Titre: Title:	Improving Information Retrieval and Recommender Systems with Contextual Data and Re-Ranking
Auteur: Author:	Baharan Nouriinanloo
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Nouriinanloo, B. (2024). Improving Information Retrieval and Recommender Systems with Contextual Data and Re-Ranking [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/61634/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/61634/
Directeurs de recherche: Advisors:	Maxime Lamothe
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Improving Information Retrieval and Recommender Systems with Contextual Data and Re-Ranking

BAHARAN NOURIINANLOO
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Décembre 2024

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Improving Information Retrieval and Recommender Systems with Contextual Data and Re-Ranking

présenté par **Baharan NOURIINANLOO**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Michel DESMARAIS, président
Maxime LAMOTHE, membre et directeur de recherche
Amine MHEDHBI, membre

DEDICATION

To my parents,
Elahe and Saeid,
for your constant love and support

To my grandmother and aunt,
Taji and Elham,
for your encouragement and warmth.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisor, Professor Maxime Lamothe, for his outstanding guidance, constant support, and invaluable supervision throughout the duration of this research. His mentorship has been pivotal at every stage, equipping me with the expertise and confidence necessary to tackle the challenges of this thesis. I am deeply grateful for his dedication and insightful contributions, which have greatly influenced the outcome of this work.

I would like to express my deepest gratitude to my parents for their unwavering support and encouragement throughout my educational journey. Their confidence in my abilities and constant motivation have been a profound source of strength.

I am also sincerely thankful to Professor Amine Mhedhbi and Professor Michel Desmarais for dedicating their time and effort to reviewing my master's thesis.

Thank you all for your invaluable support and contributions to this important milestone in my academic career.

RÉSUMÉ

L'accès à l'information est un besoin humain fondamental. Dans le monde actuel axé sur les données, la croissance exponentielle de l'information a créé une demande critique pour des systèmes capables de récupérer efficacement des données pertinentes. Les systèmes de recherche d'information (Information Retrieval, IR) et les systèmes de recommandation ont émergé comme des solutions clés, permettant aux utilisateurs d'accéder à du contenu pertinent et de recevoir des recommandations personnalisées. Cette thèse vise à améliorer ces systèmes en abordant deux défis majeurs : améliorer l'explicabilité des systèmes de recommandation et optimiser le re-rank dans les systèmes IR à l'aide de modèles de langage avancés (Large Language Models, LLMs).

La première partie de cette thèse met l'accent sur l'importance de l'explicabilité dans les systèmes de recommandation pour renforcer la confiance des utilisateurs, leur satisfaction et leur prise de décision. Elle présente un système de recommandation explicable innovant développé pour un partenaire aérien, qui génère non seulement des recommandations personnalisées de destinations, mais fournit également des explications claires basées sur des caractéristiques pour chaque suggestion. En mettant en avant la transparence et l'explicabilité, ce système permet aux utilisateurs de comprendre les raisons derrière ses recommandations, comblant ainsi le fossé entre des algorithmes complexes et la compréhension des utilisateurs.

La deuxième partie de cette thèse explore les problèmes de classement dans les systèmes IR, où l'efficacité des résultats classés affecte directement la satisfaction des utilisateurs et les performances du système. Les LLMs ont suscité un grand intérêt en raison de leurs puissantes capacités de compréhension et de génération de texte. Des études récentes ont exploité les LLMs pour le re-classement de passages en mode zéro-shot dans les systèmes IR, utilisant diverses approches, notamment les méthodes pointwise et listwise. Dans l'approche pointwise, le LLM génère un score de pertinence pour chaque passage par rapport à la requête, indépendamment des autres passages. En revanche, dans l'approche listwise, le LLM génère une liste classée des étiquettes de documents en fonction de leur pertinence par rapport à la requête.

Cette thèse propose une étape préliminaire de filtrage basée sur un LLM en mode pointwise pour identifier les passages pertinents et non pertinents avant le processus de re-classement, dans le but d'améliorer les performances des re-rankers listwise dans les systèmes IR. Cette approche repose sur une technique avancée de conception de prompts appelée Plan-and-Solve. De plus, cette thèse conduit une étude empirique pour examiner le rôle de la cohérence interne

(self-consistency) et de l'élicitation de confiance dans l'amélioration des performances des re-rankers pointwise basés sur les LLMs, en affinant la pertinence et la précision des résultats classés.

Nous constatons qu'en utilisant un petit nombre de scores de pertinence générés par des humains, associés aux scores de pertinence générés par les LLMs, il est possible d'éliminer efficacement les passages non pertinents avant le re-classement. Nos expériences montrent également que ce filtrage préliminaire basé sur un LLM en mode pointwise permet au LLM d'améliorer ses performances dans les tâches de re-classement en mode zéro-shot. En outre, nos résultats révèlent que l'élicitation de confiance peut améliorer les performances des re-rankers pointwise basés sur les LLMs en mode zéro-shot. Notamment, la fréquence brute des réponses d'un LLM dans le re-classement pointwise est moins utile que les scores de confiance obtenus à partir de ces réponses.

ABSTRACT

Access to information is a fundamental human need. In today's data-driven world, the exponential growth of information has created a critical demand for systems that can efficiently retrieve relevant data. Information Retrieval (IR) systems and Recommender Systems have emerged as key solutions, enabling users to access relevant content and receive personalized recommendations. This thesis aims to improve these systems by addressing two major challenges: enhancing explainability in recommender systems and improving re-ranking in IR systems using Large Language Models (LLMs).

The first part of this thesis focuses on the importance of explainability in recommender systems to improve user trust, satisfaction, and decision-making. It introduces a novel explainable recommender system developed for an airline partner, which not only generates personalized destination recommendations but also provides clear, feature-based explanations for each suggestion. By prioritizing transparency and explainability, the system allows users to understand the rationale behind its recommendations, thereby bridging the gap between complex algorithms and user comprehension.

The second part of this thesis explores ranking problems in IR systems, where the effectiveness of the ranked results directly affects user satisfaction and system performance. LLMs have gained significant attention for their strong text understanding and generation capabilities. Recent studies have leveraged LLMs for zero-shot passage re-ranking in IR systems, employing various methods, including pointwise and listwise approaches. In the pointwise approach, the LLM generates a relevance score for each passage concerning the query, regardless of the other passages. In contrast, in the listwise approach, the LLM generates a ranked list of document labels based on their relevance to the query. This thesis proposes an LLM-based pointwise pre-filtering step to identify relevant and non-relevant passages before the re-ranking process, aiming to enhance the performance of listwise re-rankers in IR systems. This approach is based on the advanced prompt engineering technique known as Plan-and-Solve. Additionally, this thesis conducts an empirical study to investigate the role of self-consistency and confidence elicitation in improving the performance of LLM-based pointwise re-rankers by refining the relevance and accuracy of ranked results.

We find that by using a small number of human-generated relevance scores, coupled with LLM relevance scoring, it is effectively possible to filter out irrelevant passages before re-ranking. Our experiments also show that this LLM-based pointwise pre-filtering then allows the LLM to perform better at zero-shot re-ranking tasks. In addition, our results reveal that

confidence elicitation can enhance the performance of LLM-based pointwise zero-shot passage re-rankers. Notably, the raw frequency of answers from an LLM in pointwise re-ranking is less useful than the confidence scores elicited from them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

CF	Collaborative Filtering
COT	Chain of Thought
FN	False Negative
FP	False Positive
IDF	Inverse Document Frequency
IR	Information Retrieval
LLM	Large Language Model
LTR	Learning to Rank
MF	Matrix Factorization
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing
PS	Plan and Solve
RG	Relevance Generation
RG-S(0, k)	Rating Scale 0-to-k Relevance Generation
TF-IDF	Term Frequency Inverse Document Frequency
TN	True Negative
TP	True Positive
UPR	Unsupervised Passage Re-ranking

CHAPTER 1 INTRODUCTION

Access to information is a fundamental human need in daily life. In a fast-paced, data-driven world, the volume of information available to individuals has grown exponentially. This rise in information has created a demand for efficient systems that enable users to retrieve relevant data quickly and accurately. To address this need, numerous Information Retrieval (IR) systems have been developed, each designed to bridge the gap between the query of a user and the information repository [1]. The core functionality of an IR system lies in its ability to determine the relevance of a piece of content—be it text, images, audio, or video—to a user-issued query. By doing so, these systems ensure that users can efficiently access the most pertinent data to fulfill their informational needs.

Among the wide array of applications enabled by IR systems, recommender systems [2] stand out as critical tools for personalizing the user experience. By analyzing user behavior, preferences, and historical interactions, recommender systems generate suggestions tailored to individual users. These systems have become widespread, serving as the backbone of platforms such as e-commerce websites [3], tourism [4], and job matching applications [5]. Their primary goal is to assist users in navigating vast collections of items, whether these are products, media content, or potential travel destinations, by providing a curated list of recommendations.

Over the years, many research has focused on advancing recommender systems, resulting in the development of various specialized models tailored to specific domains and objectives. A significant emerging trend within this field is the focus on Explainability. As users become increasingly conscious of the decisions made by automated systems, there is a growing demand for transparency and accountability. Explainable Recommender Systems [2] address this need by providing users with clear and interpretable explanations of why certain recommendations were made. These explanations enhance user trust, facilitate informed decision-making, and improve the overall user experience.

To achieve explainability, different methodologies have been incorporated into the design of recommender systems [2]. These approaches ensure that the internal processes and outputs of the system are interpretable, offering users insights into how their preferences and behaviors influenced the recommendations. In this thesis, **Chapter 4** is dedicated to designing and implementing an explainable recommender system for an airline partner. The proposed system not only generates personalized destination recommendations but also provides feature-based explanations to clarify its decisions. Additionally, the system highlights the importance of

a re-ranking module, leading us to consider its critical role in improving the relevance and ordering of recommendations to the users. A well-designed recommender system should provide a ranked list of items that best align with the preferences, history, and explicit requests of the users. Despite its importance, ranking remains a challenging problem due to the need to balance various factors such as accuracy, diversity, and computational efficiency.

The problem of ranking is fundamental to IR systems [6]. In ranking tasks, a ranking model (or function) is employed to generate an ordered list from a given set of objects, where the relative order of objects reflects their degrees of relevance, preference, or importance, depending on the specific application. With the recent advancements in LLMs [7], new methods have emerged to address ranking problems in IR systems. LLMs, based on the Transformer architecture [8], demonstrate significant capabilities in understanding and generating natural language, making them highly effective for tasks that require deep semantic understanding, such as re-ranking in IR systems.

Consequently, researchers have been exploring the integration of LLMs as zero-shot re-rankers [9–12], leveraging their ability to refine initial retrieval results by considering nuanced semantic relationships and contextual details. The use of LLMs in zero-shot ranking tasks can generally be categorized into four main approaches: Pointwise [9, 13], Listwise [10, 14], Pairwise [12], and Setwise [11].

Pointwise approaches focus on evaluating individual passage in relation to a query. Two popular pointwise methods include generation-based and likelihood-based approaches. In the generation-based approach, an LLM determines relevance through a "yes/no" task, ranking documents based on the likelihood of generating a "yes" response [15, 16]. Likelihood-based approaches, such as Query Likelihood Modeling (QLM) [13], involve an LLM generating a relevant query for a document and ranking documents based on the likelihood of reproducing that query.

In contrast, listwise approaches [10] consider a list of candidate documents in relation to a given query. In this approach, the LLM is asked to generate a ranked list of documents based on their relevance to the query.

In **Chapter 5**, a novel pointwise pre-filtering step is proposed, which uses a LLM to identify relevant and non-relevant passages before they are passed to a listwise re-ranker. The pre-filtering step aims to reduce the number of irrelevant passages sent to the LLM, thereby enhancing the overall performance of the re-ranker. This method is based on the Plan-and-Solve [17] prompt engineering technique, which guides the LLM by introducing a planning stage before the solving process.

Furthermore, **Chapter 6** presents an empirical study to investigate the role of self-consistency and confidence elicitation in LLM-based pointwise re-rankers. This chapter investigates the effect of self-consistency [18], which involves sampling multiple and diverse reasoning paths and selecting the most consistent answer, on LLM-based pointwise re-rankers. Additionally, the impact of confidence elicitation [19], the process of estimating the confidence of LLM in its responses without model fine-tuning or accessing internal information, is also evaluated.

1.1 Thesis Organization

The structure of this thesis is as follows:

- **Chapter 2**
  This chapter provides foundational concepts and definitions related to Information Retrieval and recommender systems.

- **Chapter 3**
  This chapter surveys recent advancements in IR and recommender systems, focusing on explainability and the role of LLMs in ranking tasks. State-of-the-art techniques and their limitations are discussed to identify gaps in the existing research.

- **Chapter 4**
  In this chapter, the design and implementation of an explainable recommender system for an airline partner are presented. The system provides feature-based explanations for its recommendations which make them more transparent and clear.

- **Chapter 5**
  This chapter introduces a new pre-filtering step, before re-rankers, to enhance the performance of IR systems. The proposed method employs planning and devising prompting techniques to refine the ranking process.

- **Chapter 6**
  This chapter explores the role of confidence elicitation and self-consistency in pointwise approaches for LLM-based re-ranking. Through empirical studies, the impact of these techniques on ranking effectiveness is evaluated, and strategies for further improvement are proposed.

In summary, this thesis contributes to advancing IR and recommender systems by integrating explainability into recommender systems and leveraging the capabilities of LLMs for improv-

ing re-ranking tasks. These contributions aim to enhance user satisfaction, transparency, and trust in modern information systems.

CHAPTER 2 BACKGROUND

In this chapter, we introduce the foundational concepts and techniques commonly employed in Information Retrieval (IR) systems. First, we provide an overview of the core principles that define IR, including its typical architecture and how components interact within the system. Then, we explain fundamental metrics and datasets to evaluate the efficiency of the IR systems. Finally, we describe two widely used ranking algorithms, TF-IDF (Term Frequency-Inverse Document Frequency) and BM25, which play a crucial role in evaluating and ranking document relevance. These algorithms serve as primary tools in IR, forming the basis for determining how well a document matches a user query.

2.1 Core Definitions in IR Systems

All of the definitions in this section are taken from the study [20].

Information retrieval involves finding content—typically unstructured text documents—that meets a specific information need within large collections of data, generally stored on computer systems.

An information need refers to the subject that the user wants to learn more about.

In IR systems, documents refers to the different units that have been selected for the retrieval system. These could be single memos or sections of a book. The group of documents used for retrieval will be called the (document) collection. It is also commonly known as a corpus, which is a collection of texts.

A query represents the attempt of the user to communicate their information needs to the computer.

A document is considered relevant if the user believes it holds valuable information related to their specific information need.

The vector space model represents a set of documents as vectors within a common vector space. This model is essential for various information retrieval tasks, including scoring documents in response to a query, classifying documents, and clustering documents.

A standard method for measuring the similarity between two documents d_1 and d_2 is to compute the cosine similarity of their vector representations $\vec{V}(d_1)$ and $\vec{V}(d_2)$. This is defined as:

$$\text{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|},$$

where the numerator $\vec{V}(d_1) \cdot \vec{V}(d_2)$ denotes the dot product (or inner product) of the vectors $\vec{V}(d_1)$ and $\vec{V}(d_2)$, and the denominator represents the product of their Euclidean lengths. The dot product $\vec{x} \cdot \vec{y}$ of two vectors is calculated as $\sum_{i=1}^{M} x_i y_i$, where M is the number of components in the vectors. For a document d with vector representation $\vec{V}(d)$ consisting of components $\vec{V}_1(d), \ldots, \vec{V}_M(d)$, the Euclidean length of d is given by:

$$|\vec{V}(d)| = \sqrt{\sum_{i=1}^{M} \vec{V}_i(d)^2}.$$

The denominator in the cosine similarity formula normalizes the vectors $\vec{V}(d_1)$ and $\vec{V}(d_2)$ to unit vectors, making the similarity score independent of vector length.

2.2 Metrics and Datasets for Evaluating IR Systems

Two fundamental metrics for evaluating the effectiveness of an IR system are precision and recall. These metrics are first defined for the case where an IR system returns a set of documents in response to a query.

Precision (P) is the fraction of retrieved documents that are relevant to the query, calculated as:

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant}|\text{retrieved}).$$

Recall (R) is the fraction of all relevant documents that have been retrieved, defined as:

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved}|\text{relevant}).$$

For a clearer understanding, we can organize retrieval outcomes in a contingency table:

	Relevant	Nonrelevant
Retrieved	True Positives (tp)	False Positives (fp)
Not Retrieved	False Negatives (fn)	True Negatives (tn)

With this, precision and recall can also be expressed as:

$$P = \frac{tp}{tp + fp},$$
$$R = \frac{tp}{tp + fn}.$$

A single measure that balances precision and recall is the F-measure, defined as the weighted harmonic mean of precision P and recall R. The formula for the F-measure is given by:

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \quad \text{where} \quad \beta^2 = \frac{1-\alpha}{\alpha}, \quad \alpha \in [0,1] \quad \text{and} \quad \beta^2 \in [0,\infty]$$

The measures such as precision, recall, and the F-measure assess retrieval effectiveness based on unordered sets of documents. However, to evaluate ranked retrieval results—common in modern search engines—we need different metrics that consider document order. One such metric is Normalized Discounted Cumulative Gain (NDCG), which is particularly useful in ranking scenarios with non-binary relevance levels.

NDCG is evaluated over the top k search results. For a set of queries Q, let $R(j,d)$ denote the relevance score assigned to document d for query j. Then, NDCG is then calculated as follows:

$$\text{NDCG}(Q,k) = \frac{1}{|Q|}\sum_{j=1}^{|Q|} Z_{kj} \sum_{m=1}^{k} \frac{2^{R(j,m)}-1}{\log_2(1+m)},$$

where Z_{kj} is a normalization factor ensuring that the ideal ranking for query j yields an NDCG of 1. If fewer than k documents are retrieved for a query, the summation is adjusted to cover only the available documents.

Several benchmark datasets are extensively used to evaluate the performance of IR systems. Key datasets include the TREC 2019/2020 Deep Learning Tracks and the BEIR benchmark, each designed to address distinct IR challenges and encourage innovative solutions.

TREC 2019/2020 Deep Learning Tracks The Deep Learning tracks organized by TREC in 2019 and 2020 have become foundational in the evaluation of large-scale IR systems, particularly for document and passage ranking tasks. These tracks received significant attention from the IR community, attracting widespread participation. Participants were tasked with ranking documents and passages independently, with separate judgment pools and performance metrics for each task. Notably, the organizers provided an initial ranked

list for reranking, allowing participants to focus exclusively on developing re-ranking models if desired.

The datasets for TREC DL contain small query sets with four-point scaled relevance judgments, enabling fine-grained evaluation of ranking systems. In TREC DL 2019 [21], the dataset included 43 queries with an average of 215.4 judgments per query for passage ranking and 378.1 for document ranking. The 2020 track [22] expanded to 54 queries for passage ranking with 210.9 average judgments per query and 45 queries for document ranking with an average of 202.2 judgments per query.

BEIR Benchmark The BEIR dataset [23] is a comprehensive and diverse benchmark designed to assess IR systems across a wide range of retrieval tasks and domains. Some commonly used retrieval tasks in BEIR include:

- **Covid:** Focuses on retrieving scientific articles relevant to COVID-19-related queries.
- **NFCorpus:** A biomedical dataset.
- **Touche:** An argument retrieval dataset.
- **DBpedia:** Tasked with retrieving relevant entities from the extensive DBpedia corpus.
- **SciFact:** Designed for claim verification.
- **Signal:** Tasked with retrieving relevant tweets for a given news title.
- **News:** A task focused on retrieving news articles that correspond to particular news headlines.
- **Robust04:** Evaluates poorly performing or challenging topics.

2.3 TF-IDF Technique

In the field of IR, Term Frequency-Inverse Document Frequency (TF-IDF) is a foundational technique widely recognized for its effectiveness in determining the significance of words within a document collection. TF-IDF plays a crucial role in identifying terms that hold particular relevance in individual documents by measuring how often a word appears in a document relative to its frequency across the entire corpus. Discussing TF-IDF in this thesis is essential because it provides a baseline approach to text vectorization, allowing us to convert text data into numerical vectors that represent the importance of each word. This

method forms a foundational comparison point for exploring more advanced natural language processing techniques, highlighting both the strengths and limitations of traditional feature extraction in comparison to newer methods.

Term Frequency (tf) refers to a weighting method where each term in a document is assigned a weight based on the number of times it appears in that document. This method seeks to calculate a score reflecting the relationship between a query term t and a document d through the weight of t in d. The most straightforward approach is to set the weight equal to the frequency of term t in document d. This weighting method is known as term frequency, denoted as $tf_{t,d}$, where the subscripts represent the term and the document, respectively [20].

The following definitions for this section are adapted from the study by Manning et al. [20].

$$tf_{t,d} = \text{the count of occurrences of query term } t \text{ in document } d$$

Document Frequency (df) is a technique for adjusting the importance of terms that appear frequently across many documents in a collection, which often have low discriminative power in relevance assessments. Instead of directly scaling term weights by collection frequency (the total number of times a term appears across all documents), it is more common to apply df. Document Frequency specifically refers to the count of documents containing a particular term, allowing us to reduce the term's weight based on how broadly it appears, thus improving relevance evaluations.

$$df_t = \text{the number of documents in the collection that contain a term } t$$

Inverse Document Frequency (idf) serves to adjust the weight of a term based on its occurrence across a collection of documents. Let N represent the total number of documents, and the idf for a term t is defined as follows:

$$idf_t = \log \frac{N}{df_t}.$$

This formulation indicates that terms that appear infrequently across the documents will have a higher idf score, whereas commonly occurring terms will gain a lower idf value.

The tf-idf weighting scheme combines the concepts of term frequency and inverse document frequency to create a composite weight for each term in every document. The weight assigned to term t in document d is defined as follows:

$$tfidf_{t,d} = tf_{t,d} \times idf_t.$$

This means that $tfidf_{t,d}$ achieves the highest value when term t appears frequently within a limited number of documents, thus providing significant discriminating power for those documents. Conversely, the weight is lower when the term appears less often in a document or when it is common across many documents, resulting in a diminished relevance signal. Finally, the weight reaches its lowest point when the term is found in nearly all documents.

