., Shafi, J., Choi, J., & Ijaz, M. F. (2022). A complete process of text classification system using State-of-the-Art NLP models. *Computational Intelligence and Neuroscience*, *2022*, 1–26. https://doi.org/10.1155/2022/1883698

Dubey, A. (2022). A Practical Guide to LambdaMART in LightGBM - DataDrivenInvestor. https://medium.datadriveninvestor.com/a-practical-guide-to-lambdamart-in-lightgbm-f16a57864f6

Feng, X., Zhong, B., Dong, Y., & Qiu, J. (2018). Dynamic Weighted ensemble classification for credit scoring using Markov Chain. *Applied Intelligence*, *49*(2), 555–568. https://doi.org/10.1007/s10489-018-1253-8

Gul, R., & Al-Faryan, M. A. S. (2023). From Insights to impact: Leveraging data analytics for data-driven decision-making and productivity in banking sector. *Humanities and Social Sciences Communications*, *10*(1). https://doi.org/10.1057/s41599-023-02122-x

Gupta, A. K., & Bhatia, R. (2021). Ensemble approach for web page classification. *Multimedia Tools and Applications*, *80*(16), 25219–25240. https://doi.org/10.1007/s11042-021-10891-3

Honnibal, M., & Montani, I. (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing* [To appear].

Hu, Z., Wang, Y., Qu, P., & Li, H. (2019). Unbiased LambdaMART: An Unbiased Pairwise Learning-to-Rank Algorithm. *Association for Computing Machinery*. https://doi.org/10.1145/3308558.3313447

Huang, A. H., Wang, H., & Yang, Y. (2022). Finbert: A large language model for extracting information from financial text. *Contemporary Accounting Research*.

Huang, J.-T., Sharma, A., Sun, S., Li, X., Zhang, D., Pronin, P., Padmanabhan, J., Ottaviano, G., & Yang, L. (2020). Embedding-based Retrieval in Facebook Search. *Association for Computing Machinery*. https://doi.org/10.1145/3394486.3403305

Jain, V. (2023). An ensemble approach to data mining for real-time information retrieval. https://www.wwt.com/wwt-research/ensemble-approach-to-data-mining-for-real-time-information-retrieval

Jansen, B. J., Booth, D., & Spink, A. (2008). Determining the informational, navigational, and transactional intent of web queries. *Information Processing and Management*, *44*(3), 1251–1266. https://doi.org/10.1016/j.ipm.2007.07.015

Joachims, T., Freitag, D., & Mitchell, T. M. (1996). WebWatcher: a tour guide for the world wide web. *ResearchGate*. https://www.researchgate.net/publication/2820578_WebWatcher_A_tour_guide_for_the_World_Wide_Web

Kanhabua, N., & Anand, A. (2016). Temporal Information Retrieval. *SIGIR '16: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. https://doi.org/10.1145/2911451.2914805

Ke, G. (2017). LightGBM: a highly efficient gradient boosting decision tree. https://papers.nips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

Learning to rank — XGBoost 2.1.0-DeV documentation. (n.d.). https://xgboost.readthedocs.io/en/latest/tutorials/learning_to_rank.html

Li, H. (2015). *Learning to rank for information retrieval and natural language processing*. https://doi.org/10.1007/978-3-031-02155-8

LightGBM.LGBMRanker — LightGBM 4.1.0.99 Documentation. (n.d.). https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRanker.html

LiKewen, LiuWenying, ZhaoKang, ShaoMingwen, & LiuLu. (2015). A novel dynamic weight neural network ensemble model. *International Journal of Distributed Sensor Networks*, *11*(8), 862056. https://doi.org/10.1155/2015/862056

Mizzaro, S. (1998). How many relevances in information retrieval? *Interacting with Computers*, *10*(3), 303–320. https://doi.org/10.1016/s0953-5438(98)00012-5

Nguyen, P., Wang, J., & Kalousis, A. (2016). Factorizing LambdaMART for cold start recommendations. *Machine Learning*, *104*(2-3), 223–242. https://doi.org/10.1007/s10994-016-5579-3

Saračević, T. (2007). Relevance: A review of the literature and a framework for thinking on the notion in information science. Part II: Nature and Manifestations of Relevance. *Journal of the Association for Information Science and Technology*, *58*(13), 1915–1933. https://doi.org/10.1002/asi.20682

Shi, B., Özsoy, M. G., Dong, R., Smyth, B., Hurley, N., & Lawlor, A. (2020). Fourteenth ACM Conference on Recommender Systems. *Association for Computing Machinery*. https://doi.org/10.1145/3383313

Tekli, J. (2022). An overview of cluster-based image search result organization: background, techniques, and ongoing challenges. *Knowledge and Information Systems*, *64*(3), 589–642. https://doi.org/10.1007/s10115-021-01650-9

Trotman, A. (2005). Learning to rank. *Information Retrieval Journal*, *8*(3), 359–381. https://doi.org/10.1007/s10791-005-6991-7

Troussas, C., Krouska, A., Sgouropoulou, C., & Voyiatzis, I. (2020). Ensemble learning using fuzzy weights to improve learning style identification for adapted instructional routines. *Entropy*, *22*(7), 735. https://doi.org/10.3390/e22070735

