Retrievability in IR

MS Project Report

Submitted in Partial Fulfilment of the Requirements

for the Degree of

Master of Science

by

Aman Sinha

5th Year BS-MS, 18MS065

Under Supervision of

Dr. Dwaipayan Roy

Department of Computational and Data Sciences

DPS Coordinator:

Prof. Rangeet Bhattacharyya

Department of Physical Sciences

to

Department of Physical Sciences

Indian Institute of Science Education and Research (IISER) Kolkata

Mohanpur - 741246, INDIA

May 2023

DECLARATION

I, **Aman Sinha (Roll No: 18MS065)**, hereby declare that, this report titled **"Retrievability in IR"** submitted to Indian Institute of Science Education and Research Kolkata towards the partial requirement of **Master of Science** in **Department of Physical Sciences**, is an original work carried out by me under the supervision of **Dr. Dwaipayan Roy** and department coordinator **Prof. Rangeet Bhattacharyya** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold academic ethics and honesty. Whenever a piece of external information or statement or result is used then, that has been duly acknowledged and cited.

Kolkata - 741246 **Aman Sinha**

May 2023 **18MS065**

IISER KOLKATA

Certificate

This is to certify that the work contained in this project report entitled "**Retrievability in IR**" submitted by **Aman Sinha** (Roll No. **18MS065**) to the Indian Institute of Science Education and Research, Kolkata towards the partial requirement of **Master of Science** in **Department of Physical Sciences** has been carried out by him under my supervision and that it has not been submitted elsewhere for the award of any degree.

Dwaipayan Roy.

Dr. Dwaipayan Roy

Project Supervisor
IISER Kolkata

ACKNOWLEDGEMENT

I wish to extend my sincere and heartfelt obligation towards everyone who helped me see this project through till the end. First and foremost, I would like to extend my profound gratitude to Dr. Dwaipayan Roy, my esteemed supervisor from CDS department of IISER Kolkata. I am incredibly fortunate to have received his guidance, valuable feedback, and unwavering encouragement throughout the course of this research. Dr. Dwaipayan Roy's expertise, patience, and insightful suggestions have been instrumental in shaping this thesis. His constant support and belief in my abilities have been a constant source of motivation for me. I am truly grateful for his invaluable mentorship and unwavering commitment to my academic growth.

I am profoundly grateful to my parents for their unwavering support throughout my master's thesis. Despite the distance between us, their love and encouragement have been invaluable. Knowing that they were taking care of themselves gave me the peace of mind to focus wholeheartedly on my research. I am truly blessed to have such dedicated and selfless parents. I dedicate this achievement to them with heartfelt appreciation.

I would also like to express my heartfelt thanks to Priyanshu Raj Mall, Srujana Mohanty, Abhisek Kar, Aditya Dutta for their steadfast support and time. I am also grateful to the Indian Institute of Science Education and Research Kolkata for providing the necessary resources and facilities to complete this project to the best of my ability. To everyone mentioned above and those who may not be explicitly named but have contributed in various ways, I offer my heartfelt gratitude and deepest respect. Your contributions, whether big or small, have made a significant impact on this thesis, and for that, I will forever be grateful.

Kolkata - 741246 **Aman Sinha**

May 2023

Abstract

The rapid digitization of information has led to an unprecedented accumulation of knowledge across various formats, making effective information retrieval essential for accessing and utilizing this vast pool of information. In the field of Information Retrieval (IR), search engines like Google, Bing, and DuckDuckGo play a pivotal role in influencing our daily lives by presenting us with search results. With this influence comes the responsibility to ensure unbiased retrieval of websites and documents. Retrievability, a quantitative measure capturing a document's ability to be retrieved by a retrieval model regardless of the search query, plays a crucial role in assessing retrieval effectiveness.

This master's thesis investigates critical aspects of IR systems, aiming to identify flaws, propose improvements, and uncover biases affecting their performance evaluations. The research begins by exposing a major flaw in the artificial query generation method used for Retrievability analysis. Building upon this finding, an improved method is proposed, demonstrating enhanced correlation for accurately assessing document retrievability. The study further uncovers a new bias in the Relevance Judgement process, which favors highly retrievable documents, potentially distorting the evaluation of IR systems. Recognizing this bias is crucial for fair assessments, and future research should focus on mitigating its impact. An examination of the RM3 technique reveals its ability to boost overall performance but at the cost of making unique relevant documents less findable. Balancing retrieval effectiveness with the retrieval of unique and relevant documents becomes a vital consideration for practitioners and researchers. Additionally, a slight correlation is observed between document ranks from PageRank and Retrievability measures, suggesting a relationship between the two metrics. This finding opens avenues for exploring their interplay and mutual reinforcement in future research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Throughout the ages, civilizations have generated and disseminated information through diverse means and mediums. As time elapsed, the volume of information produced has increased and has been preserved through a range of storage media. An enduring difficulty in the management of information has been how to locate relevant information that meets one's informational needs. One way libraries have tackled this challenge is by utilizing a categorical system, such as the Dewey Decimal Classification system introduced in 1876 [Dewey, 1876], which enables information seekers to track down pertinent information by navigating a series of categories. Nevertheless, the exponential increase in information generation, particularly in the last couple of decades, has rendered this traditional method of information categorization insufficient for effectively organizing and retrieving the vast amounts of information available.

This new challenge of finding relevant information led to the emergence of the field of Information Retrieval (IR), which aims to organize, structure, and provide access to information to facilitate searchers in locating the information they need. The value of well-documented and informative content is negligible if it cannot be retrieved promptly and accurately when required. In contemporary times, information is available in various formats beyond textual information, and IR research has expanded to encompass these diverse information types. Nevertheless, textual information, including webpages, news articles, and patents, is still the most commonly sought-after form of information. Therefore, systems were developed to handle the large amounts of currently available information.

The emergence of initial Information Retrieval Systems (IR Systems) introduced a new paradigm of storing documents that eliminated the need for strict categorization for efficient retrieval. Instead, users could input a few keywords as a query, and the IRS would retrieve a set of documents likely to contain the information relevant to the search [Belkin, 1980; Sanderson, 2008]. An IR System comprises two main components for retrieval: an index that represents the corpus and a retrieval algorithm that has several parts. The first retrieval algorithms were based on boolean retrieval [Van Rijsbergen, 1979]. However, these algorithms returned a vast set of documents, making it impractical to examine each document in the set. To tackle this issue, document ranking was introduced [Salton and Yang, 1973], where documents were scored based on their relevance to the query. A simple approach to scoring is to count the number of query term occurrences in each document [Roelleke, 2013]. The document with the most occurrences of the query terms was likely the most relevant and thus presented to the searcher first. This way, the user would be able to swiftly locate the most pertinent document in the top position, and subsequent documents were considered less relevant to the given query. This search paradigm facilitated access to vast amounts of information, as users could locate pertinent documents from vast collections by entering a few keywords. As the sophistication of ranking systems increased, the responsibility for users to construct effective queries diminished, with algorithms assuming the task of inferring users' information requirements and prioritizing the most pertinent documents accordingly.

A significant portion of IR research has focused on assessing the efficiency and performance of the proposed retrieval algorithms for IR systems. The Cranfield evaluation approach [Cleverdon, 1991] became a standard method to evaluate an IR System. This eventually led to the beginning of the Text Retrieval Conference (TREC), a US government agency dedicated to creating test collections to facilitate the evaluation of IR Systems [D. K. Harman, 1993]. This structured and systematic approach to evaluation led to a surge in IR research, resulting in the development of novel retrieval algorithms that were then tested and evaluated in a standardized manner, allowing for comparison with state-of-the-art retrieval algorithms available at that particular period. However, as retrieval and ranking algorithms became more effective, it became increasingly difficult to fully understand how these algorithms score documents and how the associated parameters affect the final document score. For example, BM25 [Stephen E. Robertson et al., 1992] has multiple parameters in its ranking function, and without a thorough investigation of the mathematical underpinnings of the algorithm, it is challenging to discern how each parameter influences the final score of a document. This issue became even more problematic when the size of the collections which is used for evaluation grew, making it impossible for the researchers to make informed judgments by manually analyzing the results.