2.4 BM25 Algorithm

The BM25 algorithm is a probabilistic retrieval model used to rank documents based on their relevance to a query [24]. It builds upon the Okapi BM11 [25] algorithm, adding key improvements that make it effective in balancing term frequency, inverse document frequency, and document length normalization to produce a relevance score for a document given a query. BM25 assumes that relevant and non-relevant documents follow different statistical distributions, an idea grounded in the probabilistic retrieval framework. In this thesis, we use BM25 as the initial retriever in the information retrieval pipeline. Its primary role is to retrieve an initial set of passages relevant to a given query, serving as the first phase in the retrieval process.

The key components of BM25 are as follows:

Term Frequency (tf): BM25 uses a modified term frequency, $f(t, D)$, to represent the number of times a term t appears in a document D. Unlike simple term frequency, BM25 accounts for saturation effects to prevent overly high tf values from disproportionately influencing the relevance score.

Inverse Document Frequency (IDF): Inverse document frequency, or IDF, assesses the rarity of a term across the entire corpus. BM25 assigns higher importance to rare terms and lower importance to common terms. IDF is calculated as:

$$\text{IDF}(t) = \log\left(\frac{N - n(t) + 0.5}{n(t) + 0.5} + 1\right)$$

where:

- N is the total number of documents in the corpus.
- $n(t)$ is the number of documents containing the term t.

Document Length Normalization: To address potential biases from varying document lengths, BM25 includes a normalization factor. Longer documents may inherently contain more instances of a term, which can lead to overestimation of relevance. Document length normalization is achieved by dividing the term frequency by the document's length and applying a normalization factor.

Query Term Saturation: BM25 further includes a saturation function to moderate the effect of very high term frequencies on relevance scores. Excessively high frequencies often correspond to less informative terms, and thus, this function reduces their impact.

The BM25 relevance score for a document D with respect to a query Q is defined as:

$$\text{BM25}(D, Q) = \sum_{t \in Q} \text{IDF}(t) \cdot \frac{f(t, D) \cdot (k_1 + 1)}{f(t, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

where:

- t is a term in the query Q.
- $f(t, D)$ represents the frequency of term t in document D.
- $|D|$ is the length of document D in terms of the number of words.
- avgdl is the average document length in the corpus.
- k_1 and b are hyperparameters:
  - k_1 (typically in the range [1.2, 2.0]) controls term frequency saturation.
  - b (typically around 0.75) controls the effect of document length normalization.

The combination of term frequency saturation, document length normalization, and inverse document frequency weighting in BM25 effectively captures the relevance of a document to a query, making it a popular choice in modern information retrieval systems.

CHAPTER 3 LITERATURE REVIEW

Information Retrieval (IR) and Recommender Systems are the central themes of this thesis. This chapter organizes the related work to highlight how each area contributes to the development of our methods and the motivations for this study. We begin by reviewing research on recommender systems, with a particular focus on explainable recommender systems, which informs our approach to enhancing transparency and interpretability in recommendation models. Next, we explore advancements in word embedding techniques, as these are integral to the content-based destination recommender system we develop. We then discuss various keyword and keyphrase extraction methods, which we apply to extract features from the descriptions of popular destinations for our airline partner. These extracted features are used to justify and explain the recommendations produced by our system.

Subsequently, we investigate research involving Large Language Models (LLMs), reviewing studies on prompt engineering, methods to enhance prompts, and strategies aimed at improving the reliability and consistency of LLM-generated responses.

Finally, we examine research on ranking models in IR systems, exploring the approaches proposed for each in the literature. We conclude by studying recent advancements in leveraging LLMs and evaluating their impact on the effectiveness of LLM-based re-rankers in IR systems, which is one of the primary goals of this thesis.

3.1 Recommender Systems

Prior work has demonstrated that Recommender Systems play a critical role in addressing the challenge of information overload, enhancing the online experiences of users by filtering vast amounts of data to present relevant content [26]. These systems provide personalized recommendations that cater to individual user preferences across a wide range of domains, including tourism [4, 27], entertainment [28], e-commerce [3, 29], and job matching [5, 30]. Studies indicate that by offering suggestions aligned with interests of users, recommender systems help users make more efficient and effective decisions, especially within human-centered online services like e-commerce platforms and social media sites [2]. Over the past few decades, advances in recommendation systems have yielded substantial benefits, including economic gains, time savings, and social impact [2].

Existing recommendation models are typically classified into three categories: collaborative filtering (CF) recommender systems, content-based recommender systems, and hybrid rec-

ommender systems, depending on the types of input data they leverage [31].

Collaborative filtering (CF) This recommender system model is one of the most extensively studied techniques, predicts user preferences by identifying patterns in user behavior, such as purchase histories or ratings, to anticipate future interactions. Previous research highlights Matrix Factorization (MF) as a prominent CF method that learns representations of users and items based on user-item interactions alone [32, 33]. MF-based models embed unique user and item identities into continuous vector spaces, enabling efficient calculation of matching scores in recommendation systems [34, 35].

Content-based This type of recommender systems leverage additional contextual information, such as user demographics or item descriptions, to enhance recommendation quality. Studies have shown that including textual information, for example, refines how these systems represent users and items, thereby improving the relevance of recommendations [36, 37].

Hybrid This category of recommender systems aims to overcome the limitations of individual techniques by combining multiple approaches. Previous work shows that the most effective hybrid methods often integrate collaborative filtering with content-based filtering, achieving improved accuracy and robustness in recommendation quality [38]. Through this integration, hybrid models address challenges unique to each technique while leveraging their strengths to deliver more tailored recommendations.

As part of this thesis, we develop a content-based recommender system for the most popular destinations offered by our airline partner, using detailed descriptions and information about each location.

3.2 Explainable Recommender Systems

Prior research defines explainability as the ability of a system to provide transparent, understandable explanations for its decisions, particularly beneficial in recommendation models [39]. In recommender systems, the integration of explainability has led to the development of explainable recommendation models, which not only suggest items but also clarify the **why** behind each recommendation [2]. Studies have shown that explainable recommendation systems improve user trust, satisfaction, and the overall effectiveness of recommendations by offering clear justifications for each suggestion [40]. This additional layer of interpretability allows users to better understand and interact with the system, while system designers benefit from enhanced tools for diagnosing, debugging, and optimizing recommendation algorithms.

Recent work highlights that explainable recommender systems represent a promising intersection of explainable AI and recommendation technologies, ultimately aiming to improve the trustworthiness of these systems [2].

Existing literature categorizes explainable recommendation models based on two primary taxonomies: the taxonomy for models and the taxonomy for evaluations [2].

Explainable recommendation models are categorized according to the following two main criteria:

- Producing explanations: Models may employ either intrinsic or post-hoc explanation techniques. In model-intrinsic approaches, explanations stem from the internal logic of a model, offering interpretability directly from its structure. Many recent studies have emphasized the effectiveness of intrinsic explanations as key components within recommendation frameworks [41]. Post-hoc techniques, on the other hand, are model-agnostic and generate explanations by analyzing only the inputs and outputs of the model, treating it as a black box [42, 43]. Prior work demonstrates that post-hoc explanations are often more versatile and well-suited for complex deep recommender systems, where interpreting millions of parameters directly is infeasible [44].
- Presenting explanations: Explanations can be presented in structured or unstructured formats. Structured explanations, often visualized through frameworks like graphs or knowledge graphs, offer well-organized, logic-driven insights into recommendations [45, 46]. Conversely, unstructured explanations provide simpler, user-friendly summaries or ratings directly from the recommendation model. Studies indicate that structured approaches better clarify user-item relationships, while unstructured methods, focusing on ease of understanding, support users in interpreting recommendations on a more intuitive level [47].

Evaluating the effectiveness of explainable recommendation systems is another focus of related research, with two main evaluation criteria emerging [2]:

- Evaluation Perspectives:Research highlights that perspectives such as Effectiveness, Transparency, and Scrutability are essential for evaluating explainable recommendation systems [48–50]. Studies show that these dimensions enable a holistic assessment of how well explanations improve the user experience and system transparency [2].
- Evaluation form: Evaluation methodologies in prior studies have typically fallen into four categories: Quantitative Metrics, Case Studies, Real-World Performance, and Ablation Studies. Quantitative metrics provide measurable assessments of explainability,

while case studies evaluate how well explanations align with human reasoning [2]. Real-world performance measures the practical impact of explanations on user interactions, and ablation studies examine the contribution of specific algorithmic components to the clarity and effectiveness of recommendations. These varied evaluation methods allow researchers to rigorously assess and refine explainable recommendation systems, as demonstrated in recent comprehensive studies [2].

As part of this thesis, we implement an explainable recommender system by leveraging post-hoc techniques and presenting explanations in an unstructured text format. For evaluation, we conducted a case study by consulting experts at the airline company to gather feedback on the results of our recommender system.

3.2.1 Feature-based Explanation

Feature-based explanations have closely aligned with content-based recommendation approaches in prior works. Content-based recommendation systems generate recommendations by matching user profiles to item attributes and features from potential choices, as noted in previous studies [51–53]. These recommendations are generally straightforward to explain due to their reliance on item features.

In different applications, content-based recommendations leverage item features to enhance personalization. For instance, the study in movie recommendations highlight features like genres, cast members, or directors, while book recommendations often consider genre, price, or authorship to align with user preferences. Research has demonstrated that presenting users with relevant item attributes that resonate with their preferences is a typical approach to feature-based recommendation explanations [40].

Some prior work has emphasized using specific item features to create recommendations and explanations. For example, Vig et al. [54] used movie tags as key features, presenting these features to help users understand why particular movies were recommended. This approach not only explained the relevance of movies to users but also enhanced their engagement with the system. Ferwerda et al. [52] conducted a user study that demonstrated a significant correlation between explanations and increased user trust and satisfaction in recommendations.

Studies have also explored different explanatory styles for presenting content features. Hou et al. [55] employed radar charts to illustrate the rationale behind item recommendations, showing both why specific items were recommended and why others were not. This approach provided users with clear visual cues about the relevance of each recommendation, enhancing their understanding and interaction with the system.

The primary goal of our airline partner project is to explain and recommend destinations to customers based on common features of the suggested cities. Therefore, we implement a feature-based explanation in this thesis.

3.2.2 Post Hoc Explainable Recommendation

Prior work has shown that when recommendation systems are too complex to provide clear explanations, post-hoc or model-agnostic techniques offer an effective solution by generating explanations separately from the recommendation model itself [55]. These techniques employ independent models to clarify recommendations after they have been made, hence the term post-hoc.

For instance, in many e-commerce platforms, recommendations are generated through sophisticated hybrid models, but explanations are often based on simple statistical insights, such as "70% of your friends purchased this item." Research has demonstrated that data mining methods, like frequent itemset mining and association rule mining, can create templates for such explanations, and the most suitable template is selected using post-hoc statistics like maximum confidence. Although these explanations are generated independently, studies affirm that they remain accurate, leveraging real data rather than the recommendation model itself [55].

Singh et al. [56] examined post-hoc explanations within learning-to-rank algorithms for web search, aiming to interpret decisions from a model-agnostic standpoint. They trained a black-box ranker and then applied the generated ranking labels as a training dataset for an interpretable tree-based model. This tree-based approach, acting as a post-hoc explanation tool, provided insight into ranking decisions in a point-wise manner. Additionally, Peake et al. [57] demonstrated a point-wise model for post-hoc explanations, while Singh's further work offered a pair-wise post-hoc explanation approach, addressing different explanation requirements in ranking contexts.

In the broader machine learning field, model-agnostic explanations often use simpler models to approximate the behavior of complex models locally, helping clarify decision-making processes around specific instances [40]. Ribeiro et al. [58] introduced LIME (Local Interpretable Model-agnostic Explanation), which leverages sparse linear models to approximate complex classifiers locally, identifying the features that most influence predictions. Building on this approach, Singh et al. [59] adapted LIME to ranking models by reformatting the task as a binary classification over query-document pairs, using linear SVM models to interpret relevance scores. Here, the model coefficients reveal words within documents that strongly indicate relevance.

Cheng et al. [60] advanced the mathematical foundations for post-hoc explanations in recommendation systems through influence analysis. By using influence functions, which measure the impact of individual training points on model predictions, they developed Fast Influence Analysis (FIA), a method that interprets predictions by tracing back to influential user-item interactions in training data. Their work demonstrated how past interactions shape latent factor model predictions, providing neighbor-style explanations by highlighting the most impactful interactions for each recommendation.

We use complex deep learning-based models to develop the destination recommender system, which makes it challenging to provide transparent explanations for its recommendations. To offer users clear explanations, we leverage different models rather than relying solely on the initial recommender system. Thus, in this thesis, we implement post-hoc techniques to improve transparency in recommender systems.

3.2.3 Evaluating Explainable Recommendations: A Case Study Approach

As demonstrated by previous studies, case studies are commonly used as a qualitative analysis method in explainable recommendation research, providing valuable insights into the rationale and effectiveness of explanation models. By offering concrete examples, case studies enable researchers to illustrate the explanation generation process, allowing users to intuitively evaluate its validity. These studies often include comparisons of explanations generated by different models to highlight when and why the proposed approach is effective, as well as its limitations. Such comparative analysis deepens understanding of strengths and weaknesses of each model, particularly since explanations are designed for human interpretation, which involves logical reasoning, intuition, and other subjective factors. As a result, concrete examples can enhance the credibility and trustworthiness of explanations [2, 40].

For example, Chen et al. [61] used case studies to clarify sequential recommendations, finding that many sequential recommendation patterns align with user behavior categories, such as "one-to-multiple" or "one-to-one." "One-to-multiple" refers to cases where a series of follow-up purchases stems from a single item, while "one-to-one" indicates that each purchase is triggered by the prior one. This study highlighted how these patterns aid users in understanding the reasoning behind recommended items and how these items relate to past purchases.

In another study, Hou et al. [55] applied case studies to evaluate user preferences, item quality, and the explainability of hotel recommendations. They introduced a metric called Satisfaction Degree on Aspects (SDA) to measure user satisfaction across various item attributes and conducted case studies to demonstrate how their model interprets and justifies recommendations, illustrating its approach to aligning recommendations with user preferences.

To evaluate the results of our recommender system, we conducted a case study with some experts at the airline company. For specific destinations, we provided recommendations from our model along with explanations and asked the expert to assess the accuracy of these suggested cities and their common features that could be proposed to customers.

3.3 Word Embedding

Prior research has revealed that word embedding, a continuous or distributed representation, serves as a foundational element in neural models of text semantics [62,63]. In this technique, words are encoded as dense vectors within a multi-dimensional space, where the spatial proximity of vector representations reflects semantic similarity, enabling neural models to capture semantic relationships between words effectively [64].

This thesis employs the following word embedding techniques to represent the descriptions of each destination in the development of a content-based destination recommender system and for the keyword matching process:

Word2Vec Model Mikolov et al. [65] introduced Word2Vec, a set of neural network-based model architectures designed to generate word embeddings from large-scale text data. Word2Vec relies on distributed representations to capture semantic word similarity by projecting words into a continuous vector space. Two primary architectures within Word2Vec, Continuous Bag-of-Words (CBOW) and Skip-Gram, are effective in capturing semantic relationships between words:

- CBOW Model: This model predicts a target word based on the context of surrounding words.
- Skip-Gram Model: In contrast, the Skip-Gram model takes a target word and predicts its context words.

GloVe: Global Vectors for Word Embedding Pennington et al. [66] introduced GloVe (Global Vectors), an unsupervised learning algorithm that generates word vector representations based on global word co-occurrence statistics from a corpus. GloVe builds embeddings by analyzing the frequency with which word pairs co-occur within a context window, capturing both global and local statistical information. By combining matrix factorization for global context with local context windows, GloVe captures word semantics comprehensively.

BERT: Bidirectional Encoder Representations from Transformers Devlin et al. [67] proposed BERT (Bidirectional Encoder Representations from Transformers), a pre-trained language model architecture that generates context-aware embeddings by analyzing sentences bidirectionally. Unlike Word2Vec or GloVe, which produce static word representations, BERT generates context-dependent embeddings that vary based on a context of a word within a sentence. The BERT architecture includes multiple layers of a bidirectional transformer encoder equipped with multi-head self-attention, allowing tokens to attend to various positions in the sequence and capture nuanced contextual information.

3.4 Keyword Extraction

The studies [68–70] defined keyword extraction as the automated process used to identify and extract the most significant words and phrases within a given text. The challenge of efficiently identifying relevant keywords from documents has a long history and is foundational to many applications across different fields. Solutions for keyword extraction significantly enhance tasks such as text summarization, clustering, thesaurus construction, opinion analysis, classification, query expansion, recommendation systems, data visualization, information retrieval, indexing, and digital libraries [71]. Generally, approaches to keyword extraction fall into two main categories: keyword assignment and keyword extraction. keyword assignment is related to a multi-label text classification problem in which a document is assigned with keywords from a predefined set or controlled vocabulary, often consisting of a dictionary or thesaurus relevant to the domain of the document. keyword extraction approach directly retrieves keywords from the document itself. This can be achieved through either unsupervised methods or Unsupervised approaches [68].

The baseline and fundamental method in unsupervised approaches is TF-IDF [72], which calculates keyword importance by comparing a term's frequency within a document to its overall frequency in a larger corpus. While effective and easy to implement, TF-IDF depends on access to a large corpus may not always be feasible. To overcome such limitations, alternative approaches have been developed. RAKE (Rapid Automatic Keyword Extraction) [70], for instance, identifies keywords by analyzing term co-occurrence patterns and frequency, producing a score based on the degree of connectivity and frequency of terms within a document. TextRank, another widely used algorithm, constructs an undirected term graph, linking terms that co-occur within a set word window. A ranking algorithm is then applied to prioritize terms based on their interconnectedness in the graph. Building on these foundational models, recent advancements like TopicRank [69] and PositionRank [73] have introduced new methodologies for incorporating contextual and positional information. Top-

icRank clusters terms into topic groups, allowing for topic-based ranking to capture broader document themes, while PositionRank factors in a term's position and frequency within a document to adjust its ranking dynamically, enhancing extraction in scholarly texts. Graph-based unsupervised approaches have further evolved with techniques like TopicalPageRank and CiteTextRank, which leverage topic distributions, often learned from external sources such as Wikipedia, to measure term importance relative to document topics [74]. More recent embedding-based methods use pre-trained word embeddings to determine keyword relevance. For instance, PositionRank [73] uses embeddings to assign edge weights in a word graph, running a weighted PageRank algorithm to calculate final scores for terms based on both local and global term relations.

One of the earliest attempts to automate keyword extraction as a supervised binary classification task was introduced by Turney [75] with the development of GenEx, a custom-designed algorithm that outperformed traditional models like decision trees. Among the most widely used supervised systems is KEA [76], which applies the Naïve Bayes algorithm and leverages two core features—TF-IDF and a term's first occurrence position—to classify terms as keywords or non-keywords. Building on this, Hulth [77] proposed a system that incorporates linguistic features such as part-of-speech (PoS) tags, combining them with traditional statistical features like term frequency and position. This additional linguistic information is then processed through a supervised rule-induction model using bagging, enhancing keyword identification accuracy. Several supervised approaches have refined these foundational models. Medelyan and Witten [78], for example, introduced enhancements to KEA, while CeKE [79], developed by Caragea et al., integrates citation network information, enriching keyword predictions by considering contextual relationships in academic literature. More recent work has employed neural network models, with Meng et al. [80] applying deep learning for scientific text keyword prediction and Gollapalli and Li [81] treating keyword extraction as a sequence tagging problem. Comprehensive overviews of these methods and related research are provided by Hasan and Ng [82] and Papagiannopoulou and Tsoumakas [83], who offer in-depth reviews of supervised keyword extraction methodologies and advancements.

The following two keyword extraction methods are leveraged in this thesis to extract keywords from each destination description, enhancing the explainability of the destination recommender system and justifying the recommendations make by our model:

YAKE! The study by Campos et al. [68] introduced YAKE! (Yet Another Keyword Extractor), a lightweight, statistical, unsupervised algorithm designed for automatic keyword extraction from single documents without the need for external corpora or training. The algorithm follows five main steps: (1) text pre-processing for candidate term identification,

transforming the document into a machine-readable format to improve candidate identification; (2) extraction of statistical features from individual terms; (3) computation of a heuristic term score that reflects term importance; (4) generation of candidate keywords via n-gram construction and scoring of each candidate; and (5) deduplication and ranking based on a similarity measure, which filters out overlapping or redundant keywords. YAKE! is corpus-independent and suitable for various domains and languages, as it does not rely on external resources such as WordNet, Wikipedia, or language-specific tools beyond a static stopword list. Uniquely, YAKE! can extract keywords containing interior stopwords (e.g., "Game of Thrones") with greater precision than conventional methods. Its linear scalability in document length, term frequency independence, and open-source availability further enhance its accessibility and versatility for keyword extraction tasks.

KeyBERT KeyBERT, introduced by Grootendorst [84], is a simple and efficient keyword extraction method that leverages BERT embeddings to generate keywords and keyphrases that closely represent the content of a document. Unlike traditional keyword extraction techniques, KeyBERT focuses on identifying the most contextually relevant keywords by assessing semantic similarity. The process begins by generating document embeddings with BERT to obtain a comprehensive representation of the document. Next, word embeddings are created for candidate N-grams within the text. Using cosine similarity, KeyBERT then calculates the similarity between each candidate N-gram and the document embedding to identify the most representative keywords and phrases. This straightforward yet powerful approach allows for the extraction of high-quality keywords that align closely with the main content of the document.

3.5 Keyphrases Extraction

Based on the prior studies [85, 86], keyphrase extraction is the task of selecting phrases that capture the main topics of a document. Keyphrases highlight the most important topics of a document and enable concise summarization. As shown in previous studies, keyphrases are essential for various downstream applications, such as classification [87], clustering [88], summarization [89], and document recommendation [90]. Additionally, keyphrases enhance information retrieval capabilities by supporting semantic search, faceted search [91], query expansion [92], and interactive document retrieval [93].