Vicente-López, E., De Campos, L. M., Fernández-Luna, J. M., Huete, J. F., Tagua-Jiménez, A., & Tur-Vigil, C. (2014). An automatic methodology to evaluate personalized information retrieval systems. *User Modeling and User-Adapted Interaction*, *25*(1), 1–37. https://doi.org/10.1007/s11257-014-9148-9

Wongthongtham, P., Chan, K. Y., Potdar, V., Abu-Salih, B., Gaikwad, S. S., & Jain, P. (2018). State-of-the-Art Ontology annotation for personalised teaching and learning and Prospects for smart Learning Recommender based on multiple intelligence and Fuzzy Ontology. *International Journal of Fuzzy Systems*, *20*(4), 1357–1372. https://doi.org/10.1007/s40815-018-0467-6

Wu, Q., Burges, C. J. C., Svore, K. M., & Gao, J. (2010). Adapting boosting for information retrieval measures. *Information Retrieval Journal*, *13*(3), 254–270. https://doi.org/10.1007/s10791-009-9112-1

Zhu, Y. (2023). Large Language models for information retrieval: a survey. https://arxiv.org/abs/2308.07107

9 APPENDIX A

Table 7: Parameters for the LambdaMART algorithm in the LightGBM framework.

Parameter	Setting
objective	'lambdarank'
boosting_type	'gbdt'
metric	'ndcg'
num_leaves	31
max_depth	-1
learning_rate	0.1
n_estimators	50
Group/Data Structure	Articles per query in each portfolio

9.1 *Descriptions of variables*

- **objective:** The objective is set to 'lambdarank'. This objective is specifically designed for ranking tasks in LightGBM and is needed to implement LambdaMART.
- **boosting_type:** Refers to traditional Gradient Boosting Decision Tree.
- **metric:** Using 'ndcg' (Normalized Discounted Cumulative Gain) as the metric. However, since the focus is on feature importance rather than the actual ranking, NDCG serves more as a performance indicator. It is crucial to remember that the metric's role here is to guide the training process, rather than to evaluate the final model's effectiveness in our use case.
- **num_leaves:** Setting this to 31 is the default. It is important to be cautious with this parameter, as too many leaves can lead to overfitting, especially in datasets with complex structures. This can be included later on in more extensive parameter tuning
- **max_depth:** Maximum tree depth for base learners, set to default.
- **learning_rate:** A learning rate of 0.1 is coming from the hyperparameter tuning, but this is also the default. A more extensive parameter tuning would validate this decision.
- **n_estimators:** Number of boosted trees to fit; is set to 50 based on hyperparameter tuning.

- **Group/Data Structure:** This is a crucial aspect of the setup. It reflects the structure of your data, i.e., the articles per query per portfolio.

10 APPENDIX B

10.0.1 *Overview Step Plan*

STEP 1: DATA PREPARATION

1. Structure the dataset so that each row represents an individual article, including its features (sentiment score, credit risk score, subject score), binary relevance label, and identifiers for the associated search, query, portfolio and department id.
2. Group articles based on their corresponding portfolio and query, ensuring structure for the input of LambdaMART algorithm.

STEP 2: MODEL CONFIGURATION

1. Configure LambdaMART parameters in the LightGBM framework, see Appendix 9.
2. Ensure to set the objective to 'lambdarank' and the metric to 'ndcg', focusing on feature importance rather than ranking accuracy.

STEP 3: MODEL TRAINING AND PARAMETER TUNING

1. Train the LambdaMART model using the prepared dataset.
2. Use the Validation set for parameter tuning

STEP 4: FEATURE IMPORTANCE EXTRACTION

1. Post-training; extract feature importances from the model.
2. Analyze the importances to determine the influence of each feature on the binary relevance of articles.
3. Convert feature importances to averaged and normalized proportional weights to use in an ensemble architecture.

11 APPENDIX C

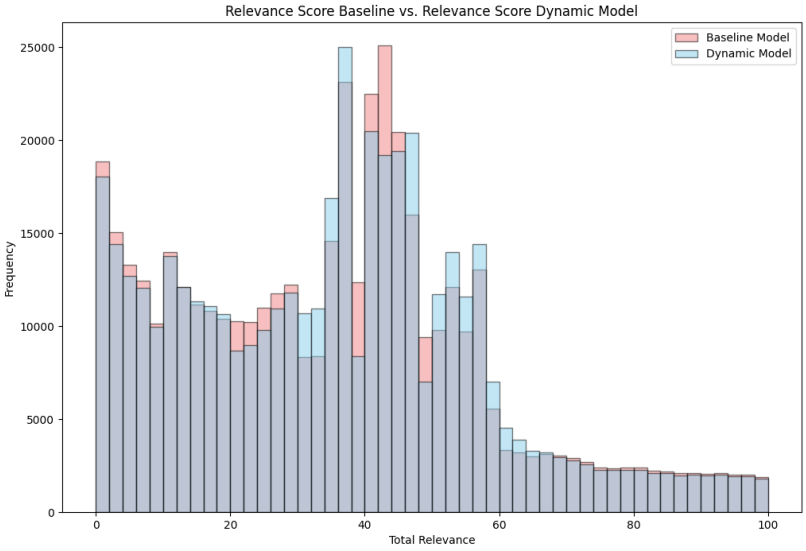

Figure 14: Histogram Model Comparison