The assessment of IR system efficiency focuses on critical aspects such as CPU usage, memory usage, and query completion time. These factors are fundamental in determining the utility of a system, as high performance levels may be compromised by excessive resource consumption or prolonged query completion times, which limit its practical applications. Therefore, the evaluation of IR systems considers both efficiency and performance criteria. In recent times, a new evaluation criterion has emerged, namely, the retrievability bias associated with retrieval algorithms. This evaluation approach seeks to determine whether a retrieval algorithm introduces any biases in the selection of document irrespective of the query posed.

The issue of bias is widespread and prevalent across many domains. Recently, search engines have emerged as a significant concern owing to the proliferation of politically aligned entities, dissemination of fake or malicious news, and other surreptitious agendas. The manifestation of search bias can take various forms, including deliberate bias such as political alignment, and unintentional bias such as algorithmic bias. The detection of intentional biases can be accomplished by comparing the search results of various systems over time [Mowshowitz and Kawaguchi, 2005]. However, such biases are challenging to counteract as they are designed deliberately. The involvement of relevant authorities may be necessary to mitigate them. In the context of IR, biases can sometimes be both positive or negative. For instance, PageRank [Brin and Page, 1998] introduces a bias towards highly linked documents, thereby favoring popular pages and boosting performance. While retrieval algorithms are constructed to be biased or show preference towards relevant documents, there is a possibility of unintentional biases arising that do not assist in distinguishing between relevant and irrelevant documents.

Algorithmic bias denotes an inclination of an algorithm to prefer certain documents or groups of documents at the expense of others, which results from document characteristics that do not have a direct bearing on the relevance of the document corresponding to the query. It is important to note that this definition of algorithmic bias excludes intentional biases programmed into algorithms, such as PageRank [Brin and Page, 1998]. Algorithmic bias pertains specifically to the inadvertent negative biases that compromise the algorithm's ability to discern relevance, for instance, a bias favoring longer documents as they tend to contain a greater number of terms [Sparck Jones, 1972]. This kind of bias can impede the search system's performance as less relevant documents may receive higher scores owing to the algorithmic bias embedded in the retrieval algorithm. As a result, there may be a decline in user satisfaction, as searchers would need to exert extra effort to find the pertinent information, either by modifying their queries or exploring the ranked list more thoroughly.

1.1 Motivation

Although evaluating bias in information retrieval (IR) systems is a relatively new idea [Azzopardi and Vinay, 2008a; Azzopardi and Vinay, 2008b], the IR community has long recognized the potential of algorithmic bias to affect retrieval algorithms, dating back to the earliest models [Sparck Jones, 1972]. While biases were not explicitly measured, the mathematical foundations of the retrieval algorithm often exhibited the potential for bias to manifest in the function. One example of this was TF.IDF, which tended to favor longer documents, leading to retrieval of irrelevant, lengthy documents. To counter this, modifications were made to the algorithm [Roelleke, 2013], resulting in Pivoted TF.IDF [Singhal et al., 1996], which permitted users to adjust the degree of length normalization applied to document scoring, thereby mitigating the length bias and enabling the algorithm to be utilized on a wide range of document collections.

Retrievability was presented by Azzopardi and Vinay as a method for evaluating the degrees of bias existing in a search system. The researchers [Azzopardi and Vinay, 2008b] emphasized that any component of the retrieval procedure could introduce bias, ranging from the retrieval algorithm to the collection itself. In essence, retrievability examines each document within the collection based on its ease or difficulty of retrieval, given a specific search system. The researchers initially suggested retrievability using a transportation planning analogy. Picture yourself at a central transport hub in a city, from which buses, trains, and other transport means travel to numerous destinations. This symbolizes the search system, and as a searcher, you can access it. You have a target location in mind, so you must first determine whether it's feasible to reach or come close to it using the available transportation options. This is comparable to a searcher with an information requirement and access to an Information Retrieval System (IRS). The searcher can interrogate the system using an array of terms that best represent their requirements. Each route in the transportation planning can be viewed as a search that a searcher might initiate. You may opt to take the train to your destination initially and then transfer to a bus later in the journey. This is akin to a searcher submitting a query and then modifying that query based on the newly acquired information. In the end, you'll either arrive at your destination, abandon the attempt, or change to a new destination that is more accessible. Searchers do the same by either locating the information they need, giving up, or searching for new data. In the transportation planning analogy, some destinations are easy to reach, while others are incredibly difficult or impossible to access. Similarly, some documents in the collection are simple to retrieve, while others are extremely challenging or unattainable through the IRS provided to a searcher. The searcher can employ various strategies to find pertinent information, but their level of control remains ultimately constrained by the IR System which they are using.

Since the introduction of the retrievability measure, numerous studies have been conducted to explore the degree of retrieval bias exhibited by various retrieval models. Query expansion (QE) is a popular technique widely employed to enhance the performance of information retrieval systems by appending supplementary terms or concepts related to the user's initial query. Nevertheless, the extent to which query expansion affects retrieval bias is not yet comprehensively understood, and additional research is necessary to ascertain whether query expansion can alleviate or exacerbate retrieval bias. Although only two query expansion models have been analyzed for bias in a preliminary investigation [Wilkie and Azzopardi, 2017], this thesis intends to reproduce the results on several standard retrieval models and conduct retrievability analysis on RM3, which is the most effective QE method in the classical (non-neural network) domain of information retrieval. The *Fairness Hypothesis* [Wilkie and Azzopardi, 2014] is still under debate and QE is one such scenario where the hypothesis was observed to collapse for the studied two QE methods. By studying the scenario for the RM3 QE method, this investigation contributes to expanding and generalizing the observation to more QE methods, providing a better understanding of the impact of QE on term selection from a retrieval bias perspective.

An important concern in the realm of information retrieval is related to the process of document pooling, which is used for generating relevance judgments. Generally, this process entails the gathering of documents that have obtained high rankings from multiple retrieval models for a given query [D. Harman, 1993]. These documents are considered to represent the possible pool of the most pertinent documents, based on the efficacy of the utilized retrieval models. Subsequently, assessors evaluate solely these documents, which form the basis for the evaluation of effectiveness in various workshops, such as TREC, CLEF, NTCIR and FIRE. Thus, relevance judgment serves as a critical element in objectively assessing the performance of different search models. Nevertheless, a potential issue arises when the pool of documents employed in the relevance judgment is biased. If the retrieval models used in the pooling process exclude relevant but less retrievable documents from the pool, then the evaluation of retrieval models based on such relevance judgments could unfairly penalize models that retrieve these documents. As a result, retrieval models that frequently return significant but less retrievable documents at the top ranks may exhibit lower performance in evaluations based on relevance judgment, and retrieval models that agree with the biases of the retrieval models utilized in the document pooling process may receive higher performance metrics due to their better overlap with the relevance judgment documents. To investigate if document pooling bias is actually there or not, this thesis examines the distribution of retrievability scores for the judged documents as compared to the remaining documents. Any tendency of the score distribution of the judged documents towards higher scores would indicate bias in the document pool towards more retrievable documents.