In prior research, keyphrases have been classified into two main types: extractive and abstractive. Extractive keyphrases are directly drawn from the text, whereas abstractive keyphrases are generated and may not be explicitly present within the document. These types are often

referred to as present and absent keyphrases, respectively [82]. Automated keyphrase identification methods are designed to identify extractive keyphrases within the document or to generate abstractive phrases that encapsulate the text's primary content effectively [94].

The following keyphrase extraction method is used in this thesis to extract keyphrases from each destination description, improving the explainability of the results generated by the destination recommender system:

Keyphrase Extraction Model: distilbert-inspec The model is a specialized, fine-tuned version of DistilBERT [95], optimized for keyphrase extraction on the Inspec dataset. Keyphrase extraction models are transformer models fine-tuned as a token classification problem where each word in the document is classified as being part of a keyphrase or not. It is designed to capture keyphrases that convey the main topics of scientific and technical documents, this model offers a concise summarization capability. By utilizing the efficient architecture of DistilBERT, it provides a balance between computational efficiency and accuracy, making it suitable for resource-constrained environments. Unlike traditional rule-based or statistical methods, this transformer-based model leverages contextual embeddings to enhance term significance within a document, allowing for more relevant keyphrase extraction [96].

3.6 Large Language Models

As demonstrated by previous studies, Large Language Models (LLMs), primarily based on Transformer architectures [8], represent a breakthrough in language modeling, incorporating vast parameter sets—often reaching hundreds of billions [7]. These models, trained on extensive textual datasets [97], exhibit remarkable capabilities in both understanding and generating natural language, advancing the field in previously unachievable ways. Research on widely recognized LLMs such as GPT-3 [98], PaLM [99], Mixtral [100], and LLaMA [101] has highlighted their ability to handle complex queries and execute a broad range of linguistic tasks, underpinned by emergent capabilities that arise with model scale.

Existing studies suggest that increasing the scale of LLMs—by expanding model size or augmenting training data—leads to enhanced performance across various applications, an insight consistent with the scaling law [102]. For example, larger models like GPT-3, with 175 billion parameters, have been shown to surpass smaller predecessors such as GPT-2 (with 1.5 billion parameters), exhibiting unique skills like few-shot in-context learning [7]. As a result, the term large language models has come to signify these high-capacity, pre-trained models.

Research has further explored the use of LLMs in conversational AI, particularly with models like ChatGPT, which is fine-tuned from the GPT series to deliver human-like responses and interactions [7]. This shift toward conversational agents reflects the broader historical evolution from task-specific statistical models to flexible, task-agnostic models, which now leverage contextual understanding for better adaptability. LLMs are thereby considered general-purpose models with unparalleled scope in task performance, reshaping the field of natural language processing.

LLMs have been applied across multiple domains, influencing areas such as information retrieval, where studies show that AI-driven systems—like enhanced search engines and chatbots—are transforming information access [1]. In multimodal settings, vision-language models are expanding the possibilities for cross-domain interaction [103, 104]. Current research highlights the integration of LLM-driven features in commercial applications, such as Microsoft 365's Copilot and the plugin architecture in ChatGPT, underscoring the growing ecosystem of LLM technology and its substantial impact across industries.

3.6.1 Prompting

Prior work [105] has established prompting as the primary method for leveraging LLMs in solving various tasks. Studies indicate that the quality of prompt design can significantly influence the performance of LLMs on specific tasks. The process of manually designing effective prompts, referred to as prompt engineering, has been shown to play a crucial role in eliciting targeted responses from LLMs across a range of applications [106, 107].

Key Ingredients Existing research identifies four core components that enhance the effectiveness of prompts for LLMs: task description, input data, contextual information, and prompt style [7].

- Task Description. Prior work emphasizes that a clear task description provides necessary guidance for the LLM, typically expressed in natural language. Studies show that adding keywords to specify input or output formats enhances clarity, which in turn helps the model respond more effectively to the prompt.
- Input Data. Researchers have demonstrated that descriptions of input data can vary depending on complexity; specialized data, like tables or knowledge graphs, may require formatting adjustments. For example, prior studies suggest linearizing structured data into sequences [108] or using programming-like syntax to improve compatibility with LLMs, which can facilitate accurate responses.

- Contextual Information. Existing studies also point to the inclusion of relevant supplementary information, such as retrieved documents in question answering tasks, as a way to boost response quality [109]. Evidence suggests that providing clear examples within prompts helps models understand and execute complex tasks with higher accuracy.

- Prompt Style. Research indicates that tailoring prompt style to fit the specific LLM improves task performance, with studies showing that approaches like using directive language ("Let us think step by step" [110]) or contextualizing prompts for specific expertise can enhance reasoning. For chat-based LLMs, breaking down complex prompts into manageable sub-tasks within a multi-turn conversation has also proven effective [111].

Design Principles Building on these core components, researchers have proposed several design principles to improve prompt efficacy for diverse tasks.

- Expressing the Task Goal Clearly. Studies recommend unambiguous task descriptions to reduce misinterpretation, enhancing model responses [112]. Effective prompts often outline specific task objectives, including input-output requirements (e.g., "Generate a concise summary of a document") and constraints (e.g., "summary length should not exceed 50 words").

- Decomposing into Sub-Tasks. Research suggests that complex tasks are more manageable when broken into sequential sub-tasks, allowing LLMs to address each component individually. For instance, one study highlights the effectiveness of listing sub-tasks in a structured, itemized format, enabling more accurate completion of intricate tasks [113].

- Providing Few-Shot Demonstrations. Prior studies on in-context learning demonstrate that including a few examples within the prompt can help LLMs map inputs to desired outputs effectively. Evidence shows that these few-shot demonstrations, which consist of input-output pairs, improve the LLM's performance on complex tasks without additional training.

- Utilizing Model-Friendly Formats. Research indicates that certain formats are better aligned with LLM training data, leading to improved understanding. Studies suggest using symbols such as `###` or `"""` to segment instructions from context. Additionally, for challenging tasks, translating the prompt to English has been shown to yield better results due to the predominance of English data in LLM training sets.

3.6.2 Chain-of-Thought Prompting

Chain-of-Thought (CoT) prompting has emerged as an advanced strategy for enhancing the reasoning abilities of LLMs in complex tasks. Prior studies demonstrate its effectiveness across domains such as arithmetic reasoning [114], commonsense reasoning [115], and symbolic reasoning [116]. Unlike standard in-context learning (ICL), which typically uses simple input-output pairs, CoT prompting introduces intermediate reasoning steps, creating a structured pathway that links inputs to outputs. Research shows that this sequence of reasoning steps helps guide LLMs through the logical processes needed to achieve accurate responses [116, 117].

In foundational CoT prompting work, each demonstration pair ⟨input, output⟩ is expanded to include a reasoning sequence, represented as ⟨input, CoT, output⟩, where CoT stands for the chain of intermediate steps linking input to output [116]. These studies indicate that presenting LLMs with such structured demonstrations enables the model to replicate this reasoning process with new inputs. Although manually annotated CoTs are typically needed due to their complexity, simple cues, like "Let's think step by step" [110], have been shown to prompt models to generate reasoning sequences on their own. Variations of these prompts, such as "Take a deep breath and work on this problem step-by-step" [118], have further enhanced model performance by encouraging more detailed reasoning. CoTs are usually produced as natural language sequences. However, for tasks that demand high precision and logical accuracy, recent studies reveal that code-based CoTs can be more effective [119]. Additionally, research has explored dynamically switching between text and code formats for CoTs, allowing models to combine the readability of text with the structure of code for improved outcomes in tasks requiring detailed reasoning [120].

3.6.3 Enhanced CoT Generation

To address the issues of incorrectness and instability in the generation of CoT by LLMs, various enhanced CoT generation methods have been developed [18, 121]. These methods typically fall into two categories: sampling-based and verification-based techniques, both aiming to improve the quality of CoT generation. Since this thesis focuses on the self-consistency [18] concept, a subcategory of sampling-based methods, we will limit our discussion to sampling-based approaches.

LLMs often exhibit instability during inference, which can lead to unfaithful reasoning steps [7]. To address this issue, recent research has explored alternative decoding strategies beyond traditional greedy methods. One widely adopted approach is self-consistency,

introduced by [18]. Rather than generating a single reasoning path, self-consistency samples multiple reasoning paths during inference. After generating several potential solutions, the method performs an ensemble over the corresponding answers and selects the most consistent one through a majority voting process. While this approach reduces errors, it can still be lead to wrong answers if most paths are mislead. An alternative method [122] refines this process by selecting the k most complex reasoning paths, as more detailed reasoning paths usually have better results. Multi-path CoT Reasoning (MCR) [123] further enhances this by referencing reasoning steps from other paths during generation, creating across multiple reasoning paths to generate the final answer.

3.6.4 Planning for Complex Task Solving

Prompting with ICL and CoT reasoning has emerged as a versatile approach for handling a variety of language-based tasks. Despite its flexibility, this approach encounters challenges with more complex tasks, such as mathematical reasoning [124] and multi-hop question answering [125]. To address these limitations, prompt-based planning has been introduced, offering a method for decomposing intricate tasks into manageable subtasks and establishing a structured plan of actions to guide task completion.

Prompt-based planning generally involves three core components: a task planner, a plan executor, and an environment . The task planner, which is played by LLMs, is responsible for generating a comprehensive plan to tackle the target task. The plan can be presented in various forms, e.g., an action sequence in the form of natural language [126] or an executable program written in programming language [127]. Then, the plan executor, the second component, is tasked with carrying out the actions defined in the plan. It can be implemented by models like LLMs for textual tasks [17] or by tools like code interpreters for coding tasks [128]. The environment, which forms the third component, refers to where the plan executor carries out the actions, which can be set differently according to specific tasks, e.g., the LLM itself [129] or an external virtual world like Minecraft [130].

In text-based approaches, LLMs generate plans as natural language sequences that outline actions for the plan executor to follow in solving complex tasks. Here, LLMs can be directly prompted to create a sequence of actions. For example, Plan-and-Solve [17] uses specific instructions like “devise a plan” to prompt the LLM for zero-shot planning, while elfplanning [131] adds demonstrations within the prompt to guide the LLM through ICL, encouraging a step-by-step planning process.

In this thesis, we leverage the plan-and-solve prompting approach to guide LLMs in generating relevance scores for passages concerning a query within IR systems.

3.6.5 Self-Consistency in ranking task

LLMs demonstrate positional bias in how they use context. This particularly affects listwise ranking. The study by [132] proposes Permutation Self-Consistency, a decoding strategy aimed at improving the quality, consistency, and prompt-order invariance of LLM. The idea behind this strategy is to marginalize different list orders in the prompt, generating an order-independent ranking with reduced positional bias. Their findings reveal that while self-consistency does not generalize to listwise ranking, permutation self-consistency enhances listwise ranking in LLMs. In this thesis, we examine the impact of self-consistency on the quality of the relevance scores generated for each document in relation to the query, within the context of pointwise ranking by LLMs.

3.6.6 Confidence Elicitation in LLMs

Confidence elicitation refers to the process of estimating the confidence of a LLM in its responses without requiring model fine-tuning or access to internal workings [19].

Within this context [133] introduced the concept of verbalized confidence, which prompts LLMs to explicitly express their confidence in predictions. However, their approach primarily focuses on fine-tuning models on specific datasets where confidence is available, leaving the model's ability to generate confidence in a zero-shot setting largely unexplored. The external calibrator proposed by [134], relies on internal model representations, which are frequently inaccessible. The two other works [19,135] are mainly focused on the use of prompting strategies in confidence elicitation. Studies [19], have empirically evaluated the capacity of LLMs to express uncertainty, revealing gaps in the models' ability to provide reliable confidence estimates. This research shows that while LLMs are capable of generating confidence scores, these are often poorly calibrated, especially in scenarios involving high uncertainty. This thesis investigates the impact of confidence elicitation within the context of pointwise ranking by LLMs.

In this thesis, we leverage LLMs to address ranking tasks within IR systems, aiming to improve the relevance and accuracy of retrieved information. We introduce innovative prompt designs that incorporate advanced techniques, including COT reasoning, self-consistency, and confidence elicitation. These techniques work together to enhance the decision-making capabilities of LLMs, enabling more nuanced interpretations of query relevance. By implementing COT reasoning, we facilitate a step-by-step approach to complex queries, breaking down reasoning tasks to improve accuracy. The self-consistency mechanism helps validate outputs by aggregating multiple model responses, leading to more reliable and robust results.

Additionally, confidence elicitation enables the model to assess the certainty of its predictions, providing a confidence measure that can be used to rank outputs more effectively. Collectively, these prompt engineering strategies advance the performance of IR systems, contributing to more precise and contextually relevant information retrieval.

3.7 Ranking Models in IR Systems

Ranking plays a fundamental role across diverse applications in IR, including document retrieval [136], collaborative filtering [137], key term extraction [138], sentiment analysis [139], and product rating [140]. In these tasks, models or functions are developed to generate ranked lists where object order reflects relevance, preference, or significance based on specific application contexts. Among these, document retrieval has gained particular attention [6].

Three main categories of ranking models have been widely studied and proven effective: probabilistic retrieval methods, Learning to Rank (LTR) methods, and deep learning-based methods.

Bag-of-Words Retrieval

Traditional bag-of-words (BOW) methods, such as BM25 and TF-IDF, have been foundational for document retrieval, representing queries and documents as sets of words [24,25,141]. Research in this area has focused on advanced term-weighting techniques to assess relevance by measuring term overlap. This approach enables efficient retrieval at scale across large document collections.

However, BOW methods exhibit limitations due to their reliance on exact term matches, which restricts their ability to retrieve semantically relevant documents when query terms are absent. This issue, known as the vocabulary mismatch problem, has been well-documented [142], prompting the development of more sophisticated models that capture deeper semantic relationships.

Learning to Rank (LTR)

LTR methods advance beyond traditional BOW approaches by using machine learning to rank documents based on learned features and signals [**?**, 6, 143]. Rather than relying solely on term frequencies or document lengths, these methods automatically learn model parameters from user interaction logs, which capture behavioral patterns such as clicks and query reformulations.

LTR approaches can be categorized into three primary types:

Pointwise Approach Pointwise LTR methods rank individual documents by defining a loss function at the document level. Previous research categorizes pointwise approaches into regression-based, classification-based, and ordinal regression-based models [144, 145].

Pairwise Approach Pairwise methods use document pairs as training data, formalizing the task as pairwise classification. Studies have developed models like Ranking SVM [146], RankBoost [147], and RankNet [148] to predict the relative ordering between pairs of documents, yielding substantial improvements in rank precision.

Listwise Approach Listwise LTR methods consider entire lists of documents as training instances, optimizing a loss function over the list. Techniques such as ListNet [149], RankCosine [150], and StructRank [151] exemplify this category. More recent research has explored optimizing directly for IR metrics, as in AdaRank [152] and SoftRank [153], highlighting the flexibility of listwise models for ranking tasks.

Deep Learning-Based Methods

Recent advances in deep learning have demonstrated strong performance across many natural language processing tasks, including named entity recognition [154], sentiment analysis [155], and semantic similarity [156]. This capability has extended to document ranking models, where deep learning methods offer the advantage of automatically learning complex language patterns.

Deep neural networks (DNNs) have become the foundation for many deep learning-based ranking models, as highlighted by recent literature [157]. Unlike LTR methods that depend on hand-engineered features, DNNs derive high-level representations directly from raw text, enabling them to perform "soft matching" between queries and documents. This allows retrieval to occur in a continuous vector space, supporting nuanced comparisons that go beyond exact term matching.

Despite their strengths, deep learning-based ranking models face notable challenges. Studies have shown that their effectiveness improvements over LTR models are sometimes incremental. Moreover, DNN models are generally computationally demanding and require substantial amounts of training data, which limits their performance in zero-shot or few-shot scenarios. Additionally, the high computational cost of soft matching remains a barrier for large-scale retrieval applications [158].

3.7.1 Ranking with LLMs

Recently, extensive research has been conducted on the effectiveness of LLMs under zero-shot settings in document ranking in IR tasks, yielding impressive results [9, 10, 12, 14, 15].

The methodologies for using LLMs in zero-shot ranking tasks can generally be divided into four main approaches: Pointwise [9, 13, 15], Listwise [10, 14], Pairwise [12], and Setwise [11]. These approaches use various prompting strategies to guide the LLM in generating a relevance estimation for each candidate document.

Pointwise

Prior work has explored pointwise ranking through two main prompting techniques: generation-based and likelihood-based approaches. Liang et al. [15] and Nogueira et al. [16] used a generation-based method where the LLM is prompted to determine relevance through a "yes/no" task, ranking documents based on the normalized likelihood of generating a "yes" response. Sachan et al. [13] further extended this by employing query likelihood modeling (QLM), where an LLM, prompted with a document, generates a relevant query, and the model ranks documents based on the likelihood of reproducing this query. Zhuang et al. [9] introduced a fine-grained pointwise approach, incorporating relevance labels such as "Highly Relevant," "Somewhat Relevant," and "Not Relevant" rather than a binary choice. This allows the model to better differentiate between documents with varying levels of relevance to the query, leading to more accurate rankings.

Listwise

Previous studies on listwise ranking [10, 14] have demonstrated that the primary objective of the listwise approach is to directly rank a list of documents for a given query, producing a ranked list of document labels based on their relevance to the query. In these studies, both the query and a list of documents are directly input into a prompt. Due to prompt length constraints in LLMs, a sliding window method is employed, re-ranking a subset of documents within each window iteration.

Pairwise

Pairwise ranking methods have also shown promise in zero-shot LLM applications [12]. Qin et al. demonstrated the effectiveness of presenting LLMs with a query and document pairs, instructing them to identify which document is more relevant. This pairwise approach ranks

documents by iteratively comparing pairs based on their relative relevance, resulting in an ordered document list.

Setwise

Recent work by Zhuang et al. [11] introduced the setwise ranking method for LLMs. By prompting an LLM with a query and a set of candidate documents, the model selects the most relevant document from the set. This approach reduces the number of comparisons needed by applying sorting algorithms, such as heap sort, which efficiently rank documents by to comparing more than two documents at each step, thereby reducing the total number of comparisons and speeding up the sorting process.

In this thesis, we introduce novel techniques and prompt designs for LLMs aimed at enhancing the ranking performance within IR pipelines. Our approach leverages multiple strategies, including CoT prompting, Plan-and-Solve prompting, self-consistency prompting, and confidence elicitation to derive more accurate relevance scores and rank the passages accordingly. Specifically, our pipeline begins with a BoW retrieval method, such as BM25, serving as the initial retriever to generate a ranked list of relevant passages. We then add a LLM-based pointwise pre-filtering step, before the re-ranker, to decrease the number of irrelevant passages before passing to the re-ranker. Furthermore, we conduct an empirical study to investigate the role of self-consistency and confidence elicitation in LLM-based pointwise re-rankers. The use of advanced prompting techniques such as CoT and plan-and-solve facilitates a structured reasoning process within the LLM, while self-consistency prompting enables us to improve the robustness and reliability of relevance scores by running the model multiple times for each query-passage pair. Additionally, confidence elicitation techniques allow the LLM to express uncertainty levels associated with each relevance decision, providing deeper insights into the ranking rationale of the model.

CHAPTER 4 ENHANCING THE PERFORMANCE OF A DESTINATION RECOMMENDER SYSTEM FOR AN AIRLINE PARTNER USING FEATURE EXPLAINABILITY AND KEYWORD EXTRACTION

4.1 Introduction

In the past, the destination recommender system of our airline partner operated with a limited scope and lacked the level of explainability that modern recommender systems offer. The system primarily relied on basic heuristics, such as analyzing a travel history of customers or identifying similar customer profiles, to generate recommendations. While this approach did provide suggestions based on some level of historical patterns, it had several significant limitations.

One of the primary issues with the previous system was its lack of personalization. Rather than offering truly individualized recommendations, the system would often recommend a common set of destinations to groups of customers with similar travel histories, without considering the unique characteristics and preferences of each individual. As a result, customers were often presented with recommendations that didn't fully align with their specific travel interests or goals, diminishing the overall value and relevance of the suggestions.

More critically, the old system lacked explainability. It could not provide clear or meaningful reasons why particular destinations were recommended to each customer. As travel decisions are often influenced by a variety of personal factors—such as preferred climate, activities, culture, or entertainment options—the absence of an explanation for why certain cities were recommended left many customers feeling uncertain about the relevance of these suggestions. This lack of transparency hindered the ability of our airline partner to build trust with their customers, as they were unable to communicate the specific attributes or experiences that might make a destination an appealing choice.

The limitations of this approach became increasingly apparent as customer expectations evolved. Modern travelers, particularly those engaging with advanced technology in other areas of their lives, now expect personalized, data-driven recommendations that feel aligned to their unique interests. They are also more likely to engage with and trust recommendations when given clear explanations of why each destination is a good match for their preferences.

In light of these issues, our airline partner recognized the need to enhance its destination recommender system to offer not only greater personalization but also robust explainability. The goal was to develop a system capable of identifying and highlighting the key characteristics of

each destination—such as natural features, cultural events, or entertainment options—that align with the individual interests of travelers.

4.2 Preliminary Experiment

Before starting the development phase of the destination recommender system at the airline company, we consulted with experts from the customer loyalty team to understand their expectations and opinions about the new system. The following statements are paraphrased from informal interviews with two experts working at the industrial partner:

1- What are your expectations from Recommender Systems?

My main expectation for the recommender system is that it should be flexible enough to address specific business needs. Its primary objectives include detecting similarities between cities for targeted marketing, providing explainability, and establishing a mathematical framework to leverage individual customer activities. This will enable us to offer personalized recommendations or to segment customers based on their travel interests and make tailored recommendations to each segment.

I expect the system to accurately identify key attributes and characteristics of each city we serve and to compare these characteristics across cities, identifying those with the most similarity. It should be intelligent in its comparisons, providing diverse recommendations based on distinct features. For example, a city similar to "Cancun" could include other beach destinations or cities known for vibrant nightlife, even if they lack beaches.

Additionally, I anticipate that the system will rely on the latest technologies and integrate data from various sources to precisely capture unique traits of each city. This includes using advanced NLP models, such as deep learning models like BERT or count-based models like TF-IDF, and exploring creative data sources (e.g., web scraping, AI-driven tools like ChatGPT). It is also essential that the system is rigorously tested to remove any biases that could skew recommendations.

Beyond using cutting-edge technology and high-quality data, the model should be adaptable, continually evolving as new data sources and cities are added. This ongoing evolution will ensure it remains relevant and aligned with recent customer trends and changes in destinations. Therefore, the model should be regularly reviewed, with new data sources evaluated for meaningful improvements. Finally, as a versatile tool, the model is expected to serve marketing as well as other organizational needs, supporting various teams to address their business objectives.

2- Which performance indicators are important for you when considering Recommender Sys-

tems ?

Evaluating a content-based recommender system can be challenging, as some results may appear subjective. Initially, I recommend building an expert panel to assess if the recommendations of the system align well with expected results. However, the key metric of success will come from real-world testing with customers. We should conduct controlled experiments, comparing customer engagement across three groups: those using the new model, those with the old model, and those receiving random recommendations.

By tracking customer engagement and booking rates, we can assess if the new model drives higher interaction and conversion compared to the old model and random recommendations. This setup will allow us to statistically measure the differences between groups and determine the effectiveness of the new model.

3- What are your preferences when considering Recommender Systems?

I prefer a flexible model that can continuously adapt to the latest customer trends and changes in destinations. The model should be efficient, providing timely results without long delays, which makes using pre-trained models like BERT or GloVe crucial. Personalization is also a high priority: the model should offer explainable, individualized recommendations based on recent activity of each customer. This personalized approach is essential to enhancing the overall customer experience.

4.3 Approach

In response to the objective of our airline partner to develop a destination recommender system that combines personalized suggestions with robust explainability, we propose a content-based, feature-driven recommendation model. This system is designed to overcome the limitations of the previous heuristic-based approach by leveraging advanced techniques that offer a higher degree of personalization and provide transparent explanations for each recommendation. By focusing on both the individual characteristics of each destination and the unique preferences of each traveler, this model aims to increase customer engagement, satisfaction, and trust in the recommendation process.

Our approach centers on a content-based recommendation model due to its alignment with the requirements of the airline partner for transparency and adaptability. The model will operate by analyzing the features of destinations (such as climate, local attractions, cultural events, and entertainment options) and matching them with the recent travel histories and stated preferences of individual customers. This method will not only identify relevant destination attributes but also serve as the basis for post-hoc explainability, an essential

component for meeting the expert expectations outlined by the customer loyalty team at the airline company.

To achieve the required level of personalization and flexibility, we integrate the latest embedding models, particularly those based on deep learning, such as BERT and similar transformer-based architectures. These embeddings capture semantic relationships between destinations and facilitate comparisons based on a wide array of destination features. However, because these advanced models are inherently complex and may lack direct interpretability, we implement a post-hoc explainability layer. This layer translates the underlying data into clear, understandable reasons for each recommendation, ensuring customers receive transparent explanations as to why a specific destination has been suggested.

Given that the customer loyalty team of the industrial partner has emphasized the importance of certain touristic keywords for marketing and communication purposes, we incorporate keyword and keyphrase extraction techniques to broaden the scope of relevant features. By employing a range of extraction methods, we identify key attributes that resonate with each destination and align with experts' expectations. To further refine these results, we also explore synonyms and related terms, expanding the analysis beyond exact keyword matches. This comprehensive approach will ensure the model remains flexible, relevant, and able to adapt as the preferences of customers evolve. To ensure the accuracy and relevance of the recommender system, we have weekly sessions with experts from the customer loyalty team. These sessions allow us to assess the effectiveness of the recommendations in meeting expectations of experts for personalization, explainability, and alignment with customer interests. During these meetings, the experts review the recommended destinations and the feature-based explanations provided for each suggestion, offering their feedback.

4.3.1 Dataset

To develop a robust content-based recommender system, we need comprehensive, contextual data about destinations of the airline company. For this purpose, we collected data for the 200 most popular destinations of that company, using a range of reliable and diverse sources to capture the unique characteristics of each location. These sources include:

- Wikipedia: A free, widely-used online encyclopedia maintained by a community of volunteers. It provides comprehensive information about destinations, covering aspects such as history, culture, landmarks, and climate.[1]

[1] `https://en.wikipedia.org/wiki/Main_Page`

- Wikitravel: A global, community-driven travel guide that offers detailed, practical travel information. Wikitravel entries are structured to give insights into local attractions, dining, nightlife, and other travel tips.[2]

- ChatGPT: For each destination, we used ChatGPT to generate additional contextual information, providing a broad overview of the unique features of each city. This includes notable attractions, cultural highlights, and potential travel experiences that align with customer interests.

- Internal Excel File (Destination Guides): This internal dataset, provided by the airline partner, contains information about each destination, created by teams in that company. It includes key destination details that are specifically relevant to customers of the company based on business objectives.

- Air Canada Vacations: This website offers a variety of vacation packages that combine flights, hotels, cruises, tours, and car rentals. Information from Air Canada Vacations highlights popular travel packages and top-rated activities.[3]

- Viator: A global booking platform for tours, activities, and travel experiences, Viator provides detailed information on various activities available in each destination. This platform is a valuable source for understanding the types of experiences customers might seek, enhancing the ability of the recommender system to match customers with relevant travel options.[4]

By combining data from these diverse sources, we ensure that the recommender system has a rich foundation to capture the unique attributes of each destination.

4.3.2 Developing the Destination Recommender System

To build an effective destination recommender system, we utilized various embedding methods to convert data for each destination into vector representations, enabling the model to interpret and compare textual information. We experimented with three different embedding models, each with unique characteristics:

- TF-IDF Vectorizer: This feature extraction technique, available in the scikit-learn library,[5] converts raw text documents into a matrix of TF-IDF features. TF-IDF helps

[2] https://wikitravel.org/en/Main_Page
[3] https://vacations.aircanada.com
[4] https://www.viator.com/en-CA/
[5] https://scikit-learn.org/stable/

highlight the importance of words in each document, making it suitable for identifying key terms associated with each destination.[6]

- en_core_web_lg: This is the largest English language model in the spaCy library,[7] with a size of 788 MB. It uses GloVe word embeddings, which capture semantic relationships between words and are well-suited for understanding general language patterns relevant to travel and destination descriptions.[8]

- all-mpnet-base-v2: Developed by Sentence-Transformers,[9] this model, based on the BERT architecture, maps sentences and paragraphs into a 768-dimensional dense vector space. It is designed for tasks like clustering and semantic search, making it highly effective for capturing nuanced semantic relationships between destinations based on their descriptions.[10]

Using these models, we embed the textual data for each destination and then computed similarity scores by applying cosine similarity. This approach allows us to quantify how closely related two destinations are, based on their contextual attributes, which is essential for making accurate and meaningful recommendations.

4.3.3 Expert-Defined Touristic Keywords

The experts in customer loyalty team at our airline partner have provided a list of essential and commonly keywords to be incorporated when recommending destinations to customers. These keywords reflect the different of a destination that are likely to resonate with interests and preferences of travelers.

The keyword list encompasses various themes, including activities, attractions, and cultural elements. Figure 4.1 presents a keyword cloud illustrating these terms.

To ensure the suggestions of the recommender system align with these keywords, we need to justify the recommendations based on their presence in each description of the destinations. In the following, we apply keyword and keyphrase extraction techniques to detect these terms within the destination content, providing a foundation for interpretable and keyword-relevant recommendations.

[6] https://scikit-learn.org/1.5/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[7] https://spacy.io

[8] https://spacy.io/models/en#en_core_web_lg

[9] https://www.sbert.net

[10] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

Figure 4.1 Touristic keywords defined by experts

4.3.4 Feature Explainability Techniques

To implement feature explainability and makeing the results of our destination recommender system more transparent, we employ various keyword and keyphrase extraction techniques, ranging from statistical approaches to more advanced transformer-based methods. Specifically, we implement YAKE! [68], a statistical extraction technique, and KeyBERT [84], a pre-trained language model-based approach using BERT. Additionally, we incorporate a multi-word phrase extraction technique that leverages transformer models, specifically 'keyphrase-extraction-distilbert-inspec' [96], to capture keyphrases that capture multiple words, thus providing more contextually rich information.

Implementation of YAKE! We configure the following parameters for the YAKE! implementation:

- language = "en": Specifies the language of the text.
- max_ngram_size = 3: Sets the maximum length of n-grams for keyword extraction.
- deduplication_threshold = 0.5: Controls the deduplication limit, helping to avoid repetitive terms.
- numOfKeywords = 40: Specifies the number of keywords or keyphrases to extract.

These parameters are optimized to capture relevant phrases up to trigrams, balancing detail with briefness in keyword lists.

Implementation of KeyBERT For our KeyBERT setup, we use the following configurations:

- model = "all-mpnet-base-v2": Embedding model to represent documents and words.
- keyphrase_ngram_range = (1, 2): Defines the n-gram range for extracted keyphrases, allowing both unigrams and bigrams.
- use_mmr = True: Activates Maximum Marginal Relevance (MMR) to ensure diversity in extracted keywords.
- top_n = 40: Specifies the number of keywords or keyphrases to extract.

The length of the extracted keywords can be customized by adjusting the ngram parameter in both of the above techniques. After testing different values, we found that limiting n-grams to a length of 2-3 produces more accurate and contextually relevant phrases, enhancing the overall interpretability of the extracted features.

Implementation of Transformer-Based Keyphrase Extraction In addition to YAKE! and KeyBERT, we use a dedicated keyphrase extraction model, keyphrase-extraction-distilbert-inspec, that operates on multi-word phrases. Unlike the previous techniques, this approach is specifically tuned for extracting phrases, and it does not require extensive parameter tuning. Instead, we use the default settings of the model, which are optimized for identifying significant keyphrases in various contexts.

Combining Extraction Results After applying each method, we merge the outputs from YAKE!, KeyBERT, and the DistilBERT keyphrase extraction model to create a comprehensive list of keywords and keyphrases. This aggregated list is used for each destination, providing a nuanced, multi-faceted representation of important terms.

Mapping Extracted Keywords to Desired Keywords One challenge that arises is mapping the extracted keywords to a predefined list of desired keywords identified by domain experts. To address this, we employ the word2vec_sample model, a pre-trained word embedding model based on a dataset of 100 billion words from Google News, representing each word in a 300-dimensional space. For each keyword in our list, we retrieve the top 10 semantically similar words using the word2vec. The purpose of using this model here is to broaden the search for keywords by including similar words, so that the analysis is not limited to just exact matches of the specified keywords. This way, if a keyword like beach is in the list of expert-defined keywords but is not directly found in the text data, we can still detect related concepts like shore, coast, or sand.

The following step-by-step actions are as follows:

- Keyword Similarity Mapping: For each expert-defined keyword, we use Word2Vec to find words with similar meanings or associations. This creates a dictionary of each keyword and its set of similar words.
- Text Matching: When processing the text and description for each destination, we check for occurrences of both the exact keywords and their similar words. If a similar word appears in the city text, it is considered a match for the original keyword, and the keyword is added to the list of features for that destination.
- Outcome: We save the matched keyword (not the similar word itself) as the feature associated with the destination. This allows us to capture the expert-defined concepts even when they appear in slightly different forms in the text.

This approach helps ensure that the main concepts or features of each city are captured more accurately, even if the exact terms from the keywords list are not present in the text data.

4.4 Experimental Results

In this section, we present an in-depth analysis of the results obtained from various models and techniques used in developing an explainable destination recommender system. We detail the performance of each approach, moving progressively from simpler models to more complex ones to highlight the strengths and limitations of each in providing relevant, personalized travel recommendations. Our aim is to demonstrate how these models contribute to improving the quality of recommendations, ultimately enhancing the overall user experience.

4.4.1 Recommendation System Results

The following subsections showcase the recommendation results from the content-based recommender system. Tables 4.1, 4.2, and 4.3 illustrate the top recommended cities for three sample input cities—Zurich, Paris, and Tokyo—generated using the TF-IDF, en_core_web_lg, and BERT models, respectively. By comparing the results across models, we analyze the degree of relevance and similarity of the recommendations to the main city based on varying levels of model sophistication.

The results in tables 4.1, 4.2, and 4.3 reveal notable differences in the relevance and diversity of the recommended cities across models. With the `TF-IDF` approach, recommendations are generally related to cities that share certain textual characteristics with the main city. However, the `en_core_web_lg` model, which leverages more sophisticated word embeddings,

Table 4.1 Recommended cities for three different cities based on Tf-Idf approach.

Main City	**Recommended Cities**
Zurich	Geneva, Hamilton (Ontario), Sault Ste. Marie (Ontario), Greater Sudbury, Munich, Bogota, Kingston (Ontario), Regina (Saskatchewan), Milwaukee
Paris	Rome, Lisbon, Santiago, Mexico City, Madrid, Milan, Athens, Brussels, São Paulo
Tokyo	Seoul, Osaka, Copenhagen, Washington D.C., Warsaw, Doha, Beijing, Budapest, Bogota

Table 4.2 Recommended cities for three different cities based on en_core_web_lg model.

Main City	**Recommended Cities**
Zurich	Copenhagen, Budapest, Vienna, Frankfurt, Munich, Reykjavík, Amsterdam, Warsaw, Milan
Paris	Geneva, Lyon, Montréal, Rio de Janeiro, Barcelona, Mexico City, Brussels, Bogota, Santiago
Tokyo	Seoul, Shanghai, Osaka, Reykjavík, Beijing, Vienna, New Delhi, London, Istanbul

Table 4.3 Recommended cities for three different cities based on BERT model.

Main City	**Recommended Cities**
Zurich	Geneva, Vienna, Frankfurt, Nice, Paris, Shanghai, Brussels, Amsterdam, Rome
Paris	New York City, Lyon, New Orleans, Montréal, Las Vegas, Nice, Chicago, Algiers, London
Tokyo	Osaka, Honolulu, Beijing, New York City, Seoul, San Francisco, Portland (Oregon), Shanghai, Washington, D.C.

offers improved relevance by identifying deeper connections based on linguistic and semantic similarities.

Finally, the `BERT` model, a more advanced transformer-based model, provides recommendations that exhibit a greater understanding of contextual nuances, leading to highly relevant and culturally or geographically similar cities. For instance, the recommendations for Tokyo with BERT include cities like Seoul and Osaka, which are culturally aligned and geographically proximate. This progression highlights that as the models become more advanced, the quality of recommendations improves, aligning more closely with implicit expectations of users based on historical travel patterns.

4.4.2 Feature Explainability Results

As discussed in the previous section, we employ three different keyword and keyphrase extraction techniques to enhance the explainability of our destination recommender system. The techniques—YAKE!, KeyBERT, and keyphrase-extraction-distilbert-inspec—allow us to identify distinctive phrases and features that characterize each recommended city. This level of explainability provides users with insights into why a particular city is recommended, based on its unique attributes, landmarks, or cultural highlights.

Tables 4.4, 4.5, and 4.6 present the extracted keywords and keyphrases for three main cities—Zurich, Paris, and Tokyo—demonstrating the results for each technique and showcasing how each approach captures the essence of these cities from different perspectives.

Table 4.4 Keyphrases for three different cities by YAKE!

City	**Keyphrase**
Zurich	Swiss National Museum, Zurich Film Festival, Zurich Tram Museum, Lake Zürich
Paris	Paris Fashion Week, Eiffel Tower, Musée Picasso Paris, Louvre Museum
Tokyo	Tokyo National Museum, Tokyo Metro Ginza, Tokyo Tower Aquarium, Tokyo Imperial Palace

The results from each technique provide meaningful and contextually relevant features for each city, reflecting the specific cultural and historical attributes associated with each destination.

Table 4.5 Keyphrases for three different cities by KeyBERT

City	**Keyphrase**
Zurich	Zurich heritage, European artworks, Romanesque architecture, BolteLang Galerie
Paris	Landmarks Eiffel, Iconic Louvre, Parisian skyline, Grands Boulevards
Tokyo	Tokyo temples, Shinto architecture, Museum Shinjuku, Landmarks imperial

Table 4.6 Keyphrases for three different cities by keyphrase-extraction-distilbert-inspec

City	**Keyphrase**
Zurich	Antique shops, Art galleries, Historic buildings, Magnificent Lake Zurich
Paris	Art museum, City of lights, Eiffel Tower, Louvre Museum
Tokyo	Ancient temples, Art galleries, Imperial history, Memorable shopping

For instance, YAKE! highlights key events and iconic landmarks such as "Zurich Film Festival" and "Tokyo Imperial Palace," while KeyBERT captures architectural and cultural elements, such as "Romanesque architecture" for Zurich and "Shinto architecture" for Tokyo. The `keyphrase-extraction-distilbert-inspec` model, however, appears to produce a more diverse set of keyphrases that encompasses broader aspects of each city, capturing not only landmarks but also experiences and general characteristics such as "City of lights" for Paris and "Ancient temples" for Tokyo.

By employing these keyword and keyphrase extraction techniques, we enhance the transparency of the recommendation system, offering users insights into the unique characteristics of each recommended destination, thereby making the recommendation process more interpretable and trustworthy.

4.4.3 Keyword Matching Results

In this section, we present the results of the keyword matching process, aimed at expanding our initial set of keywords by identifying and incorporating similar terms. Using the Word2Vec model, we matched each keyword from our core list with semantically related words, effectively enhancing the flexibility and coverage of the recommender system. This approach ensures that the recommender can recognize and connect broader user queries and descriptions to destinations more accurately.

Table 4.7 demonstrates this process by listing five example keywords from our expert-curated list along with five semantically similar words generated by the Word2Vec model.

Table 4.7 Main features and their matched features.

Main Feature	**Matched Features**
Sea	Ocean, seas, oceans, waters, coastal
Art	Printmaking, Art, arts, paintings, artist
Beach	Beaches, seashore, Beach, shoreline, seaside
Music	Jazz, Music, songs, musicians, tunes
Entertainment	Entertainments, music, amusements, gaming, leisure

This keyword expansion process significantly broadens the original list by incorporating synonyms and closely related terms. For instance, expanding "Sea" to include "ocean," "waters," and "coastal" enables the system to better accommodate variations in user terminology and regional language differences. Similarly, mapping "Art" to terms like "printmaking" and "paintings" allows the system to understand a interest of users in various art forms even if they do not specify "art" directly.

By setting the model parameter that controls the number of related keywords generated, experts can adjust the number of the keyword matching to meet different objectives. For instance, a broader selection may enhance recommendation flexibility, while a narrower set may yield more specific matches. This flexibility enables our recommender system to provide more accurate and contextually relevant destination suggestions, tailored to user preferences expressed in a variety of ways.

4.4.4 The Final Results of the Post Hoc Approach

Our post hoc explainable recommender system provides transparency into the destination recommendation process by highlighting key features associated with each recommended city. This approach is designed to recommend cities similar to a given main city and then justify the recommendations by extracting keywords and keyphrases that characterize these cities. We evaluated our approach for 200 of the most popular destinations offered by our airline partner, although only a subset of these results was presented in the thesis and related presentations due to space and time constraints.

Tables 4.8, 4.9, and 4.10 illustrate the top 3 recommended cities for three main cities—Zurich, Paris, and Tokyo—along with the top 10 features that make these cities similar.

Table 4.8 Zurich and recommended cities with features.

Main City	**Features**
Zurich	Nature, lake, hiking, museums, art, architecture, culture, food, shopping, festivals
Recommended Cities	**Features**
Geneva	Lake, nature, hiking, museums, art, culture, luxury, safety, transportation, climate
Vienna	Architecture, museums, art, culture, music, festivals, luxury, safety, transportation, food
Frankfurt	Skyscrapers, architecture, museums, culture, transportation, shopping, festivals, food, climate, safety

The system consistently identifies both relevant cities and relevant features across cities. For example, attributes of Zurich like nature, lake activities, and cultural richness align it with cities like Geneva and Vienna, while focus of Paris is on art, luxury, and cultural landmarks matches it with cities such as Lyon and Montréal. Meanwhile, modern amenities and outdoor activities of Tokyo are reflected in cities like Osaka and Seoul.

The results demonstrate the usefulness of the system in identifying relevant cities based on shared features, highlighting the key attributes that make one city similar to another. For instance, appeal of Zurich lies in the rich natural landscapes, such as lakes and hiking opportunities, alongside the cultural offerings like museums and art, which position it as similar to cities like Geneva and Vienna. These cities share a combination of natural beauty

Table 4.9 Paris and recommended cities with features.

Main City	**Features**
Paris	art, architecture, museum, culture, luxury, food, shopping, theater, festival, music
Recommended Cities	**Features**
Lyon	art, architecture, food, culture, museum, theater, festival, luxury, nature, skyscraper
Montréal	art, architecture, food, culture, museum, festival, theater, music, shopping, luxury
Brussels	art, architecture, food, culture, museum, theater, festival, luxury, shopping, geography

Table 4.10 Tokyo and recommended cities with features.

Main City	**Features**
Tokyo	Hotel, beach, hiking, subway, shopping, architecture, art, port, downtown, surf
Recommended Cities	**Features**
Osaka	Hotel, hiking, subway, shopping, sea, climate, art, downtown, island, surf
Seoul	Hotel, hiking, ski, shopping, sea, architecture, art, port, resort, downtown
Shanghai	Hotel, subway, shopping, sea, architecture, climate, art, resort, downtown, island

and cultural heritage, emphasizing features such as scenic views, outdoor activities, and a strong presence of art and history.