Retrievability analysis relies on an accurate estimation of a large subset of all possible queries that are likely to be posed by users [Azzopardi and Vinay, 2008b]. An ideal approach would be to utilize a subset of the search log of a search system over the target corpus since it contains all the possible queries that people are interested in regarding the corpus. However, this search log is often unavailable for retrievability analysis. As a result, an artificial query generation method based on the Query Based Sampling approach is used, where N-grams are sampled from corpus documents for various N. Most subsequent research that explores or employs retrievability has used the artificial query generation method to create query sets for retrievability analysis, rather than a real query log. However, it remains unclear whether retrievability scores from an artificial query set constitute an acceptable approximation of retrievability scores from a real query set sampled from a query log. If the correlation between the two is found to be low, then a question arises regarding which query set provides a more accurate estimate of the retrievability bias of the IR system. Consequently, the validity of previous works that apply retrievability analysis blindly to estimate retrieval bias without a correct query set could come into question. Therefore to find out, this thesis aims to determine the correlation of retrievability scores for documents using both real and artificial query sets to investigate the robustness of the measurement against variations in the query set used for retrievability analysis. Furthermore, an attempt is made to propose an improved query generation method that enhances the correlation between the scores obtained from artificial queries and those obtained from real queries.

The retrievability score of a document could be regarded as an accumulation of the number of queries that the document could potentially answer, which implies that the retrievability score reflects the usefulness of a document. Another measure of usefulness, in terms of popularity, is the PageRank algorithm proposed by Brin and Page [Brin and Page, 1998]. PageRank is utilized to evaluate a webpage's significance or pertinence by considering the quantity and caliber of links that refer to it. The effectiveness of PageRank is due to its capacity to rank pages based on their credibility and dependability, which has considerably advanced the precision and suitability of search engine outcomes. As both measures aim to capture the usefulness of documents, it is expected that they should be somewhat correlated, since many documents will be useful in terms of both information coverage and popularity. To investigate this, the extent of correlation between PageRank scores and Retrievability scores over a corpus is computed. PageRank computation of a document requires interlinking between the documents, which forms a graph structure. Wikipedia is a large corpus with documents on diverse topics, and articles refer to other articles within the Wikipedia, making it a rich source for interlinking between the documents. WT10g [Bailey et al., 2003] is a standard TREC dataset that also contains link information. The two corpora, Wikipedia and WT10g, are chosen for the experiment. If the correlation between PageRank and Retrievability scores is weak, then the combination of both scores can be tested to demonstrate boosted effectiveness when used as prior document weights in the retrieval model's scoring function. To evaluate the performance, the relevance judgment present in the WT10g dataset is utilized. By conducting this study, insights could be obtained on how Retrievability and PageRank scores could be used to improve retrieval performance, which could potentially benefit IR systems.

1.2 Goal

The aim of this thesis is to investigate the following:

1. Study the impact of query expansion on retrieval bias by comparing the retrieval bias of standard retrieval models and a baseline query expansion technique using *retrievability* analysis.
2. Investigate the presence of any bias in the document pooling process for creating relevance judgments based on the distribution of retrievability scores of documents.
3. How much correlation exists between retrievability scores obtained from retrieval simulations using real query logs versus those generated artificially?
4. Improve the artificial query generation method to more accurately emulate real-world queries.
5. What is the extent of correlation between PageRank scores and Retrievability scores of the articles in the Wikipedia?

1.3 Outline

This report is organized as follows:

- Chapter 1 presents the motivation for this thesis, the goals and research questions that it poses, and outlines the structure of this thesis.
- Chapter 2 briefs the necessary background knowledge to the field of Information Retrieval (IR) to understand the works and discussions that will follow in the subsequent chapters.

- Chapter 3 introduces the concept of retrieval bias, measures that attempt to quantify such biases, and then presents in detail the Retrievability analysis, which is one of the general methods to assess retrieval bias.

- Chapter 4 reproduces prior retrievability analysis results on standard retrieval models using a different corpus. Then it studies the retrieval bias imposed by a baseline state-of-the-art query expansion technique in contrast to its performance gain. It also investigates the retrievability score distribution of judged documents in the relevance judgment of the corpus and tries to highlight the bias in the pooling of documents for judgment.

- Chapter 5 performs a retrievability analysis, which involves the examination of two corpora using distinct query sets and the utilization of the BM25 retrieval model. The primary objective is to compare the retrievability scores obtained from artificially generated queries with those derived from a query log, with the aim of assessing the viability of artificial queries for retrievability analysis. The research findings indicate a lack of correlation between the retrievability scores obtained from artificially generated queries and those derived from the query log. Additionally, the proposed modified approach demonstrates a higher correlation with the query log-based retrievability scores, suggesting its potential as a more suitable method for query generation in situations where a historical query log is unavailable for retrievability analysis on IR systems.

- Chapter 6 explores the relationship between PageRank and Retrievability by analyzing two distinct document collections, Wikipedia and the TREC Web Corpus. It presents the methodology employed to extract in-links and out-links information from these corpora, as well as the computational challenges associated with performing PageRank computations on large datasets. The chapter will also present the PageRank algorithm used, along with the steps taken to address the aforementioned challenges. The chapter concludes with observations on the correlation between PageRank and retrievability, contributing to a deeper understanding of link analysis algorithms and information retrieval in large-scale document collections.

- Chapter 7 concludes the thesis. This thesis has addressed critical issues in the field of IR systems, uncovering limitations and proposing improvements. The flaws in artificial query generation for Retrievability analysis and the bias in Relevance Judgement were identified, calling for more robust approaches and fair evaluations. The impact of employing the RM3 technique was explored, highlighting the trade-offs in retrieval effectiveness. Furthermore, a correlation between PageRank and Retrievability measures suggested potential interplay. These findings underscore the complexity of IR systems and the need for ongoing research. Future work should evaluate the proposed improved method, address bias, assess the impact of RM3 on finding unique relevant documents, and explore the integration of retrievability measures into ranking algorithms. These efforts will enhance search result quality and advance the effectiveness of IR systems.

Chapter 2

Information Retrieval Fundamentals

This chapter presents a comprehensive examination of the foundational knowledge that readers should possess in order to comprehend the contents of this report. Its primary objective is to provide a concise summary of information retrieval (IR) systems, retrieval models, and evaluation methodologies, which constitute crucial elements for comprehending the subsequent chapters' content and discussions.

2.1 Information Retrieval System

Information Retrieval (IR) is an academic field that has a long-standing and well-established history, dating back to the late 1940s. At its inception, a small group of professionals were involved, with the primary objective of retrieving unstructured text. The term "document" was frequently used interchangeably with "information" to describe this process [Van Rijsbergen, 1979]. However, the emergence and increasing popularity of the World Wide Web in recent decades has transformed the landscape of IR. Currently, millions of people worldwide utilize IR daily, surpassing traditional methods of information access, including structured database searches. IR has broadened its scope to encompass retrieving data beyond plain text, such as multimedia archives.

This development has generated a growing interest in IR as an academic field, prompting researchers to revisit and enhance existing methods, propose novel ones, and modify the representation and structure of information to enhance automatic retrieval. The overarching objective of research in this field is to develop superior methods to achieve the same objective, with evaluation playing a critical role in the process [Manning et al., 2008].

2.1.1 Definition

Baeza-Yates and Ribeiro-Neto's [Baeza-Yates, Ribeiro-Neto, et al., 1999] description provides a commonly accepted explanation of what IR is, which is as follows:

> *"Information retrieval (IR) deals with the representation, storage, organization of, and access to information items. The representation and organization of the information items should provide the user with easy access to the information in which he is interested."*

Another, a narrower, definition of IR is given by [Manning et al., 2008]:

> *"Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)."*

Data Retrieval versus Information Retrieval

The fields of data retrieval and information retrieval in computer science are closely linked, and to further clarify the definition of information retrieval, their distinguishing characteristics and differences are presented.