Similarly, Paris, renowned for its cultural landmarks, art, luxury, and culinary delights, finds strong parallels with cities like Lyon and Montréal. Both Lyon and Montréal, like Paris, offer a rich mix of art, architecture, and food, along with a reputation for luxury and cultural festivals, making them fitting recommendations for travelers seeking a similar experience. The system highlights the importance of specific attributes such as art, theater, and festival

scenes in drawing comparisons between these cities. Meanwhile, modern urban infrastructure of Tokyo, combined with outdoor activities like hiking and surfing, positions it as similar to Osaka and Seoul, two cities with a comparable blend of contemporary amenities and nature-based experiences. The shared features between these cities, such as transportation networks, shopping districts, and coastal attractions, underline the ability of the system to recommend destinations that are similar to each other,

4.5 Evaluation of Results by Experts

To evaluate the effectiveness of our explainable destination recommender system, we conduct weekly review sessions with experts from customer loyalty team at the airline company. These experts provide valuable insights by comparing the recommended cities and associated features with the recommendations produce by existing recommender system of the company, which only considers city similarity without explainability. Given their expertise in customer preferences and destination characteristics, the experts are able to assess not only the accuracy of the recommended cities but also the relevance and quality of the explainable features for each recommendation.

For a systematic evaluation of the recommended cities, we conduct a case study using the precision@k metric, a standard measure for evaluating recommender systems. Precision@k reflects the proportion of relevant items within the top-k recommendations, calculated as:

$$\text{Precision@k} = \frac{\text{number of recommended items @k that are relevant}}{\text{number of recommended items @k}}$$

In consultation with domain experts, we set $k = 10$ and considered a recommendation relevant if it aligns with the assessments of experts regarding suitable cities for each main city. We find that our content-based recommender system improved precision, raising it from 4/10 in the current system to 7/10, as confirmed by the evaluations of the experts.

Based on the rating of the experts, among the various models we employ, only the results generated by the `en_core_web_lg` and `BERT` models are accurate enough for use, whereas the results from `TF-IDF` are considered less accurate, aligning with our initial expectations. Therefore, our precision@k evaluation focuses exclusively on the comparison between the `en_core_web_lg` and `BERT` models. Tables 4.11 and 4.12 present the comparative evaluation results for two main cities, Paris and Tokyo, highlighting how our proposed recommendation cities align more closely with the judgments of the experts.

Through these evaluations, we observe that the content-based recommendations of our sys-

Table 4.11 Paris and recommended cities comparison with precision

Main City	**Recommended Cities and Precision@k**
Paris	['Boston', 'Rome', 'Vienna', 'New York', 'Newark', 'London', 'London', 'Toronto', 'Toronto', 'Seoul'] Precision@k: 4/10
Paris	['New York City', 'Lyon', 'New Orleans', 'Montréal', 'Las Vegas', 'Nice', 'Chicago', 'Algiers', 'London', 'Brussels'] Precision@k: 7/10
Paris	['Geneva', 'Lyon', 'Montréal', 'Rio de Janeiro', 'Barcelona', 'Mexico City', 'Brussels', 'Bogota', 'Santiago', 'Lisbon'] Precision@k: 7/10

tem generally aligned better with the assessments of the experts. For instance, for Paris, the proposed system recommended cities like Lyon and Montréal, which share cultural and lifestyle similarities with Paris, unlike the current recommendation system that included less relevant cities. Similarly, recommendations for Tokyo included culturally and geographically related cities like Seoul and Osaka, which better matched the views of the experts on destination similarity. This evaluation confirms the potential of our explainable recommender system to enhance user satisfaction by providing recommendations that are both accurate and interpretable, thereby improving the quality of destination recommendation.

In our case study, we dedicate a portion to validating the features extracted for each main city and the recommended cities. Experts in customer loyalty team at the airline company assess these features, determining their meaningfulness, accuracy, and practical usability. They found the extracted features beneficial not only for recommendation purposes but also for engaging with customers in targeted marketing campaigns. For instance, by focusing on specific features such as art, culture, and luxury, the experts believe these insights could help craft personalized emails suggesting destinations that resonate with travel histories of the customers. This approach allows for tailored recommendations that enhance customer engagement, leveraging known preferences and interests of travelers.

The alignment between the features identified for "Paris" and the corresponding features for recommended cities such as Lyon, Montréal, and Brussels (see Table 4.9) met expert expectations. For example, the attributes for Paris—art, architecture, museums, culture,

Table 4.12 Tokyo and recommended cities comparison with precision

Main City	**Recommended Cities and Precision@k**
Tokyo	['Tokyo', 'Osaka', 'Vienna', 'London', 'London', 'New York', 'Newark', 'Toronto', 'Toronto', 'Seoul'] Precision@k: 4/10
Tokyo	['Osaka', 'Honolulu', 'Beijing', 'New York City', 'Seoul', 'San Francisco', 'Portland', 'Shanghai', 'Washington,D.C.', 'London'] Precision@k: 7/10
Tokyo	['Seoul', 'Shanghai', 'Osaka', 'Reykjavík', 'Beijing', 'Vienna', 'New Delhi', 'London', 'Istanbul', 'Hong Kong'] Precision@k: 7/10

and luxury—are similarly prominent in Lyon, Montréal, and Brussels. Each of these cities shares characteristics like an emphasis on culture, food, and festivals, which are popular aspects of Paris. This alignment, seen in the close match of features across recommended cities, reveal the ability of the proposed recommender system to capture the essence of each destination accurately.

4.6 Limitations

The proposed explainable destination recommender system faces some limitations. The reliance of the system on a predefined list of keywords can sometimes oversimplify city descriptions. Cities are complex and multi-faceted, and a limited set of keywords may not fully capture their unique qualities. For instance, a city known for both modern and historical architecture might lose this depth if it is grouped with cities sharing only one of these features. This limitation may impact the relevance of recommendations for users with diverse interests. Additionally, the performance of the system depends on the quality of the data sources used. If certain cities are better documented in these sources, the recommendations might favor these well-documented locations, limiting variety. Expert evaluations, while insightful, are subjective and may not perfectly match the preferences of every traveler. Moreover, the evaluation relies on Precision@k, which assesses relevance but not diversity or novelty—important factors in creating engaging recommendations. A more balanced approach with metrics like recall or user satisfaction would give a fuller picture. Finally, the

lack of a re-ranking step limits the ability of the system to prioritize the most relevant cities and features. Adding a re-ranking component could enhance the accuracy by refining the results based on user preferences, improving the overall personalization and relevance of the recommendations. Another limitation lies in the lack of LLMs in this experiment. Integrating LLMs could enhance the quality of recommendations by enabling deeper contextual understanding and better capturing nuanced aspects of cities.

4.7 Conclusion

In conclusion, this chapter presents an explainable destination recommender system aimed at providing users with personalized and meaningful destination suggestions based on feature extraction. By identifying key features that define each city, the system delivers recommendations aligned with the preferences of users, with a focus on transparency so users can understand the rationale behind each suggestion. Expert evaluations highlight the effectiveness of the system in offering relevant travel recommendations, though certain limitations remain, such as the risk of oversimplifying complex city characteristics and relying heavily on specific data sources, which may impact the diversity and accuracy of recommendations.

In the following chapters, we explore advanced information retrieval techniques, particularly ranking models, to retrieve highly relevant data from large corpora. These methods can enhance recommender systems by enabling models that incorporate nuanced ranking strategies. Additionally, we propose prompt engineering techniques designed to directly optimize recommendations. These prompting techniques leverage the power of LLMs to improve recommendation precision and adapt to diverse user contexts, potentially overcoming some of the limitations observed in this chapter.

CHAPTER 5 RE-RANKING STEP BY STEP: INVESTIGATING PRE-FILTERING FOR RE-RANKING WITH LARGE LANGUAGE MODELS

5.1 Pre-Filtering Step in Information Retrieval Pipeline

LLMs have revolutionized natural language processing (NLP) tasks with their zero-shot capabilities, achieving significant advances across a range of applications, including IR and passage ranking [100, 159]. These models, such as GPT-4 and Mixtral, are pretrained on extensive and diverse text data, their responses are increasingly human-like and closely aligned with human intentions [1]. Consequently, they are increasingly integrated into IR systems to enhance query-passage matching and ranking tasks.

A typical information retrieval system comprises multiple components organized into a processing pipeline. This pipeline features two primary stages: the retriever and the reranker [160]. While the retriever selects the most relevant passages from a large-scale corpus, the re-ranker focuses on re-ordering (i.e., re-ranking) the candidate passages, using their relevance. Although this two-stage architecture is widely used, multi-stage or cascade ranking approaches have a longer history in retrieval system design [161]. Based on these architectures, each component can thus be optimized for its given task.

The advent of LLMs has brought notable advancements to the IR pipeline. Research in this field, if we broadly include transformer-based models like BERT, has primarily concentrated on applying such models as dense or sparse retrievers in the first stage of the pipeline [1]. However, if we limit the term LLM to refer specifically to highly parameterized, decoder-only or encoder-decoder transformers, there has actually been more initial focus on employing them in the re-ranking stage—especially in zero-shot settings—than in retrieval. Recent studies on zero-shot re-ranking with LLMs, for example, have shown substantial performance improvements [10]. Existing approaches for zero-shot re-ranking with LLMs include Pointwise [13, 15], Pairwise [12], Setwise [11], and Listwise [10, 14] methods. Each approach uses specific prompting techniques to generate relevance estimates, improving the quality of re-ranking.

Despite these advancements, one challenge remains: even with advanced models, some passages retrieved by initial retrievers, such as BM25 [24], may still be irrelevant to the query. This results in inefficiencies and limits ranking performance.

In this chapter, we propose a novel LLM-based pointwise pre-filtering method that filters out

irrelevant passages before they are given to the re-ranker. We design a pointwise prompting strategy which instructs the open-source LLM to generate a relevance score for each passage based on its relevance to the given query in the range of 0 to 1. Then, using a sample of generated scores, we establish a specific threshold for passage filtering. Using this threshold we can then pre-filter any new passage. Passages that exceed this threshold are retained as relevant passages and forwarded to the re-ranker, while those falling below the threshold are discarded. By implementing this straightforward process, only relevant passages are passed to the ranker, reducing the overall number of passages in the ranker. To investigate the usefulness of our proposed approach we focus on the following two research questions:

> **RQ1:** Can existing human-labeled relevance scores (i.e., qrels) be used to help LLMs filter out irrelevant passages?
>
> **RQ2:** Does filtering out irrelevant passages before re-ranking improve the results of an LLM re-ranker?

To answer **RQ1** we investigate prompting Mixtral-8x7B-Instruct (with 4-bit quantization) to assign a quantitative value to the relevance of passages retrieved by BM25. We then leverage human-labeled relevance scores (i.e., qrels in the TREC and BEIR datasets), to determine a relevance value below which passages should be deemed irrelevant (i.e., a relevance threshold). We evaluate our approach on three datasets (TREC-DL2019, TREC-DL2020, and four BEIR tasks) and find that it is generally possible to find a threshold value–using F1 score–that maximizes the relevance of the retrieved passages. We also find that this threshold value (0.3) appears mostly stable across all of the tested datasets.

To answer **RQ2**, we investigate the use of a pre-filtering step–to filter out irrelevant passages– before re-ranking passages with Mixtral-8x7B-Instruct (loaded with 4-bit quantization). Again, we evaluate our approach on three datasets (TREC-DL2019, TREC-DL2020, and four BEIR tasks) and find that the use of a pre-filtering step significantly improves the resulting re-ranking of passages. Indeed, after using our pre-filtering step, a limited model such as Mixtral-8x7B-Instruct (loaded with 4-bit quantization) can become competitive with– and in one case surpass–much larger, and resource intensive, models such as GPT-4.

5.2 LLM-based Pre-Filtering

In this section, we propose a new pointwise LLM-based pre-filtering step to score passages based on their relevance to the query and filter out irrelevant passages before any re-ranking is conducted. Below, we explain the significance of this step, describe the prompting strategy

used for generating relevance scores in the pointwise manner, outline the process of analyzing the generated scores, and explain how we set the thresholds.

Our proposed pre-filtering step is straightforward yet efficient. The main goal of this step is to filter irrelevant passages before passing them to a re-ranker, thus decreasing the total number of passages that require re-ranking. After the initial retrieval stage (e.g., based on BM25 [24]) retrieves a set of passages for a given query, each passage is evaluated by an LLM-based filter to determine its relevance to the query. This pointwise filter, using the language understanding capabilities of LLMs, assigns a relevance score to each passage (e.g., from 0 to 1 where 0 denotes a completely irrelevant passage, and 1 denotes a fully relevant one). Then, a threshold is set based on both the qrels and these scores. Passages with scores at or above the threshold are passed to the re-ranker, while those with scores below the threshold are discarded. Using the pre-filtering step, the number of irrelevant passages that can misguide the re-ranker decreases, leading to improved performance for the re-ranker. Figure 5.1 illustrates the role of the pre-filtering step in the information retrieval pipeline.

5.2.1 Pointwise Prompt Design for Score Generation

To design our pointwise prompt, we use two well-known prompting methods:

Chain-of-Thought (CoT) [110,116]: This method allows LLMs to produce intermediate reasoning steps explicitly before generating the final answer.

Plan-and-Solve (PS) [17]: This method consists of designing a plan to divide a task into smaller subtasks and then carrying out the subtasks according to the plan.

Our proposed zero-shot prompt is a combination of both of these reasoning methods:

> I will provide you with num passages along with a query. Each passage has an alphabet label next to it. Generate the relevance score for the passages concerning the query. The relevance score is a number from 0–1 based on how relevant you think the passage is to the query. Grasp and understand both the query and the passages before score generation. Then, based on your understanding and analysis quantify the relevance between the passage and the query. Give the rationale before answering.
>
> **Passage:** {passage}
> **Query:** {query}

In the first part of the prompt, we devise a plan for the LLM to understand both the query and the passage, and then, based on its analysis, generate a relevance score. In the second part,

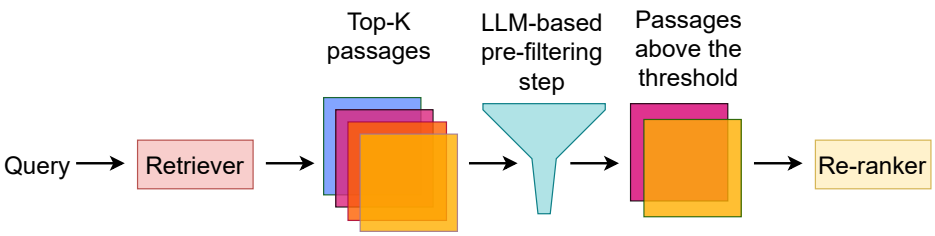

Figure 5.1 The role of the pre-filtering step in the information retrieval pipeline.

we include a sentence asking the LLM to explain the rationale behind the score generation. By designing this plan and instructing the LLM to provide a rationale before responding, its understanding of the query and the passage is incrementally enhanced. Our manual analysis of multiple different prompts led us to believe that this step-by-step approach results in the generation of more accurate relevance scores.

5.2.2 Analyzing Relevance Scores

To give context to the relevancy scores generated by LLMs for the passages, they should be compared with an existing baseline. In our experiments, we make use of the relevancy levels in the query relevance judgments (qrels) file for each passage. These relevancy levels have somewhat similar interpretations across different datasets; however, details can be different. In all cases, higher scores indicate greater relevance between the query and the passage. However, the interpretation of a **level** [**1**] score differs: in some datasets, a score of [**1**] is considered relevant; in others, irrelevant; and in some others, it is interpreted as partially relevant. We categorize the datasets based on the the interpretation of a **level** [**1**] score as follows:

- Datasets where a relevancy level of [**1**] is interpreted as either relevant (e.g., Touche [23]) or partially relevant (e.g., Covid [23]), a relevancy level of [**1**] or higher is considered relevant.
- Datasets where a relevancy level of [**1**] is interpreted as not relevant (e.g., TREC-DL19, TREC-DL20 [21,22]), a relevancy level of [**1**] or lower is considered irrelevant. In these datasets the nDCG scores use a gain value of 1 for related passages.

5.2.3 Setting a Relevance Threshold

Since the scores generated by LLMs are decimal numbers between 0 and 1, we convert these scores to be comparable with the integers that represent relevancy levels. Thus, we set a threshold to replace the decimal scores with the following values:

$$S_{\text{pre}} = \begin{cases} 1, & \text{if the score at the threshold} \\ 1, & \text{if the score above the threshold} \\ 0, & \text{if the score below the threshold} \end{cases} \tag{5.1}$$

Where passages with a relevancy score at or above this threshold are considered relevant and assigned a score of 1, while those below the threshold are given a score of 0. Next, the four elements of the confusion matrix—true negatives, true positives, false positives, and false negatives—are calculated. This is done by comparing the S_{pre} for each passage with the relevancy levels in the qrels file. Since not all passages in each dataset have relevancy levels assigned, we only use the passages with relevancy levels in the qrels file to compute these four elements. After the threshold value is selected, even passages without relevancy levels can then be considered, as the LLM can still generate scores for them, and their scores can thus be replaced based on the previously defined threshold. Based on these values, Precision, Recall, and F1 Score are computed for each threshold and the threshold with the highest F1 Score is selected.

In the context of IR systems, these elements are defined as follows:

True Negative (TN): Both the qrels files and the LLM classify the passage as irrelevant.

True Positive (TP): Both the qrels files and the LLM classify the passage as relevant.

False Positive (FP): The qrels files classify the passage as irrelevant, but the LLM classifies it as relevant.

False Negative (FN): The qrels files classify the passage as relevant, but the LLM classifies it as irrelevant.

We initially select the threshold value randomly within the predefined range of 0 to 1. Our analysis indicates that, when evaluating thresholds beyond the initial value, it is sufficient to compare the F1 scores obtained for slightly higher and slightly lower thresholds. This approach effectively captures the trend across other potential threshold values, enabling the identification of the optimal threshold.

5.3 The Advantages of Pre-Filtering

In IR systems, the primary objective is to retrieve information that fully or partially matches queries of the users. While identifying both relevant and irrelevant passages is important, minimizing false negatives (FNs)—unretrieved relevant documents—is particularly critical, as these cannot be recovered later in the pipeline. To address this, our approach focuses on optimizing the F1 score by adjusting the retrieval threshold to increase true positives (TPs) and true negatives (TNs) while reducing FNs. Although reducing false positives (FPs), or non-relevant documents, is also a goal of our approach, they are less problematic, as they can be deprioritized by the re-ranker later in the pipeline.

By removing irrelevant passages, the total number of initial passages sent to the final re-ranker, and consequently the number of calls to LLM during the re-ranking phase, will decrease. This enhancement will improve the accuracy of the re-ranking step.

Figure 5.2 illustrates the effect of the threshold on retaining or discarding passages.

We test our proposed approach using an open-source LLM (i.e., Mixtral) which is easily accessible for both academic research and industry applications. Since this is an open-source model, there is no need for commercial LLM APIs, which can be expensive and may not satisfy some data privacy concerns. Furthermore, our experiments show that our approach allows smaller, more limited models, to remain competitive with much more demanding models. This can allow resource-constrained situations to still make use of state-of-the-art re-ranking. Our pre-filtering step is designed based on Zero-Shot prompting and thus eliminates the need to retrain or fine-tune the LLM. The complexity of our method is linear, O(n), and by discarding irrelevant passages, it reduces the number of LLM inferences required in the final re-ranking step. Since access to model logits is typically restricted with closed-source LLMs, we focused on leveraging the generation capabilities of the models instead. This approach allows us to work flexibly with both open- and closed-source LLMs and also aligns with our goal of enhancing model interpretability and performance. Table 5.1 presents properties of different re-ranking methods with LLMs.

5.4 Passage Re-Ranking with LLMs

After filtering out irrelevant passages, i.e., passages below the threshold, any LLM re-ranking method—Listwise, Pairwise, and Setwise—is applicable. For our experiments, we use Listwise prompting [10], which employs an instructional permutation generation method combined with a sliding window strategy to directly output a ranked list of candidate passages. In Listwise prompting, the LLMs receive a prompt with a given query, a list of candidate

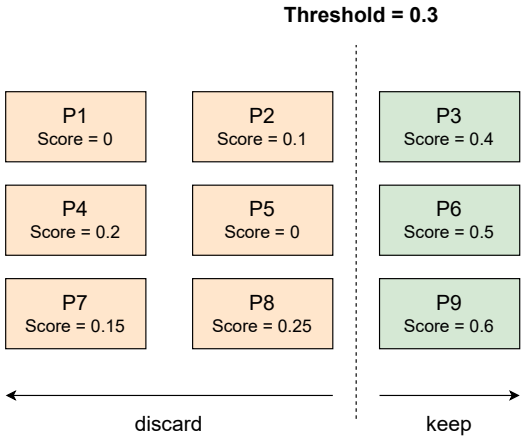

Figure 5.2 The effect of the threshold on the number of passages.

passages, and an instruction to generate a ranked list of passage labels based on their relevance to the query.

Due to the input length limitations of LLMs, it is not possible to include all candidate passages in a single prompt. To handle this issue, this approach uses a sliding window method which involves ranking a window of candidate passages, starting from the bottom of the original ranking list and moving upwards. This process can be repeated multiple times to achieve an improved final ranking. We select this approach as the final re-ranking step for two reasons; (1) this approach strikes a middle ground between efficiency and effectiveness, and (2) this method enables the use of early stopping mechanisms that focus specifically on the top-K rankings, enhancing efficiency.

5.5 Experimental Results of LLMs

5.5.1 Datasets

Consistent with previous related research [10–12], our experiments are conducted on two well-established benchmark datasets in information retrieval research. These benchmark datasets include TREC-DL [21, 22] and BEIR [23].

TREC

TREC is a widely used benchmark dataset in information retrieval studies. To allow comparison with prior work, we use the test sets of the 2019 and 2020 competitions: TREC-DL2019

Table 5.1 Properties of different re-ranking methods with LLMs. #LLM calls: the number of LLM API Calls in the worst case. Logits: access to the LLM's logits is required. Batching: batch inference is allowed. Generate: Token generation is required. N: the number of passages to re-rank. K: the number of top-k relevant passages to find. c: the number of compared passages at each step. N': the number of filtered passages, which is often much smaller than the initial N, and in the worst case, it is equal to N.