A data retrieval system employs a synthetic language with a limited vocabulary and syntax to retrieve data that has a clearly defined structure and semantics. The query must provide a complete specification of the desired data, and even a small mistake can result in a total failure of the process, as the system aims for total matching where the document must satisfy every part of the query. Such a system is suitable for a database where users can enter specific key terms like an identification code. However, it is unable to address the issue of retrieving information in a satisfactory manner, where the query only partially matches the document.

In contrast, information retrieval systems typically use natural language text for the query, which may be incomplete. The primary objective of an information retrieval system is not only to extract information from unstructured data but also to evaluate its relevance to the query. This is necessary because, unlike database searches, information retrieval systems do not have clear-cut hits or misses. Ideally, such a system should retrieve all relevant documents while minimizing the number of irrelevant ones, enabling users to quickly browse through a list of partly matching ranked results and select those that meet their information needs [Baeza-Yates, Ribeiro-Neto, et al., 1999; Belew, 2000].

2.1.2 IR Tasks

The principal task in IR is to provide relevant documents from a collection that meet a user's undiscriminating information needs, commonly called *ad hoc retrieval*. The subject matter of *information need* is the area of interest that the user wishes to gain further knowledge about. The user conveys their requirements to the system by issuing a one-time query that they *believe* accurately reflects their needs. A document returned by the system is considered relevant if the user perceives it to contain information that is pertinent to their initial information need [Manning et al., 2008].

Other tasks in IR include the retrieval of multimedia content such as speech, music, images, and videos. In *cross-language* IR, the system attempts to retrieve results in multiple languages for a query in one. Furthermore, routing, filtering, and additional processing of already retrieved sets of information are also essential tasks [Manning et al., 2008]. However, this thesis primarily focuses on the ad hoc retrieval task for documents.

2.1.3 Ad Hoc Retrieval

The fundamental and standard interaction stages and elements of an ad hoc IR system are depicted in Figure 2.1.

To begin with, it is essential to acquire all documents that will be indexed. The initial step of this process involves pre-processing the document collection,

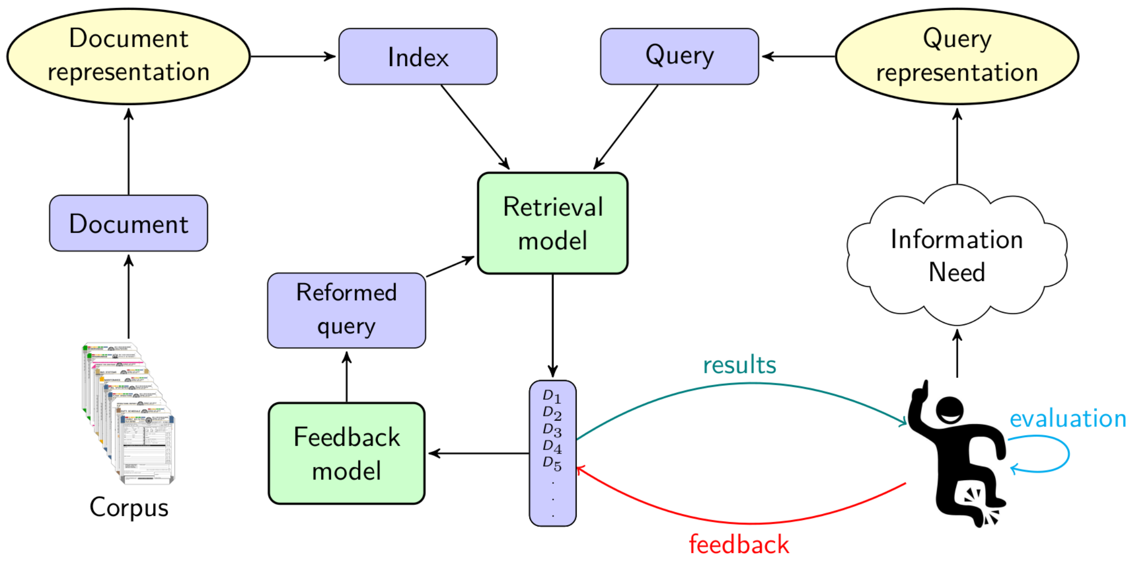


Figure 2.1: Graphical Representation of IR Process from [Roy, 2022]

which involves converting it into a format that the indexer can interpret. The indexer subsequently dissects the documents into discrete units known as tokens and may undertake certain normalization techniques such as case-folding, stop-word elimination, equivalence classing, stemming, and lemmatization. These tokens are then incorporated into the index.

After completing those steps, a user can initiate a search for documents to satisfy their information needs. However, as the imprecise nature of the user's query cannot be directly understood by the IR system, the user must attempt to articulate their information need via a query and submit it to the system. Typically, the same normalization techniques employed for the documents are also utilized for the query to enable accurate matching. Subsequently, the IR system retrieves documents that partly match the query. With the aid of retrieval models, these documents can be ranked based on their similarity to the query. Ultimately, a ranked list of documents is returned to the user. Optionally, some IR systems offer the possibility of refining the list of results by providing relevance feedback [Baeza-Yates, Ribeiro-Neto, et al., 1999, pp. 9-10].

2.2 Indexing

Performing linear scans on documents is the most fundamental approach to document retrieval, which is also referred to as grepping. This technique is named after the Unix command grep. While this method suffices for simple querying on small collections, it presents several limitations [Manning et al., 2008, p. 19]:

- The efficiency of this method relies on the size of the document collection, as its performance diminishes with the collection's growth.
- This method is not suitable for complex query matching operations.
- Ranked retrieval is not possible with this method, which is essential for obtaining the most relevant answer to an information need.

To avoid the laborious task of searching through all documents for every query, it is commonly recommended to *index* them beforehand. One approach to accomplish this is by constructing a term-document matrix, wherein each document is assigned a unique documentID and the mapping is recorded in a lookup table. The rows and columns of this matrix contain vectors that represent the terms present in each document and the documents that contain each term, respectively. However, constructing such a matrix for large collections with a vast vocabulary results in a matrix with bulky dimensions. This is primarily because the matrix includes an abundance of redundant information, such as document-term combinations that do not exist and must be marked with 0's.

To address this issue, a solution to build an *inverted index*. Witten et al. [Witten et al., 1999, p. 109] suggest that the inverted index is the most efficient method for text applications. The inverted index only stores term-document combinations that actually occur and consists of two essential components: a dictionary and postings. The dictionary, also called a lexicon, encompasses all the terms in the collection. Each term has a corresponding list that contains information about the documents in which the term appears, and the position of the term in the document can be optionally stored. This information also automatically indicates the frequency of the term, as the number of positions can be summed to obtain that information. Each item in such a list is referred to as a posting, and the combination of all lists is called postings. The fundamental concept of the inverted index is illustrated in Figure 2.2.

In order to create an inverted index, four main steps are required as described by [Manning et al., 2008, p.19]:

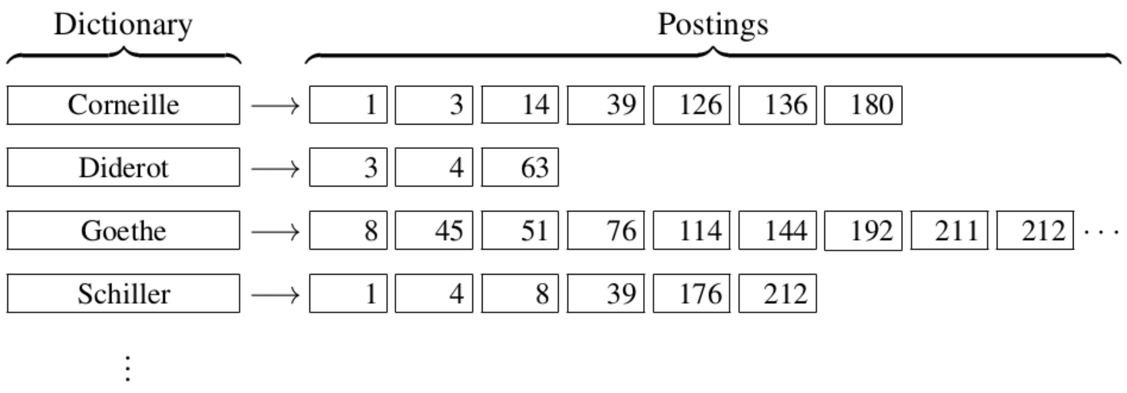


Figure 2.2: Inverted index schematic adapted from [Manning et al., 2008, p. 7].

- Collection of all documents that require indexing
- Extraction of tokens from the text of each document to create a token list
- Linguistic pre-processing of these tokens to create indexing terms
- Indexing of the documents for each term to create the inverted index

To enhance the optimization of the process, termIDs can be used instead of the terms in the dictionary. This involves assigning a unique number to each term and creating a mapping between the terms and their corresponding termIDs. This approach is similar to the one used for documents.

2.2.1 Tokenization

Tokenization is the process of dividing a document into smaller semantic units known as tokens. Tokens are composed of character sequences from the original document and are useful for subsequent processing. A group of tokens that contain the same character sequence is referred to as a type.

During tokenization, certain characters, such as punctuation marks, can be omitted, while some character combinations present challenges for the tokenization process. For instance, apostrophes in English have diverse semantic meanings, including word contractions and possession indications. Similarly, hyphenation is used to group words, bond nouns, or split up vowels in words. These challenges highlight the significance of language familiarity to the tokenizer, which may utilize classifiers to recognize languages based on character subsequences. Additionally, certain character sequences, such as multi-word city names, programming languages like C++, and URLs for web pages, must be treated as a single token.

German, for example, poses a different challenge where compound nouns are written without spaces, necessitating the use of a *compound-splitter* to separate them into individual tokens. This issue is further compounded in some Asian languages such as Chinese, Korean, and Japanese, where text is written without white spaces to separate words, and thus, word segmentation must be performed first. However, this process can introduce errors since word boundaries are not always apparent [Manning et al., 2008, pp. 22-28].

2.2.2 Normalization

Prior to the incorporation of tokens into the dictionary of an IR system, it is common to apply several normalization techniques to eliminate surface-level disparities and amalgamate multiple tokens into a unified form. This practice often enhances the accuracy of query results by facilitating better matches between user queries and the indexed data. Typically, the normalization methods used in index generation are also employed in query processing to achieve this objective.

Case-folding

One frequently utilized normalization method is the process of transforming all letters to lowercase, referred to as “case-folding.” This technique results in tokens with capital letters at the start of a sentence being altered to the same form as they would have in the middle of a sentence. However, implementing this approach carries some drawbacks, such as the loss of information. In particular, distinguishing between common and proper nouns becomes impractical, and acronyms may merge with a word of similar spelling but an entirely distinct meaning. To circumvent this information loss in English texts, a viable alternative is to utilize true-casing. This approach only converts tokens to lowercase when they are at the beginning of a sentence or are entirely or primarily in uppercase, such as in a headline. Nevertheless, given that most users do not prioritize proper capitalization and submit their queries in all lowercase, case-folding is typically the optimal solution [Manning et al., 2008, p. 30].

a	is	an	it	and	its	are	of	as	on	at	that	be
the	by	to	for	was	from	were	has	will	he	with	in	

Figure 2.3: 25 commonly used words from the Reuters-RCV1 corpus [Manning et al., 2008, p. 26].

Stop Word Removal

Frequently used terms may not effectively aid in the selection and differentiation of relevant documents. Such terms are known as stop words, since the text processing ceases upon encountering them, and they are not incorporated into the dictionary. Typically, the creation of a stop list involves ranking the terms based on their frequency of occurrence in the document collection and appending the most prevalent ones to the list. Collection frequency denotes the total number of instances in which a term appears throughout the document collection. Additionally, it is often necessary to modify the stop list manually to tailor it to the semantic field of the documents. The utilization of a stop list has the benefit of significantly reducing the number of postings required for storage in the index [Croft et al., 2010, p. 64]. Figure 2.3 illustrates an example stop list of commonly used words in English texts.

In most cases, the absence of stop words from the index has a negligible impact on search queries, as long as the stop list is thoughtfully created and a minimal number of words are excluded. Nevertheless, there are exceptions, such as when searching for song titles or verses or executing phrase queries. These searches may consist of very common words, causing critical parts to be omitted. For instance, if the example stop list in figure 2.3 is applied to the renowned soliloquy from William Shakespeare's play, The Tragedy of Hamlet, Prince of Denmark, which includes the line "To be or not to be," the search would solely involve the terms "or" and "not."

In IR systems, stop lists historically comprised a sizeable number of terms, ranging from 200 to 300. Presently, it is more common to employ brief lists of approximately ten stop words or none at all, as modern systems seldom encounter performance or storage issues [Manning et al., 2008, p. 27].

Stemming and Lemmatization

The purpose of both stemming and lemmatization is to transform variant forms of a term into a standardized base form. *Stemming* involves a less sophisticated but faster technique, whereby the ends and derivational suffixes of terms are truncated in an effort to achieve this goal. The most widely utilized stemming algorithm for English texts is the Porter Stemmer, which utilizes a series of rules in five sequential phases to abbreviate the term. Whereas, *Lemmatization* entails a more comprehensive morphological analysis of the term, which results in the return of the lemma, or base form, typically found in a dictionary. Although lemmatization is a more intricate process, it usually does not result in substantial enhancement in information retrieval performance for IR objectives when compared to the simpler approach of stemming. It is important to acknowledge that the effect of both methods on query performance is not uniform and can differ depending on the nature of the queries. Some queries may gain benefits from the use of these methods while others may not.

Equivalence Classing

Following the stemming process, certain tokens are merged since they share the same meaning. However, there are instances where two tokens have some slight differences but should still be considered a match. For instance, a user searching for 'USA' might also want results that include 'U.S.A.', which only differs in superficial characters. To resolve such inconsistencies, *equivalence classes* are created, usually named after one of the tokens within the set. This enables retrieval of documents containing any member of the set when searching for one of the terms. The creation of mapping rules that eliminate specific characters allows for the easy creation of equivalence classes. Nonetheless, determining when to add characters to these rules can be ambiguous since the rules are implicit.

Alternatively, one can utilize synonym relationships between tokens, which are manually curated in a list and can be expanded. During index creation, if a document contains the term 'lift', it can also be indexed as 'elevator'. Alternatively, the tokens can be indexed as they are, and the synonym list considered when processing query terms. This method is commonly used and elaborated in section 2.5, which deals with query expansion [Manning et al., 2008, p. 28].

2.3 Retrieval Models

The effectiveness of information retrieval systems is largely dependent on the type of retrieval model employed. In this section, an overview of the prominent retrieval models is provided, along with an explanation of the key term weighting techniques that they utilize.

2.3.1 Boolean Retrieval

Boolean Retrieval is the earliest and most commonly used model for information retrieval. Despite decades of academic research promoting ranked retrieval systems, commercial information providers exclusively implemented Boolean Retrieval until the early 1990s. In *Boolean model*, binary weights are assigned to index terms, which are either present in a document (assigned a binary value of 1) or absent (assigned a value of 0). This model disregards other factors, making it impossible to grade the relevance of documents. Queries in this model use simple semantics and employ operators such as 'AND', 'OR', and 'NOT' to connect index terms.