Methods	#LLMcalls	Logits	Batching	Generate
Pointwise	$O(N)$	×	×	
Listwise	$O(K*N)$			×
Pairwise(all pairs)	$O(N^2)$	×	×	×
Pairwise(heap sort)	$O(K*\log_2 N)$	×		×
Pairwise(bubble sort)	$O(K*N)$	×		×
Setwise(heap sort)	$O(K*\log_c N)$	×		×
Setwise(bubble sort)	$O\left(K*\left(\frac{N}{c-1}\right)\right)$	×		×
Pre-Filtering	$O(N)+O(K*N')$		×	×

and TREC-DL2020. Both datasets are human-labeled and contain 43 and 54 queries, respectively. Each dataset is derived from the MS MARCO v1 passage corpus, which contains 8.8 million passages, with more comprehensive labeling. Based on the interpretation of the relevancy scores in the qrels files of these datasets, while passages with the relevancy level of [**1**] are considered irrelevant, they contribute positively to the nDCG score. Therefore, for these datasets, we conduct our experiment in two different scenarios with two different thresholds: (i) Considering passages with a relevancy level of [**1**] as relevant. (ii) Considering passages with a relevancy level of [**1**] as irrelevant.

BEIR

BEIR consists of diverse retrieval tasks and domains. Due to limited resources, we could not run our experiments on all of the BEIR tasks. Therefore, we choose to concentrate on four tasks in BEIR to evaluate the models:(i) Covid retrieves scientific articles addressing queries related to COVID-19. (ii) Touche is a dataset that focuses on argument retrieval for controversial questions. (iii) Signal is a data collection of retrieved tweets for news articles. (iv) News is a dataset that focuses on relevant news articles based on news headlines. In all of these four datasets, passages with the relevancy level of [**1**] are interpreted as relevant or partially relevant. Thus, we run our experiment only in one scenario with one threshold for these datasets: Considering passages with a relevancy level of [**1**] as relevant.

5.5.2 Implementation and Metrics

To enable fair and direct comparison with prior works, our experiments are conducted using the top 100 passages retrieved by BM25, serving as the first-stage retriever through Pyserini[1] with its default settings. We evaluate the effectiveness of approaches using the NDCG@10 metric, which is the official evaluation metric for the datasets used. As one of the main goals of this chapter is to investigate the effects of open-source LLMs on re-ranking tasks, we employ the open-source language model Mixtral-8x7B-Instruct-v0.1 [100], which has 46.7 billion parameters, for both the LLM-based pre-filtering and re-ranking steps. Due to the input length limitations of LLMs, we divide the initial list of passages into smaller chunks for the pre-filtering step, each containing 5 elements, before processing them with the LLM. We use 5 elements as it provides a balance between computational efficiency and effectiveness. For the re-ranking step, we implement the sliding window strategy introduced by Sun et al. [10], with a window size of 10 and a step size of 5.

We carry out our experiment on a Google Cloud a2-highgpu-1g machine equipped with a single NVIDIA A100 40GB GPU with 40 GB of memory, and 12 vCPUs. Due to resource constraints, the LLM is loaded with 4-bit quantization. Using these resources, we conducted our experiments separately for each group of datasets using the methodology presented in Section 5.2.

5.5.3 Setting the Threshold Values

As discussed in Section 5.2.3, our approach depends on threshold values. For all four tasks in the BEIR benchmark, we set a single threshold by considering passages with a relevancy level of [**1**] as relevant. Our analysis determines that **0.3** is the optimal threshold for BEIR, as it yields the highest F1 score compared to other values.

For both datasets in the TREC benchmark, we set two thresholds: one by considering passages with a relevancy level of [**1**] as relevant, and another by considering passages with a relevancy level of [**1**] as irrelevant. Here, we find that a threshold of **0.3** is once again optimal, similar to its performance on the BEIR dataset, as it achieves the highest F1 score compared to other values.

Our evaluation of these threshold values answers our first research question (**RQ1**). We find that qrels can be used to determine a threshold value that can effectively help our chosen LLM filter out irrelevant passages. Also, we find that the threshold 0.3 is stable for our LLM across all datasets. This implies that future datasets need not necessarily have expert judgement (or

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

qrels) to use our approach in a useful fashion. The use of a previously determined threshold may be sufficient to obtain decent pre-filtering power.

5.5.4 Minimum Number of Required Qrels

To establish a robust threshold for filtering irrelevant passages, we identify a stable value of 0.3 across all datasets, yielding comparable performance to prior benchmarks. However, our approach relies on the complete qrels file for each dataset, which may not be feasible in real-world scenarios due to the scarcity of labeled data. Therefore, we conducted an empirical study on the DL19 dataset to determine the minimum number of qrels required to achieve competitive results, particularly focusing on F1 scores and NDCG@10.

The DL19 dataset contains 43 queries, and we retrieve 100 passages per query using BM25, resulting in 4300 passages. Among these, only 2257 passages have relevance labels in the qrels file. We thus discard the remaining 2043 passages. We then partitioned the labeled dataset into two subsets: 80% for training to determine the optimal threshold and 20% for testing its effectiveness.

To simulate scenarios with limited labeled data, we experimented with various sample sizes starting at 20 and incrementally increasing to the full dataset size. For each sample size presented in Table 5.2, we used random sampling without replacement, repeated the experiments 10 times, and averaged the results to ensure consistency. The goal is to evaluate the effect of sample size on the F1 score and NDCG@10. The results are summarized in Table 5.2:

Table 5.2 Empirical study results on DL19 to find the minimum number of required labeled samples.

Sample Size	**Threshold**	**F1 Score**	**Precision**	**Recall**	**NDCG@10**
20	0.182	0.921	0.908	0.939	45.28
40	0.308	0.881	0.880	0.885	44.44
80	0.308	0.860	0.850	0.873	44.44
100	0.250	0.872	0.849	0.897	44.54
Half	0.220	0.861	0.829	0.896	44.54
All	0.200	0.861	0.827	0.897	45.28

The results indicate the following key findings:

- Stability of NDCG@10: The NDCG@10 values remain mostly consistent across all sample sizes, with negligible variations. This demonstrates that even a small subset of qrels is sufficient to identify a robust threshold for filtering irrelevant passages effectively.

- Small Sample Effectiveness: As few as 20 labeled samples achieve high F1 scores (0.921), indicating that a minimal amount of labeled data can yield reliable thresholds for filtering.

While using the entire qrels file ensures optimal performance, our findings suggest that a small percentage of labeled data is enough for determining effective thresholds. This significantly reduces the need for annotations in real-world applications while maintaining performance.

5.5.5 Results on Benchmarks

To situate our results, we compare our approach with state-of-the-art supervised and unsupervised passage re-ranking techniques.

The supervised baselines are as follows: monoBERT [162]: A BERT-large based cross-encoder re-ranker, trained using the MS MARCO dataset. monoT5 [163]: A sequence-to-sequence re-ranker that uses T5 to calculate the relevance scores using pointwise ranking loss. RankT5 [164]: A re-ranker that employs T5 and uses listwise ranking loss. Cohere Rerank[2]: A passage reranking API named rerank-english-v2.0, developed by Cohere[3], which does not explain the architecture or training method of the model. The unsupervised LLM-based baselines include: Unsupervied Passage Re-ranker (UPR) [13]: The pointwise approach with instructional query generation. Relevance Generation (RG) [15]: The pointwise approach that generates relevance judgments for a given query and candidate items. RankGPT [10]: The listwise approach generates a ranked list of passage labels based on their relevance to the query. We also compare our results to Mixtral-8x7B-Instruct without our pre-filtering step to show the improvement obtained through our pre-filtering.
Table 5.3 presents the evaluation results obtained from the TREC and BEIR datasets. Results show that: (i): The pre-filtering method can achieve the best results on the Signal and Touche datasets for NDCG@10, even outperforming commercial solutions (e.g., GPT-4). (ii): The pre-filtering method outperforms all other methods, other than BM25, for the Touche dataset. (iii): Pre-filtering achieves an average improvement of 7.2% in nDCG@10 on all datasets compared to our baseline without pre-filtering. These results answer our **RQ2**, and show that pre-filtering irrelevant passages before re-ranking improves its overall results.

[2]https://txt.cohere.com/rerank/
[3]https://cohere.com/rerank

Table 5.3 Results (nDCG@10) on TREC and BEIR datasets by re-ranking top 100 documents retrieved by BM25.

Methods	Threshold	DL19	DL20	Covid	Touche	Signal	News
BM25	NA	50.58	47.96	59.47	44.22	33.05	39.52
Supervised							
monoBERT (340M)	NA	70.50	67.28	70.01	31.75	31.44	44.62
monoT5 (220M)	NA	71.48	66.99	78.34	30.82	31.67	46.83
monoT5 (3B)	NA	71.83	68.89	80.71	32.41	32.55	48.49
RankT5 (3B)	NA	72.95	69.63	82.00	37.62	31.80	48.15
Cohere Rerank-v2	NA	73.22	67.08	81.81	32.51	29.60	47.59
Unsupervised LLM-based							
UPR (FLAN-T5-XXL)	NA	62.00	60.34	72.64	21.56	30.81	42.99
RG (FLAN-UL2)	NA	64.61	65.39	70.22	24.67	29.68	43.78
RankGPT (gpt-3.5-turbo)	NA	65.80	62.91	76.67	36.18	32.12	48.85
RankGPT (gpt-4)	NA	75.59	70.56	85.51	38.57	34.40	52.89
Pre-Filtering Step							
PF (Mixtral-8x7B-Instruct)	0.3	69.39	64.42	81.64	**43.94**	**37.09**	51.20
Baseline Without Pre-Filtering Step							
Mixtral-8x7B-Instruct	NA	60.88	55.85	66.48	43.1	35.68	47.16

5.6 Limitations

The limitations of this work include the threshold-setting process, where the initial threshold value is selected randomly. Based on this value, the F1 score is computed to find the optimized threshold. Although the threshold introduced in this chapter is derived from analysis and the results reveal their effectiveness, these thresholds are dependent on the dataset and might differ for different datasets. Additionally, our approach depends heavily on the qrels files and the interpretation of relevance levels for the dataset. We conduct an empirical study to determine the minimum number of labeled passages required, specifically on the DL19 dataset. However, the generalizability of these findings needs to be further investigated. As the pre-filtering step is an intermediate step in the information retrieval pipeline, its effectiveness is closely related to other elements in the pipeline, such as the retriever and re-ranker. Due to hardware constraints, particularly GPUs, we only ran our experiments on four datasets (Covid, News, Signal, Touche) instead of the eight datasets used in BEIR, which are referenced in similar works. We also tested our experiments on other open-source models, such as Llama-2-13b, but while the approach did provide improvements over non-prefiltered results, the results were not comparable to the current state-of-the-art.

5.7 Conclusion

In this chapter, we conduct a study on the use of a pointwise pre-filtering step before passage re-ranking with LLMs. We introduce a novel approach to further exploit the power of LLMs in IR passage ranking. Our experiments on three benchmarks (TREC-DL2019, TREC-DL2020, and four BEIR tasks) show that using our approach, smaller LLMs (i.e., Mixtral-8x7B-Instruct with 4-bit quantization), can be made competitive with much larger models. While our approach does require some expert input, we also show that the amount of input needed can be small, and furthermore, that the thresholds used appear to be effective across multiple benchmarks.

CHAPTER 6 ARTICLE 1 AN EMPIRICAL STUDY OF SELF-CONSISTENCY AND CONFIDENCE ELICITATION IN LLM-BASED POINTWISE RE-RANKING Baharan Nouriinanloo , Maxime Lamothe Submitted to ACL ARR 2024 on October 16 2024

This chapter was submitted for review to NAACL 2025 as an article bearing the title of "[An Empirical Study of Self-Consistency and Confidence Elicitation in LLM-Based Pointwise Re-Ranking]". *For this article, the main author designed and implemented the approaches, conducted experiments, analyzed results to identify limitations and propose improvements, performed literature reviews to contextualize the research, and authored key sections on methodology, results, and contributions.*

6.1 LLM-based Poinwise Re-Ranking Problems

LLMs such as GPT-4 [159] and Mixtral [100] have shown significant capabilities across a range of NLP tasks, including zero-shot document re-ranking in IR systems. Recent studies have demonstrated that LLMs can achieve impressive results in zero-shot settings for document ranking [9, 10, 12, 14, 15]. However, despite the advances in using LLMs for re-ranking, pointwise zero-shot approaches often underperform compared to other ranking methods, partly due to their inherent limitations in generating accurate relevance scores.

Methods for zero-shot document ranking with LLMs can generally be classified into four primary approaches: Pointwise [9, 13, 15], Listwise [10, 14], Pairwise [12], and Setwise [11]. Each of these approaches leverages different prompting strategies to help the LLM estimate relevance scores for candidate documents.

The initial efforts in zero-shot ranking with LLMs primarily adopted a pointwise approach, in which each document is scored independently in relation to a query, and documents are ranked according to their individual scores [13, 15]. However, this method often yields lower retrieval performance, as the generated relevance scores are sometimes inaccurate, leading to suboptimal rankings [11, 165].

While recent advancements in self-consistency [18] and confidence elicitation [19] in LLMs have demonstrated significant improvements in model reliability and decision-making, these techniques have yet to be fully explored in the context of pointwise re-ranking. Self-consistency, as proposed in [18], enhances answer quality by sampling diverse reasoning paths and selecting the most consistent output, rather than relying on a single, potentially biased answer.

Confidence elicitation, as shown in [19], allows LLMs to express the confidence level of their responses, which has been shown to improve the reliability of decision-making in other applications.

This thesis aims to empirically investigate the effect of incorporating self-consistency and confidence elicitation into pointwise zero-shot document re-ranking with LLMs. Specifically, we explore whether these techniques can enhance the accuracy and reliability of pointwise re-ranking approaches by refining the generated relevance scores and leveraging the confidence levels of LLMs.

In our experimental setup, we first prompt the LLM to generate a relevance label and corresponding confidence score for each document in relation to the query. We apply temperature sampling to introduce diversity in the generated outputs by running multiple iterations at different temperature settings. The study addresses two main research questions:

> **RQ1:** What is the impact of confidence elicitation on pointwise zero-shot LLM-based document rankers?
>
> **RQ2:** What is the impact of self-consistency when considering confidence on pointwise zero-shot LLM-based document rankers?

To answer the first question, we run our prompt across six different temperatures ([0, 0.2, 0.4, 0.6, 0.7, 0.8]) to investigate the role of confidence elicitation on generated relevance labels in a pointwise approach under varying temperature settings. For the second question, we generate multiple answers for each passage by running the LLM under two different configurations: (i): using five different temperatures ([0, 0.2, 0.4, 0.6, 0.8]), and (ii): running the LLM five times at a constant temperature of 0.7. As we generate multiple answers for each document, we employ three different aggregation methods to compute the final relevance score: (1) Averaging Approach, (2) Weighted Average Approach, and (3) Majority Vote Approach.

Figure 6.1 illustrates a sample of generated relevance labels and confidence scores for a document concerning the query in our method.

Our experiments use three re-ranking datasets: TREC-DL2019, TREC-DL2020 [21, 22], and the Touche subset of the BEIR benchmark [23], leveraging the Mixtral-8x7B-Instruct model [100] with 4-bit quantization for computational efficiency.

6.2 Study Design

In this section, we present the details necessary to replicate our study. We concentrate on the datasets, prompts, and general experimental setup that we used to answer both of our

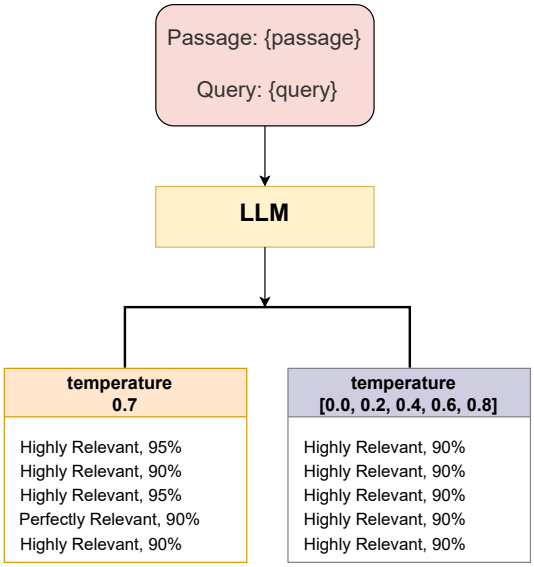

Figure 6.1 A sample of generated scores for a document concerning the query.

research questions.

6.2.1 Datasets

To be consistent with previous research in pointwise zero-shot document re-rankers that use LLMs [9, 13, 15], our experiments are conducted using two widely recognized benchmark datasets in the field of information retrieval. These benchmark datasets include TREC-DL [21, 22] and BEIR [23].

TREC: TREC is a widely used benchmark in information retrieval studies [13, 15, 132]. To allow comparison with prior work [13, 15], we use the test sets of the 2019 and 2020 competitions: TREC-DL2019 and TREC-DL2020. Both datasets are human-labeled and contain 43 and 54 queries, respectively. These datasets are constructed using the MS MARCO v1 passage corpus, which consists of 8.8 million passages and includes dense human relevance annotations for each of the 43 and 54 queries.

BEIR: The BEIR dataset consists of diverse retrieval tasks and domains. While we aim to compare our results to prior work through BEIR, due to limited resources, we focus our experiments on a single task within the BEIR benchmark: the Touche dataset. We select Touche for two primary reasons: (1) Touche has the least queries in the BEIR dataset,

thus reducing the compute needed; (2) Touche appears to be the only dataset in BEIR for which LLM have difficulty improving over the original BM25 ranking [12], thus presenting a challenge. Touche is a dataset with 49 queries and focuses on argument retrieval for controversial questions. Touche is constructed from the args.me corpus, which consists of 387k arguments.

6.2.2 Prompt Design

The quality of LLM answers is affected by the prompt used to elicit the answer [165]. In this chapter, we aim to empirically study the effect of confidence elicitation and self-consistency on pointwise zero-shot document re-rankers. Because we elicit the same question with different settings (e.g., temperature changes), changes in prompts could present a confounding factor for our results. Therefore, we aim to isolate the impact that different prompts might have on our study. Thus, to answer our research questions, we designed a single prompt based on insights from recent works [9, 19, 165], aiming to optimize the effectiveness of generated scores in pointwise re-ranking.

Our confidence elicitation instruction prompts the LLM to select a relevance label from the four available relevance labels and to provide its confidence level.

> I will provide you with a passage along with a query. Please classify the passage concerning its relevance to the query in one of these categories “Perfectly Relevant”, “Highly Relevant”, “Somewhat Relevant”, or “Not Relevant”. Provide your answer and your confidence in the following format:
> “Answer and Confidence (0-100): [ONLY the category; not a complete sentence], [Your confidence level, please only include the numerical number in the range of 0-100]%”
> Note: The confidence indicates how likely you think your answer is true.
>
> **Passage:** {passage}
> **Query:** {query}

As highlighted by [9], some passages may not directly answer a query but can still contain useful information. Therefore, classifying passages using only binary categories like "Yes" or "No" may be insufficient. To address this limitation, they incorporate fine-grained relevance labels into the prompt, enabling the LLM ranker to better distinguish between documents with varying levels of relevance to the query. Building on their approach, we designed our prompt to classify passages into four categories: "Perfectly Relevant," "Highly Relevant,"

"Somewhat Relevant," and "Not Relevant." According to their findings, this more nuanced approach allows LLMs to distinguish between partially relevant documents and those that are either fully relevant or irrelevant.

According to [165], while no single prompt wording guarantees optimal effectiveness, certain factors can influence performance. These factors include tone words, evidence ordering, and the position of evidence within the prompt. While the goal of this study is to empirically study the effect of confidence elicitation and self-consistence, and not to produce the best possible prompt for zero-shot document re-ranking, we still aim to leverage state-of-the-art techniques. Thus, we account for these factors in our prompt design.

Additionally, [19] explore prompting strategies to elicit verbalized confidence in LLM responses. Following their proposed prompt structure, we explicitly instruct the LLM to provide both its answer and a confidence score. This is achieved by adding the instruction:

"Provide your answer and your confidence," followed by: "Note: The confidence indicates how likely you think your answer is true."

This ensures that the LLM not only generates a relevance score for a passage concerning the query but also expresses its confidence in its response.

6.2.3 Implementation and Metrics

First-stage Retriever Our experiments concentrate on the use of LLM as re-rankers. Therefore, our experiments first rely on a first-stage retriever to extract potentially relevant passages, which then require re-ranking. Similarly to prior works [13, 15], we use BM25 [24], as our first-stage retriever through Pyserini[1] with its default settings.

Choice of LLM In our experiments, we employ the open-source language model Mixtral-8x7B-Instruct-v0.1 [100], which has 46.7 billion parameters with 4-bit quantization. While our experiments would benefit from being run on multiple models, resource constraints limit us to a single small model. We chose Mixtral-8x7B-Instruct-v0.1 with 4bit quantization because it is a 'recent' general-purpose model, can run on our current hardware, and allows us to study the effect of temperature. Furthermore, as a smaller open-weight model, our results can also showcase how competitive these models can be.

Relevance Score Generation Given a query and relevant passages obtained through our first-stage retriever, we prompt the LLM to provide a relevance label alongside a confidence

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

score. These labels are categorized as: *"Perfectly Relevant," "Highly Relevant," "Somewhat Relevant,"* and *"Not Relevant."* To facilitate further analysis, we map these labels to numeric values: 3 for *"Perfectly Relevant,"* 2 for *"Highly Relevant,"* 1 for *"Somewhat Relevant,"* and 0 for *"Not Relevant"*. We then re-rank the initial list based on the results produced given the requirements of each research question.

Evaluation Metrics Based on prior works, we re-rank the top 100 passages retrieved to evaluate our experiments [13,15]. We then evaluate the effectiveness of our experiments using the NDCG@10 metric, the official evaluation metric for TREC and BEIR.