However, this simple approach has a significant disadvantage of relying on exact matching, resulting in binary decisions when retrieving documents. This means that the number of retrieved documents is solely dependent on the query and can lead to an overwhelming number of results that exceed the user's capacity to review comprehensively. Moreover, the aforementioned operators are often inadequate for complex information needs, making it challenging to express them accurately. As a result, *extended Boolean models* were developed, which incorporate operators that consider the proximity of terms [Baeza-Yates, Ribeiro-Neto, et al., 1999, pp. 25-27], [Manning et al., 2008, pp. 14-15, 109].

2.3.2 Vector Space Model

This approach acknowledges the limitations of Boolean Retrieval and presents a framework that enables partial matching and the computation of relevance grades by identifying documents with the highest similarity corresponding to the query. This approach generates a ranked list of results, unlike Boolean Retrieval.

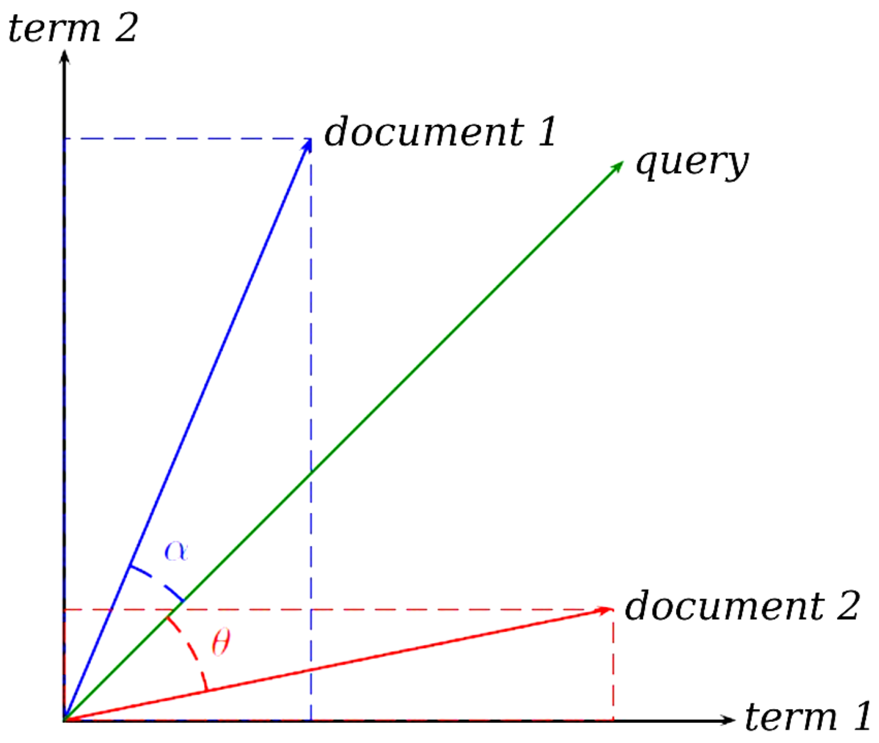


Figure 2.4: Vector Space Model illustrated.
(Image adapted from: Wikipedia, 2023c)

The fundamental concept of the Vector Space Model involves representing each document as a vector in a multidimensional vector space, where each dimension corresponds to a term in the dictionary. The presence or significance of terms is represented using coefficients, which can be binary values or non-binary term weights that are commonly scaled. Queries are also represented in the same vector space as vectors, usually as a list of terms without the use of a query language like Boolean retrieval models. Figure 2.4 illustrates this concept in a two-dimensional vector space with two terms, where document 1 has the highest similarity to the query.

There are two common techniques for determining the similarity of two documents. The first involves calculating the distance between the endpoints of the respective document vectors. However, this approach has a shortcoming, as it can yield significant distance values for documents that share highly similar content but differ greatly in length. The second method, which is commonly used, is cosine similarity represented by the cosine value of the angle between the query vector $\vec{q}$ and document vector $\vec{d}$ in the following equation:

$$similarity(q,d) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}||\vec{d}|} \tag{2.1}$$

Equation 2.1 is comprised of a numerator that denotes the inner product of two vectors, and a denominator that is the result of the product of their Euclidean norm. The denominator normalize the length of each vector, which addresses a limitation of the initial approach as mentioned in [Turtle and Croft, 1992], [Baeza-Yates, Ribeiro-Neto, et al., 1999, pp. 27-30], [Manning et al., 2008, pp. 120-121].

TF

The straightforward method of assigning a scaled weight to individual terms within a document involves equating their weight with the frequency of their occurrence in said document. This method is commonly known as *term frequency* (TF) and is expressed as $tf_{t,d}$ where t represents the term and d represents the document. Under this scheme of weighting, the ordering of terms is unimportant (referred to as a *bag of words*), and all terms are treated equally with respect to their contribution to the relevancy of a query. However, certain terms have limited or no ability to distinguish between different documents, thereby rendering this approach problematic due to its uniform weighting scheme.

TF-IDF

It is common to use a variation of tf (term frequency) called *term frequency-inverse document frequency* (tf-idf) weight, rather than raw term frequency. This is because some terms are frequently occurring and should be considered as such. To address this, the inverse document frequency is used to decrease the weight of common terms throughout the entire collection and increase the weight of rare terms. The calculation for tf-idf weight for a term t in a document d is shown in equation 2.2. The total number of documents in the collection is represented by N, and the total number of documents containing the term t is represented by df_t.

$$tfidf_{t,d} = tf_{t,d} \times log\frac{N}{df_t} \tag{2.2}$$

However, the standard tf-idf method can be biased towards large absolute term frequencies because it does not normalize them relative to the document length. To mitigate this issue, normalized tf-idf uses the document length $|d|$ [Manning et al., 2008, pp. 117-119].

2.3.3 Probabilistic Models

When utilizing either the boolean or vector space model for IR, the system possesses an imprecise semantic calculation of index terms. Consequently, with only a single query, the system can merely provide a conjecture as to whether a document contains content that is germane to the user's information need since the IR system's comprehension of the information need is indeterminate. To address this issue, probability theory can be utilized for reasoning under uncertainty to assess the probability of a document being pertinent. Several retrieval models, such as the Binary Independence Model and the Okapi BM25 Model, are based on probabilistic principles and are exemplified in [Manning et al., 2008, p. 219].

Binary Independence

The *Binary Independence model*, being one of the oldest probabilistic retrieval models, employs binary term weights, which are equivalent to Boolean values. Binary vectors of the form $\vec{x} = (x_1, ..., x_n)$ represent both the queries and the documents. If a document contains a specific term t, then x_t is set to 1, leading to a reduction of many potential documents to the same representation. This model assumes that terms occur independently within documents, disregarding any association between them. Although this assumption is incorrect, the model still performs adequately on most collections, although it has not been adapted for modern full-text retrieval [Manning et al., 2008, pp. 222-223].

Okapi BM25

The *Okapi BM25* is a non-binary model that utilizes BM25 as its weighting scheme. It was created with the objective of developing a probabilistic model for full-text retrieval that would consider term frequency and document length while minimizing the number of additional parameters [Jones et al., 2000]. The model was introduced in the 1990s and quickly gained popularity, particularly in many TREC test collections, as it demonstrated excellent performance. Additional information on TREC can be found later in the chapter. The Okapi BM25 is currently one of the most commonly used models, and various versions of the formula exist. One widely utilized variant is presented in equation 2.3.

$$\text{BM25}(q,d) = \sum_{t \in q} log \left[\frac{N}{df_t} \right] \frac{(k+1) \cdot tf_{t,d}}{k \left((1-b) + b \cdot (\frac{L_d}{L_{ave}}) \right) + tf_{t,d}} \tag{2.3}$$

The formula presented includes two tuning parameters, b and k, along with two document length metrics. L_d is the document length, while L_{ave} is the average document length of the entire collection. The value of b can range from 0 to 1, with 0 representing no scaling of document length, and 1 indicating full scaling of the term weight by the document length. Meanwhile, k scales the document term frequency, with 0 indicating no scaling, which corresponds to a binary model. It is important to note that the range of k extends from 0 to infinity [Manning et al., 2008, pp. 232-234].