Comparison Baselines To compare our results with previous findings, the following baselines are considered:

Unsupervised passage Re-ranker (UPR) [13]: A pointwise approach based on query generation, see Section 3 for more details.

Relevance Generation (RG) [15]: A pointwise approach dependent on relevance generation, see Section 3 for more details.

Rating Scale 0-to-k Relevance Generation (RG-S(0, k)) [9]: A pointwise approach based on Fine-Grained Relevance Labels. Since only the RG-S(0, k) results on the BEIR dataset have been published [9], we use this benchmark solely for Touche results.

6.3 Results

In this section, we present the results of the empirical experiments that we conducted to answer our two research questions. To investigate the impact of confidence elicitation and self-consistency on pointwise zero-shot document rankers using LLMs we address each of our research questions as follows: We first describe the motivation behind each research question; we then present the approach used to answer the RQ; finally, we present our results and discuss the implications of our findings for each RQ.

6.3.1 RQ1: What is the impact of confidence elicitation on pointwise zero-shot LLM-based document rankers?

Motivation

Many research studies have leveraged LLMs as pointwise zero-shot document rankers [9,13, 15]. However, despite these efforts, prior research has shown that pointwise methods provide

lower effectiveness compared to other LLM-based zero-shot rankers [165].

In parallel, [133] introduced verbalized confidence, where LLMs are explicitly prompted to express their confidence levels. Several studies have emphasized enabling LLMs to express confidence in their responses [19, 135], as confidence expression is crucial for trustworthy decision-making.

Despite the advancements in pointwise approaches, previous works have yet to include the LLM's confidence in generating relevance scores. This gap raises questions about how verbalized confidence might affect the quality of generated relevance labels in zero-shot LLM-based rankers. Therefore, this research question investigates the impact of incorporating verbalized confidence into pointwise zero-shot LLM-based rankers.

Approach

In this RQ we rely on the datasets, prompt, and implementation and metrics presented in Section 6.2.

We compare our results to prior approaches (See Section 6.2.3), to determine whether confidence elicitation can provide competitive results.

However, because Mixtral is not well represented in related works, it can be difficult to determine whether the results are related to confidence elicitation, or simply due to Mixtral's inherent ability with the re-ranking task. Therefore, as a sort of ablation study for RQ1, we also create a base prompt, devoid of confidence elicitation. We then use this prompt to identify a baseline performance for Mixtral, allowing us to study how much of the resulting score is due to confidence elicitation, and how much should be expected based on Mixtral alone.

To further investigate the impact of verbalized confidence we study the impact of temperature on confidence elicitation. We use a temperature sampling strategy to generate multiple outputs for each document in response to a query. Our temperature values, ranging from 0.0 to 0.8, include six levels: [0.0, 0.2, 0.4, 0.6, 0.7, 0.8], where 0.0 represents the most deterministic setting and 0.8 represents the least deterministic. These temperatures are chosen based on prior works that have studied the effect of temperature on LLM rankers [132].

Results and Findings

Our results for this research question are shown in Table 6.1. Overall, our results show that Confidence Elicitation (CE) can provide substantial improvements over the base capabilities

Table 6.1 Results (nDCG@10) on TREC and Touche based on RQ1.

Methods	Temp	DL19	DL20	Touche
BM25	-	50.58	47.96	44.22
Unsupervised pointwise LLM-based				
UPR (FLAN-T5-XXL)	-	62.00	60.34	21.56
RG (FLAN-T5-XXL)	0	64.48	62.58	22.10
UPR (FLAN-UL2)	-	58.95	60.02	23.68
RG (FLAN-UL2)	0	64.61	**65.39**	24.67
RG-S(0, 4)	-	-	-	27.57
RQ1 (Confidence Elicitation)				
Confidence Elicitation	0	**69.34**	64.91	33.94
	0.2	64.97	62.65	33.61
Mixtral-8x7B-Instruct	0.4	65.18	61.24	**36.25**
	0.6	63.81	59.03	29.26
	0.7	60.71	56.53	29.22
	0.8	65.11	59.01	29.29
Baseline Without Confidence Elicitation				
Mixtral-8x7B-Instruct	0	64.02	59.76	34.77

of our chosen LLM and that these results are in line with our comparison baselines. However, the results of CE are inconsistent. We make the following observations:

Confidence elicitation can provide an improvement in re-ranking capabilities. Indeed, for DL19 the results can take the base Mixtral from a middling performance, to better than any of our comparison baselines.

Confidence elicitation is not a guarantee of best performance. For example, while it does enhance the base Mixtral score in DL20, the improvement is not enough to outperform RG (FLAN-UL2), no matter the temperature.

Confidence elicitation is temperature sensitive. For Touche, the best result is obtained with confidence elicitation at a temperature of 0.4. All other temperatures present a decrease in performance over the base Mixtral.

Our results show that it is generally worthwhile to consider confidence elicitation for pointwise zero-shot LLM-based document re-ranking for low-temperature settings. While DL19 shows the most improvement from CE at a temperature of 0 (generally regarded as a default setting), DL20 also presents a large improvement at the same temperature. Indeed, only in the case of Touche do the results over the base Mixtral decrease for a temperature of 0. Even in the case of Touche, the resulting drop in performance is less than what is seen between our

comparison baselines for Touche (i.e., less than 1 point).

However, our results also show that confidence elicitation suffers from a lack of stability. By varying temperatures, we can see that the results of CE present non-monotonic trends over all three datasets. In all three datasets, a temperature of 0 provides generally 'good' results. However, Touche shows the best results with a temperature of 0.4, while that same temperature provides middling results for DL19 and DL20. We further investigate the impact of self-consistency and temperature on confidence elicitation in RQ2.

Confidence elicitation can improve the performance of pointwise zero-shot LLM-based document re-ranking, able to turn an average model into a pack leader. However, the process is temperature sensitive, where the worst-case scenario can even decrease performance.

6.3.2 RQ2: What is the impact of self-consistency when considering confidence on pointwise zero-shot LLM-based document rankers?

Motivation

LLMs exhibit instability during inference, which can lead to a lack of trustworthiness in the generated reasoning steps [7]. One prominent solution is self-consistency, which involves sampling multiple reasoning paths, performing an ensemble over the corresponding answers, and selecting the most consistent one through a majority voting process [18].

While the study [132] introduces the concept of permutation self-consistency to address positional biases in LLMs during listwise ranking tasks, to the best of our knowledge, previous studies have not yet explored the impact of self-consistency in the pointwise approach. Furthermore, our first RQ shows that confidence elicitation can both improve document ranking for pointwise zero-shot LLM-based rankers and is sensitive to temperature. Therefore, in this research question, we investigate the effect of applying self-consistency within pointwise zero-shot LLM-based rankers by generating multiple relevance scores per document and aggregating these scores using three different methods. Throughout this process, we also consider confidence elicitation. Through this approach, we aim to evaluate whether generating multiple answers, rather than relying on a single score, can enhance the performance of pointwise re-ranking.

Approach

In this RQ we rely on the datasets, prompt, and implementation and metrics presented in Section 6.2.

To investigate the effect of self-consistency in pointwise zero-shot LLM-based rankers, we conduct experiments using two temperature sampling strategies to generate diverse outputs for aggregation over them. In the first approach, we employ five distinct temperature values ([0.0, 0.2, 0.4, 0.6, 0.8]) to rigorously assess the impact of temperature variation on generating pointwise relevance scores, similar to the method used in [132]. While their temperature range was limited to 0 to 0.75, we extended it to 0.8 to keep temperatures with equal distances from each other. In the second strategy, following the approach in [18], we kept the temperature constant at 0.7 to gather a diverse answer set. We run the LLM five times to assess the consistency of the model's outputs under identical conditions, while also capturing the confidence levels.

To increase the robustness of the final relevance prediction we aim to mitigate potential biases from individual inference runs by enabling multiple reasoning paths. To do so, the final relevance score for each document, relative to the query, is calculated using three different aggregation methods that consider both the relevance score and the verbalized confidence score from each of the five LLM runs.

For the following three approaches s_i *is the relevance score,* c_i *the confidence level for document p from LLM result i, and n is the number of LLM responses for each document.* These three aggregation approaches are:

Averaging Approach: For each document, we compute the mean of both the relevance scores and the confidence levels across the five LLM outputs. This method provides an aggregate score that reflects the average relevance and confidence for the document, capturing the overall judgment from the multiple LLM runs.

$$\text{Average Score}(p) = \frac{1}{n}\sum_{i=1}^{n} s_i$$

$$\text{Average Confidence}(p) = \frac{1}{n}\sum_{i=1}^{n} c_i$$

Weighted Average Approach: For each document, we calculate a weighted average of the relevance scores, where the weight of each score is determined by its corresponding confidence level. This method accounts for both the relevance score and the confidence in that score, giving more influence to highly confident predictions, resulting in a more nuanced aggregation of relevance and confidence across the five LLM results.

$$\text{Weighted Average Score}(p) = \frac{\sum_{i=1}^{n} s_i \cdot c_i}{\sum_{i=1}^{n} c_i}$$

Majority Vote Approach: For each document, we identify the most frequent relevance label from the five LLM outputs and select that as the final relevance label. We then compute the mean confidence level of the predictions corresponding to the majority label. This method emphasizes the consistency of the LLM's output, ensuring that the most commonly predicted relevance is chosen while also reflecting the associated confidence.

We use the same baselines as in RQ1 (See Section 6.2.3) to compare the results of this RQ.

Results and Findings

Table 6.2 Results (nDCG@10) on TREC and Touche datasets based on RQ2.

Methods	**DL19**	**DL20**	**Touche**
BM25	50.58	47.96	44.22
Unsupervised pointwise LLM-based			
UPR (FLAN-T5-XXL)	62.00	60.34	21.56
RG (FLAN-T5-XXL)	64.48	62.58	22.10
UPR (FLAN-UL2)	58.95	60.02	23.68
RG (FLAN-UL2)	64.61	**65.39**	24.67
RG-S(0, 4)	-	-	27.57
RQ2 (Self-Consistency along with Confidence Elicitation)			
Different temperatures [0.0, 0.2, 0.4, 0.6, 0.8]			
Averaging	65.95	61.46	24.74
Weighted Average	66.93	63.36	31.22
Majority Vote	64.64	58.77	23.89
Constant temperatures 0.7			
Averaging	64.75	59.17	23.96
Weighted Average	**67.56**	61.88	28.09
Majority Vote	61.14	54.52	22.74
Baseline Without Confidence Elicitation			
Mixtral-8x7B-Instruct	64.02	59.76	**34.77**

Our results for this research question are shown in Table 6.2. Our results show that using temperature-based self-consistency does have an impact on pointwise zero-shot LLM-based document re-rankers. We make the following observations:

The Weighted Average method achieves the highest performance across both temperature sampling strategies. It surpasses the other two approaches (Averaging and Majority Vote) on all datasets.

Despite being introduced in previous work [18], the Majority Vote approach exhibits the lowest

performance across all datasets, suggesting that it is less effective for pointwise re-rankers.

Our results show that the raw frequency of answers from an LLM in pointwise re-ranking is not as useful as the confidence score one can elicit from them. Indeed, our results also show that using confidence elicitation through an average aggregation method is better than simply taking a majority vote. This is the case even when the exact same prompt is used. This result is further cemented by the weighted average showing even better than average results. This shows that factoring in the confidence of an LLM in its re-ranking task does yield improvements over using the raw frequency of answers (e.g., Majority Vote).

Our results also show that our chosen LLM generally performs better when considering self-consistency and confidence elicitation than the baseline LLM without confidence elicitation. This is true of both DL19 and DL20 for self-consistency with constant temperature and different temperatures. However, this is not the case for Touche, where the baseline LLM without confidence elicitation provides better results. This shows that while LLM can provide a confidence score that can help pointwise zero-shot LLM-based document re-rankers, these scores are not necessarily trustworthy. If these confidence scores were stable and trustworthy, we would expect the weighted average methods to provide results that are superior to the baseline without confidence elicitation in all cases.

While prior work has shown that temperature changes have little effect on the quality of list-wise re-ranking [132], our results show that they do affect pointwise re-ranking. While RQ1 showed that generally, low temperatures (0 for DL19 and DL 20, and 0.4 for Touche) could provide high-quality results, RQ2 paints a bleaker picture. None of the best results from our self-consistency experiments (i.e., RQ2) outperform the best temperature results from RQ1. This shows that while confidence elicitation *can* improve pointwise zero-shot LLM-based document re-ranking, the results are inconsistent. While temperature sampling can smooth out the largest errors in re-ranking, it does not provide a guarantee of the best results.

Pointwise zero-shot LLM-based document rankers are inconsistent even when considering confidence. While considering different temperatures and multiple queries through aggregation can smooth out some errors, it does not provide the best possible results.

6.4 Limitations

The limitations of this work include several key challenges. First, our methodology involves making multiple calls to LLMs, which demands substantial hardware resources, thereby increasing the financial cost of experimentation. Additionally, the time required to run the LLM multiple times is significant, making the process highly time-consuming. Due to re-

source constraints, our experiments are conducted on only three ranking datasets. As a result, the generalizability of our findings to other datasets remains uncertain, and it is possible that results could vary under different dataset characteristics, such as document length, query complexity, or topic domain. Expanding this study to include a wider variety of datasets would provide a more comprehensive understanding of LLM performance in different contexts. Furthermore, we limited our investigation to a single LLM that was 4-bit quantized to keep the experiments manageable with the available computing resources. The impact of this reduction in precision may have influenced our results, meaning that using a full-precision model or different quantization strategies could yield improved or more accurate outcomes. Another key limitation of this study is that it exclusively examines the re-ranking task. As such, the effectiveness of the re-ranking process is inherently bounded by the quality of the initial passage retrieval stage, which in our case is conducted using BM25.

6.5 Conclusion

Recent studies have demonstrated the potential of LLM as document re-rankers. While pointwise reranking approaches have been shown to underperform compared to other approaches, they remain needed for some applications. Several approaches have been proposed to improve LLM prompting and indeed re-ranking. Through this empirical study, we show that confidence elicitation can be useful for pointwise zero-shot LLM-based document re-rankers and that it generally outperforms prompts without elicitation. We also show that considering pointwise zero-shot LLM-based document re-ranking from multiple runs (with or without different temperatures) can smooth out re-ranking errors and provide better results than simple majority voting. While our results come with some caveats, we believe that the understanding obtained through our study can help future pointwise LLM-based research further improve over the state-of-the-art.

CHAPTER 7 CONCLUSION

This thesis explores two closely related fields: Recommender Systems and Information Retrieval (IR), focusing on their methodologies, challenges, and advancements. In this chapter, we provide a comprehensive conclusion to the thesis by summarizing its key contributions and findings, identifying limitations, and proposing directions for future research. The chapter begins with an overview of the works conducted, followed by a discussion of the limitations that emerged during the research. Finally, it concludes with suggestions for future work, aiming to address these limitations and further advance the state-of-the-art in Recommender Systems and IR.

7.1 Summary of Works

This thesis has addressed two critical challenges in the fields of Recommender Systems and Information Retrieval (IR): enhancing the explainability of recommender systems and improving re-ranking performance in IR systems using Large Language Models (LLMs). In Chapter 4, we developed an explainable recommender system for an airline partner, providing personalized destination recommendations with transparent, feature-based explanations to the users. This system improves user trust and satisfaction by bridging the gap between algorithmic complexity and user understanding. In Chapters 5 and 6, we explored ranking challenges in IR systems, focusing on zero-shot passage re-ranking using LLMs. We proposed a novel LLM-based pointwise pre-filtering step, before the re-ranker, to identify relevant passages from non-relevant ones and remove the irrelevant ones before passing them to the re-ranker, which enhances the performance of subsequent listwise re-ranking. Our experiments demonstrated that incorporating a small set of human-generated relevance scores with LLM relevance scoring improved filtering and re-ranking performance. Additionally, we conduct an empirical study to investigate the roles of self-consistency and confidence elicitation in improving the effectiveness and accuracy of pointwise re-rankers. We find that confidence elicitation can enhance the performance of pointwise zero-shot LLM-based passage re-ranking. We also find that the raw frequency of answers from an LLM in pointwise re-ranking is not as useful as the confidence score one can elicit from them.

7.2 Limitations

The destination recommender system presented in this thesis relies on a predefined list of keywords, which can oversimplify city descriptions. Cities are inherently complex, and this limitation may result in the loss of nuanced qualities, such as a city which is blend of modern and historical architecture. This oversimplification can affect the relevance of recommendations, particularly for users with diverse interests. Furthermore, the performance of the system depends heavily on the quality of data sources. Cities with richer documentation in these sources may dominate the recommendations, reducing diversity. Additionally, the evaluation relies on subjective expert reviews and the Precision@k metric, which focuses on relevance but overlooks diversity and novelty—essential factors for engaging recommendations. The lack of a re-ranking step further limits the ability of the system to personalize recommendations based on user preferences. Finally, the exclusion of large language models (LLMs) from the experiments represents a missed opportunity to enhance contextual understanding and capture nuanced city characteristics.

The proposed methods and experiments in ranking tasks in information retrieval experiments also face challenges. The threshold-setting process introduces randomness, as the initial threshold is selected arbitrarily and optimized based on the F1 score. Furthermore, these thresholds are dataset-dependent and may not generalize to other datasets. The heavy reliance on qrels files and the interpretation of relevance levels further limit generalizability. Resource constraints, particularly limited GPU availability, restricted experiments to four datasets rather than the broader range typically used in similar studies. While improvements were observed with open-source models like Llama-2-13b, the results were not comparable to the state-of-the-art. The quantization of the LLM to 4-bit precision, though necessary to manage resources, may have affected outcomes, and future work could explore the use of full-precision models or alternative quantization strategies. Additionally, regarding the experiments in self-consistency and confidence elicitation, the methodology's reliance on multiple LLM calls demands substantial computational resources, resulting in high financial and time costs. These constraints limited experiments to three ranking datasets, raising concerns about the generalizability of findings to datasets with different characteristics, such as document length or query complexity. Additionally, the study focuses solely on the re-ranking task, making its effectiveness dependent on the quality of the initial passage retrieval stage, conducted using BM25.

7.3 Future Research

Building on the limitations identified in this study, several directions for future research arise to enhance the performance and generalizability of the destination recommender system and information retrieval experiments.

Future work can focus on reducing the reliance on predefined keywords by leveraging advanced natural language processing techniques, such as topic modeling or embeddings, to capture the multi-faceted nature of cities more comprehensively. Incorporating large language models (LLMs) into the system could further improve the contextual understanding of city descriptions, enabling more personalized and nuanced recommendations. To address the limitations of current evaluation methods, future studies could include additional metrics, such as recall, diversity, or user satisfaction, to provide a more holistic assessment of recommendation quality. The inclusion of a re-ranking component based on user preferences could also enhance the ability of the system to prioritize the most relevant cities and features, improving personalization and relevance. Additionally, expanding the dataset to include under-documented cities could increase the diversity of recommendations.

In the context of ranking tasks in IR systems, future research could explore more robust approaches to threshold-setting that are less reliant on arbitrary initial values. Adaptive or data-driven threshold optimization methods could help generalize the findings across diverse datasets. Further validation of the proposed methods on a broader range of datasets, including those with varied query complexities and document lengths, would provide deeper insights into their applicability. Expanding experiments to include a wider variety of LLMs with different quantization strategies or full-precision models could offer a better understanding of the trade-offs between computational efficiency and performance. Additionally, incorporating models beyond BM25 in the initial retrieval stage may enhance the overall effectiveness of the re-ranking process. In addition, the pre-filtering step should be tested with other methods such as setwise and pairwise ranking to investigate its performance on different methods. Addressing resource constraints is critical for scaling these methodologies. Finally, future work could investigate techniques to reduce the computational cost of multiple LLM calls, such as leveraging more efficient inference strategies, distributed computing, or pruning methods. Research on optimizing LLM performance for specific tasks or datasets may also help manage resource demands while maintaining accuracy.

REFERENCES

[1] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, H. Chen, Z. Dou, and J.-R. Wen, "Large language models for information retrieval: A survey," *arXiv preprint arXiv:2308.07107*, 2023.

[2] W. Fan, X. Zhao, X. Chen, J. Su, J. Gao, L. Wang, Q. Liu, Y. Wang, H. Xu, L. Chen *et al.*, "A comprehensive survey on trustworthy recommender systems," *arXiv preprint arXiv:2209.10117*, 2022.

[3] J. Chen, L. Ma, X. Li, N. Thakurdesai, J. Xu, J. H. Cho, K. Nag, E. Korpeoglu, S. Kumar, and K. Achan, "Knowledge graph completion models are few-shot learners: An empirical study of relation labeling in e-commerce with llms," *arXiv preprint arXiv:2305.09858*, 2023.

[4] X. Wang, X. He, F. Feng, L. Nie, and T.-S. Chua, "Tem: Tree-enhanced embedding model for explainable recommendation," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1543–1552.

[5] X. Chen, W. Fan, J. Chen, H. Liu, Z. Liu, Z. Zhang, and Q. Li, "Fairly adaptive negative sampling for recommendations," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 3723–3733.

[6] T. Qin, T.-Y. Liu, J. Xu, and H. Li, "Letor: A benchmark collection for research on learning to rank for information retrieval," *Information Retrieval*, vol. 13, pp. 346–374, 2010.

[7] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[8] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[9] H. Zhuang, Z. Qin, K. Hui, J. Wu, L. Yan, X. Wang, and M. Berdersky, "Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels," *arXiv preprint arXiv:2310.14122*, 2023.

[10] W. Sun, L. Yan, X. Ma, P. Ren, D. Yin, and Z. Ren, "Is chatgpt good at search? investigating large language models as re-ranking agent," *arXiv preprint arXiv:2304.09542*, 2023.

[11] S. Zhuang, H. Zhuang, B. Koopman, and G. Zuccon, "A setwise approach for effective and highly efficient zero-shot ranking with large language models," *arXiv preprint arXiv:2310.09497*, 2023.

[12] Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, J. Shen, T. Liu, J. Liu, D. Metzler, X. Wang *et al.*, "Large language models are effective text rankers with pairwise ranking prompting," *arXiv preprint arXiv:2306.17563*, 2023.

[13] D. S. Sachan, M. Lewis, M. Joshi, A. Aghajanyan, W.-t. Yih, J. Pineau, and L. Zettlemoyer, "Improving passage retrieval with zero-shot question generation," *arXiv preprint arXiv:2204.07496*, 2022.

[14] X. Ma, X. Zhang, R. Pradeep, and J. Lin, "Zero-shot listwise document reranking with a large language model," *arXiv preprint arXiv:2305.02156*, 2023.

[15] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar *et al.*, "Holistic evaluation of language models," *arXiv preprint arXiv:2211.09110*, 2022.

[16] R. Nogueira, Z. Jiang, and J. Lin, "Document ranking with a pretrained sequence-to-sequence model," *arXiv preprint arXiv:2003.06713*, 2020.

[17] L. Wang, W. Xu, Y. Lan, Z. Hu, Y. Lan, R. K.-W. Lee, and E.-P. Lim, "Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models," *arXiv preprint arXiv:2305.04091*, 2023.

[18] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[19] M. Xiong, Z. Hu, X. Lu, Y. Li, J. Fu, J. He, and B. Hooi, "Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms," *arXiv preprint arXiv:2306.13063*, 2023.

[20] C. D. Manning, *An introduction to information retrieval*, 2009.

[21] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees, "Overview of the trec 2019 deep learning track," *arXiv preprint arXiv:2003.07820*, 2020.

[22] N. Craswell, B. Mitra, E. Yilmaz, and D. Campos, "Overview of the trec 2020 deep learning track," 2021.

[23] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=wCu6T5xFjeJ

[24] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[25] S. E. Robertson and S. Walker, "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval," in *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University.* Springer, 1994, pp. 232–241.

[26] W. Fan, Y. Ma, Q. Li, J. Wang, G. Cai, J. Tang, and D. Yin, "A graph neural network framework for social recommendations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 5, pp. 2033–2047, 2020.

[27] Q. Shen, W. Tao, J. Zhang, H. Wen, Z. Chen, and Q. Lu, "Sar-net: A scenario-aware ranking network for personalized fair recommendation in hundreds of travel scenarios," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4094–4103.

[28] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, and J. Zhang, "Chat-rec: Towards interactive and explainable llms-augmented recommender system," *arXiv preprint arXiv:2303.14524*, 2023.

[29] Y. Gong, Z. Jiang, Y. Feng, B. Hu, K. Zhao, Q. Liu, and W. Ou, "Edgerec: recommender system on edge in mobile taobao," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2477–2484.

[30] S. C. Geyik, S. Ambler, and K. Kenthapadi, "Fairness-aware ranking in search & recommendation systems with application to linkedin talent search," in *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining*, 2019, pp. 2221–2231.

[31] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[32] W. Fan, Y. Ma, D. Yin, J. Wang, J. Tang, and Q. Li, "Deep social collaborative filtering," in *Proceedings of the 13th ACM conference on recommender systems*, 2019, pp. 305–313.

[33] W. Fan, Q. Li, and M. Cheng, "Deep modeling of social relations for recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[34] X. Zhao, H. Liu, W. Fan, H. Liu, J. Tang, and C. Wang, "Autoloss: Automated loss function search in recommendations," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 3959–3967.

[35] X. Zhaok, H. Liu, W. Fan, H. Liu, J. Tang, C. Wang, M. Chen, X. Zheng, X. Liu, and X. Yang, "Autoemb: Automated embedding dimensionality search in streaming recommendations," in *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021, pp. 896–905.

[36] F. Vasile, E. Smirnova, and A. Conneau, "Meta-prod2vec: Product embeddings using side-information for recommendation," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 225–232.

[37] Z. Zhao, W. Fan, J. Li, Y. Liu, X. Mei, Y. Wang, Z. Wen, F. Wang, X. Zhao, J. Tang *et al.*, "Recommender systems in the era of large language models (llms)," *arXiv preprint arXiv:2307.02046*, 2023.

[38] R. Burke, "Hybrid web recommender systems," *The adaptive web: methods and strategies of web personalization*, pp. 377–408, 2007.

[39] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[40] Y. Zhang, X. Chen *et al.*, "Explainable recommendation: A survey and new perspectives," *Foundations and Trends® in Information Retrieval*, vol. 14, no. 1, pp. 1–101, 2020.

[41] T. Kamishima, S. Akaho, H. Asoh, and I. Sato, "Model-based approaches for independence-enhanced recommendation," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2016, pp. 860–867.

[42] C. Wang, H. Zhu, C. Zhu, X. Zhang, E. Chen, and H. Xiong, "Personalized employee training course recommendation with career development awareness," in *Proceedings of the Web Conference 2020*, 2020, pp. 1648–1659.

[43] S. Verma, V. Boonsanong, M. Hoang, K. Hines, J. Dickerson, and C. Shah, "Counterfactual explanations and algorithmic recourses for machine learning: A review," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–42, 2024.

[44] S. Jesus, C. Belém, V. Balayan, J. Bento, P. Saleiro, P. Bizarro, and J. Gama, "How can i choose an explainer? an application-grounded evaluation of post-hoc explanations," in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021, pp. 805–815.

[45] H. Park, H. Jeon, J. Kim, B. Ahn, and U. Kang, "Uniwalk: Explainable and accurate recommendation for rating and network data," *arXiv preprint arXiv:1710.07134*, 2017.

[46] Y. Zhu, Y. Xian, Z. Fu, G. De Melo, and Y. Zhang, "Faithfully explainable recommendation via neural logic reasoning," *arXiv preprint arXiv:2104.07869*, 2021.

[47] A. Sharma and D. Cosley, "Do social explanations work? studying and modeling the effects of social explanations in recommender systems," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 1133–1144.

[48] N. Tintarev and J. Masthoff, "Designing and evaluating explanations for recommender systems," in *Recommender systems handbook*. Springer, 2010, pp. 479–510.

[49] ——, "Explaining recommendations: Design and evaluation," in *Recommender systems handbook*. Springer, 2015, pp. 353–382.

[50] K. Balog and F. Radlinski, "Measuring recommendation explanation quality: The conflicting goals of explanations," in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 2020, pp. 329–338.

[51] H. Cramer, V. Evers, S. Ramlal, M. Van Someren, L. Rutledge, N. Stash, L. Aroyo, and B. Wielinga, "The effects of transparency on trust in and acceptance of a content-based art recommender," *User Modeling and User-adapted interaction*, vol. 18, pp. 455–496, 2008.

[52] B. Ferwerda, K. Swelsen, and E. Yang, "Explaining content-based recommendations," *New York*, pp. 1–24, 2018.

[53] M. Pazzani, "Content-based recommendation systems," 2007.

[54] J. Vig, S. Sen, and J. Riedl, "Tagsplanations: explaining recommendations using tags," in *Proceedings of the 14th international conference on Intelligent user interfaces*, 2009, pp. 47–56.

[55] Y. Hou, N. Yang, Y. Wu, and P. S. Yu, "Explainable recommendation with fusion of aspect information," *World Wide Web*, vol. 22, pp. 221–240, 2019.

[56] J. Singh and A. Anand, "Posthoc interpretability of learning to rank models using secondary training data," *arXiv preprint arXiv:1806.11330*, 2018.

[57] G. Peake and J. Wang, "Explanation mining: Post hoc interpretability of latent factor models for recommendation systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2060–2069.

[58] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[59] J. Singh and A. Anand, "Exs: Explainable search using local model agnostic interpretability," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 770–773.

[60] W. Cheng, Y. Shen, L. Huang, and Y. Zhu, "Incorporating interpretability into latent factor models via fast influence analysis," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 885–893.

[61] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha, "Sequential recommendation with user memory networks," in *Proceedings of the eleventh ACM international conference on web search and data mining*, 2018, pp. 108–116.

[62] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *Advances in neural information processing systems*, vol. 13, 2000.

[63] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[64] A. Moreo, A. Esuli, and F. Sebastiani, "Word-class embeddings for multiclass text classification," *Data Mining and Knowledge Discovery*, vol. 35, pp. 911–963, 2021.

[65] T. Mikolov, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[66] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[67] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[68] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt, "Yake! keyword extraction from single documents using multiple local features," *Information Sciences*, vol. 509, pp. 257–289, 2020.

[69] A. Bougouin, F. Boudin, and B. Daille, "Topicrank: Graph-based topic ranking for keyphrase extraction," in *International joint conference on natural language processing (IJCNLP)*, 2013, pp. 543–551.

[70] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text mining: applications and theory*, pp. 1–20, 2010.

[71] G. Salton, C.-S. Yang, and C. T. Yu, "A theory of term importance in automatic text analysis," *Journal of the American society for Information Science*, vol. 26, no. 1, pp. 33–44, 1975.

[72] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[73] C. Florescu and C. Caragea, "Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents," in *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: long papers)*, 2017, pp. 1105–1115.

[74] Z. Liu, W. Huang, Y. Zheng, and M. Sun, "Automatic keyphrase extraction via topic decomposition," in *Proceedings of the 2010 conference on empirical methods in natural language processing*, 2010, pp. 366–376.

[75] P. D. Turney, "Learning algorithms for keyphrase extraction," *Information retrieval*, vol. 2, pp. 303–336, 2000.

[76] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "Kea: Practical automatic keyphrase extraction," in *Proceedings of the fourth ACM conference on Digital libraries*, 1999, pp. 254–255.

[77] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proceedings of the 2003 conference on Empirical methods in natural language processing*, 2003, pp. 216–223.

[78] O. Medelyan, E. Frank, and I. H. Witten, "Human-competitive tagging using automatic keyphrase extraction." Association for Computational Linguistics, 2009.

[79] C. Caragea, F. Bulgarov, A. Godea, and S. D. Gollapalli, "Citation-enhanced keyphrase extraction from research papers: A supervised approach," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1435–1446.

[80] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi, "Deep keyphrase generation," *arXiv preprint arXiv:1704.06879*, 2017.

[81] S. D. Gollapalli, X.-L. Li, and P. Yang, "Incorporating expert knowledge into keyphrase extraction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[82] K. S. Hasan and V. Ng, "Automatic keyphrase extraction: A survey of the state of the art," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 1262–1273.

[83] E. Papagiannopoulou and G. Tsoumakas, "A review of keyphrase extraction," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 2, p. e1339, 2020.

[84] M. Grootendorst, "Keybert: Minimal keyword extraction with bert." 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4461265

[85] P. D. Turney, "Learning to extract keyphrases from text," *arXiv preprint cs/0212013*, 2002.

[86] M. Kulkarni, D. Mahata, R. Arora, and R. Bhowmik, "Learning rich representation of keyphrases from text. corr abs/2112.08547 (2021)," *URL: https://arxiv.org/abs/2112.08547*.

[87] A. Hulth and B. Megyesi, "A study on automatically extracted keywords in text categorization," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 2006, pp. 537–544.

[88] K. M. Hammouda, D. N. Matute, and M. S. Kamel, "Corephrase: Keyphrase extraction for document clustering," in *International workshop on machine learning and data mining in pattern recognition*. Springer, 2005, pp. 265–274.

[89] V. Qazvinian, D. Radev, and A. Özgür, "Citation summarization through keyphrase extraction," in *Proceedings of the 23rd international conference on computational linguistics (COLING 2010)*, 2010, pp. 895–903.

[90] I. Augenstein, M. Das, S. Riedel, L. Vikraman, and A. McCallum, "Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications," *arXiv preprint arXiv:1704.02853*, 2017.

[91] D. K. Sanyal, P. K. Bhowmick, P. P. Das, S. Chattopadhyay, and T. Santosh, "Enhancing access to scholarly publications with surrogate resources," *Scientometrics*, vol. 121, pp. 1129–1164, 2019.

[92] M. Song, I. Y. Song, R. B. Allen, and Z. Obradovic, "Keyphrase extraction-based query expansion in digital libraries," in *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, 2006, pp. 202–209.

[93] S. Jones and M. S. Staveley, "Phrasier: a system for interactive document retrieval using keyphrases," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999, pp. 160–167.

[94] E. Çano and O. Bojar, "Keyphrase generation: A multi-aspect survey," in *2019 25th Conference of Open Innovations Association (FRUCT)*. IEEE, 2019, pp. 85–94.

[95] V. Sanh, "Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[96] M. Team, "Ml6 keyphrase extraction distilbert (inspec)," https://huggingface.co/ml6team/keyphrase-extraction-distilbert-inspec, 2021, accessed: 2024-10-29.

[97] M. Shanahan, "Talking about large language models," *Communications of the ACM*, vol. 67, no. 2, pp. 68–79, 2024.

[98] B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, vol. 1, 2020.

[99] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[100] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.

[101] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[102] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[103] S. Huang, L. Dong, W. Wang, Y. Hao, S. Singhal, S. Ma, T. Lv, L. Cui, O. K. Mohammed, B. Patra *et al.*, "Language is not all you need: Aligning perception with language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 72 096–72 109, 2023.

[104] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, "A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt," *arXiv preprint arXiv:2303.04226*, 2023.

[105] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[106] V. Liu and L. B. Chilton, "Design guidelines for prompt engineering text-to-image generative models," in *Proceedings of the 2022 CHI conference on human factors in computing systems*, 2022, pp. 1–23.

[107] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint arXiv:2302.11382*, 2023.

[108] J. Jiang, K. Zhou, Z. Dong, K. Ye, W. X. Zhao, and J.-R. Wen, "Structgpt: A general framework for large language model to reason over structured data," *arXiv preprint arXiv:2305.09645*, 2023.

[109] R. Ren, Y. Wang, Y. Qu, W. X. Zhao, J. Liu, H. Tian, H. Wu, J.-R. Wen, and H. Wang, "Investigating the factual knowledge boundary of large language models with retrieval augmentation," *arXiv preprint arXiv:2307.11019*, 2023.

[110] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

[111] Z. Chen, K. Zhou, B. Zhang, Z. Gong, W. X. Zhao, and J.-R. Wen, "Chatcot: Tool-augmented chain-of-thought reasoning on chat-based large language models," *arXiv preprint arXiv:2305.14323*, 2023.

[112] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[113] S. K. K. Santu and D. Feng, "Teler: A general taxonomy of llm prompts for benchmarking complex tasks," *arXiv preprint arXiv:2305.11430*, 2023.

[114] S.-Y. Miao, C.-C. Liang, and K.-Y. Su, "A diverse corpus for evaluating and developing english math word problem solvers," *arXiv preprint arXiv:2106.15772*, 2021.

[115] A. Talmor, J. Herzig, N. Lourie, and J. Berant, "Commonsenseqa: A question answering challenge targeting commonsense knowledge," *arXiv preprint arXiv:1811.00937*, 2018.

[116] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[117] Z. Chu, J. Chen, Q. Chen, W. Yu, T. He, H. Wang, W. Peng, M. Liu, B. Qin, and T. Liu, "A survey of chain of thought reasoning: Advances, frontiers and future," *arXiv preprint arXiv:2309.15402*, 2023.

[118] S. Yang, H. Zhao, S. Zhu, G. Zhou, H. Xu, Y. Jia, and H. Zan, "Zhongjing: Enhancing the chinese medical capabilities of large language model through expert feedback and real-world multi-turn dialogue," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 19 368–19 376.

[119] W. Chen, X. Ma, X. Wang, and W. W. Cohen, "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," *arXiv preprint arXiv:2211.12588*, 2022.

[120] J. X. Zhao, Y. Xie, K. Kawaguchi, J. He, and M. Q. Xie, "Automatic model selection with large language models for reasoning," *arXiv preprint arXiv:2305.14333*, 2023.

[121] Y. Li, Z. Lin, S. Zhang, Q. Fu, B. Chen, J.-G. Lou, and W. Chen, "Making large language models better reasoners with step-aware verifier," *arXiv preprint arXiv:2206.02336*, 2022.

[122] Y. Fu, H. Peng, A. Sabharwal, P. Clark, and T. Khot, "Complexity-based prompting for multi-step reasoning," in *The Eleventh International Conference on Learning Representations*, 2022.

[123] O. Yoran, T. Wolfson, B. Bogin, U. Katz, D. Deutch, and J. Berant, "Answering questions by meta-reasoning over multiple chains of thought," *arXiv preprint arXiv:2304.13007*, 2023.

[124] J. Qian, H. Wang, Z. Li, S. Li, and X. Yan, "Limitations of language models in arithmetic and symbolic induction," *arXiv preprint arXiv:2208.05051*, 2022.

[125] N. Bian, X. Han, L. Sun, H. Lin, Y. Lu, B. He, S. Jiang, and B. Dong, "Chatgpt is a knowledgeable but inexperienced solver: An investigation of commonsense problem in large language models," *arXiv preprint arXiv:2303.16421*, 2023.

[126] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le *et al.*, "Least-to-most prompting enables complex reasoning in large language models," *arXiv preprint arXiv:2205.10625*, 2022.

[127] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, "Pal: Program-aided language models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 10 764–10 799.

[128] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[129] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[130] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.

[131] X. Jiang, Y. Dong, L. Wang, Z. Fang, Q. Shang, G. Li, Z. Jin, and W. Jiao, "Self-planning code generation with large language models," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 7, pp. 1–30, 2024.

[132] R. Tang, X. Zhang, X. Ma, J. Lin, and F. Ture, "Found in the middle: Permutation self-consistency improves listwise ranking in large language models," *arXiv preprint arXiv:2310.07712*, 2023.

[133] S. Lin, J. Hilton, and O. Evans, "Teaching models to express their uncertainty in words," *arXiv preprint arXiv:2205.14334*, 2022.

[134] S. J. Mielke, A. Szlam, E. Dinan, and Y.-L. Boureau, "Reducing conversational agents' overconfidence through linguistic calibration," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 857–872, 2022.

[135] K. Tian, E. Mitchell, A. Zhou, A. Sharma, R. Rafailov, H. Yao, C. Finn, and C. D. Manning, "Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback," *arXiv preprint arXiv:2305.14975*, 2023.

[136] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon, "Adapting ranking svm to document retrieval," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 186–193.

[137] E. F. Harrington, "Online ranking/collaborative filtering using the perceptron algorithm," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 250–257.

[138] M. Collins, "Ranking algorithms for named entity extraction: Boosting and the voted-perceptron," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002, pp. 489–496.

[139] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," *arXiv preprint cs/0506075*, 2005.

[140] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *Proceedings of the 12th international conference on World Wide Web*, 2003, pp. 519–528.

[141] F. Crestani, M. Lalmas, C. J. Van Rijsbergen, and I. Campbell, ""is this document relevant?... probably" a survey of probabilistic models in information retrieval," *ACM Computing Surveys (CSUR)*, vol. 30, no. 4, pp. 528–552, 1998.

[142] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, "The vocabulary problem in human-system communication," *Communications of the ACM*, vol. 30, no. 11, pp. 964–971, 1987.

[143] T.-Y. Liu *et al.*, "Learning to rank for information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[144] P. Li, Q. Wu, and C. Burges, "Mcrank: Learning to rank using multiple classification and gradient boosting," *Advances in neural information processing systems*, vol. 20, 2007.

[145] L. Li and H.-T. Lin, "Ordinal regression by extended binary classification," *Advances in neural information processing systems*, vol. 19, 2006.

[146] R. Herbrich, T. Graepel, and K. Obermayer, "Support vector learning for ordinal regression," 1999.

[147] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Journal of machine learning research*, vol. 4, no. Nov, pp. 933–969, 2003.

[148] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 89–96.

[149] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 129–136.

[150] T. Qin, X.-D. Zhang, M.-F. Tsai, D.-S. Wang, T.-Y. Liu, and H. Li, "Query-level loss functions for information retrieval," *Information Processing & Management*, vol. 44, no. 2, pp. 838–855, 2008.

[151] J. Huang and B. J. Frey, "Structured ranking learning using cumulative distribution networks," *Advances in Neural Information Processing Systems*, vol. 21, 2008.

[152] J. Xu and H. Li, "Adarank: a boosting algorithm for information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 391–398.

[153] M. Taylor, J. Guiver, S. Robertson, and T. Minka, "Softrank: optimizing non-smooth rank metrics," in *Proceedings of the 2008 International Conference on Web Search and Data Mining*, 2008, pp. 77–86.

[154] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE transactions on knowledge and data engineering*, vol. 34, no. 1, pp. 50–70, 2020.

[155] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1253, 2018.

[156] S. Zad, M. Heidari, P. Hajibabaee, and M. Malekzadeh, "A survey of deep learning methods on semantic similarity and sentence modeling," in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2021, pp. 0466–0472.

[157] M. Trabelsi, Z. Chen, B. D. Davison, and J. Heflin, "Neural ranking models for document retrieval," *Information Retrieval Journal*, vol. 24, no. 6, pp. 400–444, 2021.

[158] M. N. Volkovs and R. S. Zemel, "Boltzrank: learning to maximize expected ranking gain," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 1089–1096.

[159] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[160] J. Lin, R. Nogueira, and A. Yates, *Pretrained transformers for text ranking: Bert and beyond*. Springer Nature, 2022.

[161] L. Wang, J. Lin, and D. Metzler, "A cascade ranking model for efficient ranked retrieval," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011, pp. 105–114.

[162] R. Nogueira, W. Yang, K. Cho, and J. Lin, "Multi-stage document ranking with bert," *arXiv preprint arXiv:1910.14424*, 2019.

[163] R. Nogueira, Z. Jiang, R. Pradeep, and J. Lin, "Document ranking with a pretrained sequence-to-sequence model," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He, and Y. Liu, Eds., Nov. 2020.

[164] H. Zhuang, Z. Qin, R. Jagerman, K. Hui, J. Ma, J. Lu, J. Ni, X. Wang, and M. Bendersky, "Rankt5: Fine-tuning t5 for text ranking with ranking losses," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023, pp. 2308–2313.

[165] S. Sun, S. Zhuang, S. Wang, and G. Zuccon, "An investigation of prompt variations for zero-shot llm-based rankers," *arXiv preprint arXiv:2406.14117*, 2024.