2.3.4 Language Models

The typical strategy for a user to formulate an effective query involves expressing their information need using language likely to appear in relevant documents. This fundamental approach is directly implemented by *language models*, which build a probabilistic language model M_d for each document d instead of modeling the probability of relevance of a document to the query. The documents are then ranked based on the probability $P(query|M_d)$ of generating that query. To accomplish this, probability values must be assigned to language terms. Various models exist to achieve this by constructing probabilities over sequences of terms, such as the unigram, bigram, and multinomial unigram models. A language model over A can be represented by a function that places a probability measure over strings s, where the total probability always equals 1 [Manning et al., 2008, pp. 237-238]:

$$\sum_{s \in A} P(s) = 1 \tag{2.4}$$

Unigram

The unigram language model described in equation 2.5 disregards any contextual cues and the arrangement of terms within the text, thereby functioning as a "bag of words" model. Its basic nature is attributed to the fact that it estimates each term t individually, rendering it the most elementary form of a language model [Manning et al., 2008, pp. 117, 240].

$$P_{unigram}(t_1 t_2 t_3 t_4) = P(t_1) \cdot P(t_2) \cdot P(t_3) \cdot P(t_4) \tag{2.5}$$

Bigram

One instance of a more intricate model that accounts for the preceding context in its evaluation is the bigram language model as shown in equation 2.6. In this model, the likelihood of each term is dependent on the preceding term [Manning et al., 2008, p. 240].

$$P_{bigram}(t_1 t_2 t_3 t_4) = P(t_1) \cdot P(t_2|t_1) \cdot P(t_3|t_2) \cdot P(t_4|t_3) \tag{2.6}$$

Multinomial Unigram

The unigram and bigram models lack consideration of the multinomial probability of a bag of words as they don't sum over all possible token orderings. The incorporation of the multinomial coefficient C_q in equation 2.7, however, can account for this, leading to the development of the standard *multinomial unigram language model* for a given query q as expressed in equation 2.8, with M_d representing the model for a specific document d.

$$C_q = \frac{L_d!}{tf_{t_1,d}! \cdot tf_{t_2,d}! ... tf_{t_M,d}!} \tag{2.7}$$

$$P(q|M_d) = C_q \prod_{V \in q} P(t|M_d)^{tf_{t,d}} \tag{2.8}$$

In instances where all terms in a query are absent from a document, the similarity score may result in zero. To address this issue, *term smoothing* is commonly employed in these language models. This approach not only prevents the zero problem but also implements a significant portion of the term weighting component [Bennett et al., 2008], [Manning et al., 2008, pp. 241-244]. Additionally, for a specific bag of words, the coefficient C_b would simply be a constant.

2.4 Evaluation of IR Systems

The assessment of search performance in a objective manner holds significant importance as a fundamental element of IR research.

2.4.1 Effectiveness and Efficiency

The advancement of Information Retrieval (IR) research relies heavily on the experimental nature of the field, and progress is largely dependent on the exploration of new ideas. However, empirical evidence has shown that new ideas and potential enhancements for the search capabilities of a system, which may seem promising in theory, often have little to no impact when put to extensive testing. Thus, to make noticeable progress, continuous evaluation and experimentation with alternative techniques is essential to differentiate between useful changes and those that are redundant [Singhal et al., 2001].

The effectiveness and efficiency of an IR system are two ways to evaluate its performance. *Effectiveness* measures the system's ability to locate relevant information, and if a definition of relevance is available, it can be determined by comparing the ranking produced by the IR system in response to a query with the ranking created by user relevance judgments. Conversely, *Efficiency* measures the amount of time or space required by the system to generate that ranking.

Generally, IR research focuses first on improving the effectiveness of the system. Once a new technique is found to achieve this goal, resources are allocated to create an efficient implementation. This approach is commonly adopted because the primary objective is to identify relevant information. A retrieval system that produces quick results but without accuracy would be of little use in most situations [Van Rijsbergen, 1979, p. 112], [Croft et al., 2010, pp. 269-271].

2.4.2 The Cranfield Paradigm

Cleverdon's evaluation methodology facilitated the development of test collections, which included a series of documents, queries with descriptions of relevant document types, and a list of documents that had been evaluated by assessors and categorized as relevant or not [Cleverdon, 1991]. These categorized documents for each query are commonly referred to as qrels (query relevance) and they indicate whether or not a document is relevant to the given query. Initially, all documents could be evaluated for each query, but as collection sizes grew, this approach became impractical. Consequently, system pooling, which involves multiple systems running the queries and a subset of top documents from each system being judged for relevance by assessors, became the preferred approach. Although this method is not exhaustive, researchers have attempted to develop collections without system pooling [Sanderson and Joho, 2004] or using different pooling techniques [Zobel, 1998]. Biases introduced by pooling, such as length biases [Losada and Azzopardi, 2008], have been acknowledged by researchers and compared to a more comprehensive coverage approach [Buckley, Dimmick, et al., 2006; Buckley, Dimmick, et al., 2007]. Nevertheless, system pooling is still the standard method for most test collections, and it remains the accepted norm in the community for creating collections.

The standard paradigm for evaluation involves the following steps:

- The system indexes the set of documents in the test collection.
- The system is issued with a set of curated queries to be evaluated.
- The top 1000 ranked results from the system are recorded.
- The relevance of the ranked results for each query is determined by comparing them with the qrels for that query.
- The system's performance is calculated based on the previous step's results.

This approach provides a controlled environment for testing and comparing different systems to determine which is best suited for a given task. However, it is not immune to potential pitfalls, such as overfitting the model to maximize performance on one collection, which may lead to poor performance on other collections with different statistics.

Standard Test Corpora

The implementation of a standardised evaluation methodology resulted in the establishment of TREC, a workshop or conference that furnishes researchers with high-quality test collections for a variety of retrieval purposes and topics. These collections are utilized in TREC Tracks, a competition that enables researchers to assess the efficacy of their models and compete with one another. To appraise the retrieval runs, TREC employs common evaluation metrics based on the qrels. The qrels contain lists of relevant and non-relevant documents for most topics, enabling a system to be penalized for retrieving a document that is known in the relevance judgment to be non-relevant. The system's score is typically not influenced by the remaining documents that are unjudged and do not appear in the qrels for a particular topic, and hence do not contribute to the enhancement or penalty of the score.

Basic Requirements for Evaluation

In order to evaluate the effectiveness of an IR system using standard practices, the following three components are necessary:

- Document Collection: A comprehensive compilation of documents that is representative of the corpus of information that the IR system is designed to search.
- Query Set: A set of queries that expresses the user's information needs and simulates real-world search scenarios.
- Relevance Judgments: A set of binary labels that assesses the relevance of each query-document pair. These judgments are made by human assessors and serve as a benchmark for measuring the performance of the IR system.

2.4.3 Evaluation Measures

The last phase in the evaluation process, which involves the application of an evaluation metric, necessitates a detailed explication due to the existence of a vast

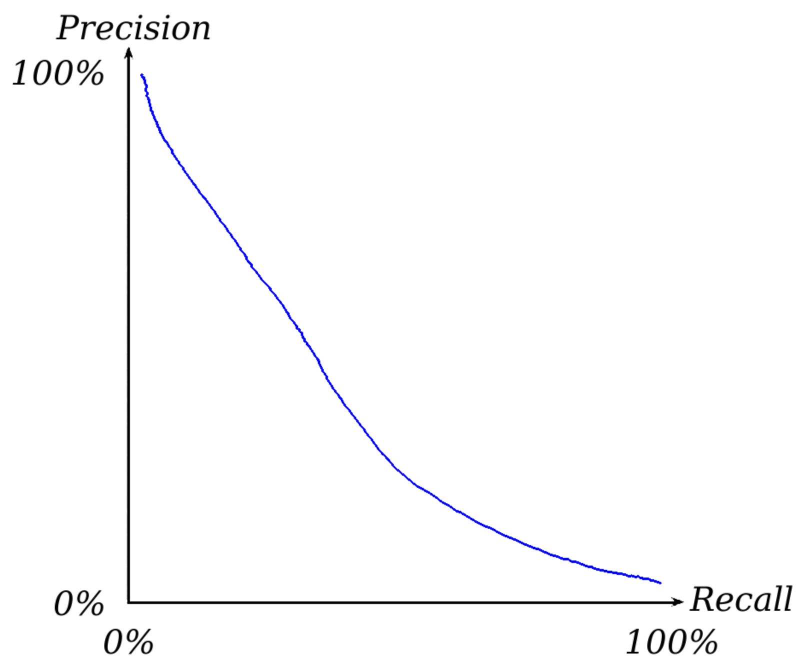


Figure 2.5: Inverse relation between Precision and Recall

array of evaluation metrics. In this context, we will expound upon some of the most preeminent metrics, as well as some lesser-known metrics.

Precision and Recall are widely accepted metrics for evaluating the effectiveness of retrieval systems and their ability to meet information needs. These two measures are often inversely related, making it impossible to achieve the best value for both simultaneously. Figure 2.5 depicts a typical relationship between the two, where the slope of the curve may vary depending on the collection and retrieval systems used, but the general pattern remains consistent. The ideal outcome is situated at the upper-right corner of the plot, where both Precision and Recall are maximized.

To consolidate Precision and Recall into a single value, F-measure can be utilized. While these three values are originally expressed as measures between 0 and 1, it is common practice to express them as percentages. In ranked retrieval settings, Mean Average Precision is a single measure that combines Precision and Recall. However, all of these metrics require relevance judgments, which can be incomplete. In such cases, Binary Preference may be used. Retrievability is another metric that does not require any relevance judgments, and it will play a central role in the experiments conducted in the practical section of this thesis.

Unranked retrieval refers to when an IR system returns a set of unordered documents for a given query. Four possible combinations of retrieval state and

	relevant	not relevant
retrieved	true positives (TP)	false positives (FP)
not retrieved	false negatives (FN)	true negatives (TN)

Table 2.1: Relevance and retrieved combinations [Manning et al., 2008, p. 155].

document relevance exist, as outlined in Table 2.1.

Accuracy

Accuracy, which refers to the fraction of correct classifications done by an IR system, is mathematically expressed in equation 2.9.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.9}$$

Although frequently employed to evaluate machine learning classification problems, this approach is typically unsuitable for IR systems. This is primarily due to the heavily imbalanced nature of IR data, where less than 0.1 percent of documents are pertinent to a given query. In fact, an IR system's accuracy can be artificially inflated by simply labeling all documents as non-relevant, a practice that would not serve the user's interests, as they anticipate the retrieval of relevant documents (true positives). Generally, users are content as long as their information requirements are met, even if some of the returned documents require scrutiny but are not pertinent (false positives) [Manning et al., 2008, pp. 155-156].

Precision

Precision is a metric utilized to determine the proportion of pertinent documents among the ones retrieved by the IR system, concerning a given query. In equation 2.10, it is evident that the computation of Precision necessitates knowledge of the relevant documents and the total number of retrieved documents. Nevertheless, a circumstance can arise where an IR system returns only one relevant document despite the presence of a significant number of relevant documents in the collection. In such cases, the Precision value can still be 100 percent, but the result set may not be satisfactory. Hence, to avoid such situations, Precision is often used alongside

Recall, as per the recommendations provided in [Manning et al., 2008, pp. 154-155].

$$Precision = \frac{TP}{TP + FP} \tag{2.10}$$

Recall

Simply maximizing Recall is not always a worthwhile approach, as it can result in 100 percent Recall being achieved by the system returning all documents from the collection as the result set. Although this method includes all relevant documents, it essentially disregards the query.

Recall is a metric that measures the ability of a retrieval system to successfully retrieve documents from a collection that are relevant to the query. This measurement requires knowledge of relevance, retrieved and non-retrieved documents. Recall ranges from 0 to 1 and is defined by Equation 2.11.

$$Recall = \frac{TP}{TP + FN} \tag{2.11}$$

An ideal IR system should possess high Recall and Precision values, which means it should retrieve as many relevant documents as possible while only retrieving a small number of non-relevant ones. Although one approach is to maximize both values, this does not always represent the optimal combination for the user's needs or the system's purpose. Therefore, another balance may be more favorable for achieving the desired outcome.

To simplify comparisons between different IR systems, the *breakeven point* is a single value that can be used. This point is where Precision and Recall are equal and is usually achieved by tuning the IR system's parameters [Manning et al., 2008, pp. 154-156, 161].

F-Measure

An approach to balancing Precision and Recall is through a measure called *F-measure*. This measure is calculated as a weighted harmonic mean of the two, with the equation outlined in 2.12. By adjusting the value of β, where $\beta < 1$ emphasizes

Precision (P) and $\beta > 1$ emphasizes Recall (R). The default value of β is 1, which equally weights both Precision and Recall, known as the *balanced F-measure* or F_1, short of $F_{\beta=1}$ [Manning et al., 2008, p. 156].

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{2.12}$$

Mean Average Precision

Currently, search engines commonly employ *ranked retrieval*, whereby the top k documents deemed relevant for a query are returned as a result list. Users tend to focus on the beginning of this list, and this behavior can be illustrated via *Precision-Recall Curves*. *Mean Average Precision* (MAP) is the most widely accepted measure used to express the quality of ranked retrieval systems. Specifically, for a given query, the Average Precision is calculated by averaging the Precision values for each relevant document found among the top k retrieved documents. A high Average Precision value indicates that many relevant documents are present at the top of the retrieved list. To obtain the MAP, this process is repeated for multiple queries, and the resulting Average Precision values are averaged. Equation 2.13 demonstrates this calculation, where $\{d_1, ..., d_{m_j}\}$ represents the set of relevant documents for a given query $q_j \in Q$, and R_{jk} denotes the set of ranked results up until document d_k is reached [Manning et al., 2008, pp. 159-160].

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} P(R_{jk}) \tag{2.13}$$

As the MAP values for different information needs can vary significantly within the same system, they are often utilized to compare a system with others using a single information need [Manning et al., 2008, p. 161].

Binary Preference

Binary Preference (bpref) measure can be employed to evaluate incomplete relevance judgments. This metric determines a preference relation of the judged documents, indicating whether the retrieved documents are relevant or not. In accordance with equation 2.14, bpref is computed by taking into account relevant documents $r \in R$

set of all relevant documents and n is a member of the first R non-relevant documents retrieved by the IR system. Generally, bpref is an effective measure for evaluating IR systems, but it may not work optimally when the number of relevant documents is very limited, as noted in [Buckley and Voorhees, 2004].

$$bpref = \frac{1}{R} \sum_{r} \left(1 - \frac{|n \text{ ranked higher than } r|}{min(R, N)} \right) \tag{2.14}$$

2.5 Query Expansion

Furnas et al [Furnas et al., 1987] conducted a study indicating that individuals tend to use the same language to describe an object only a small percentage of the time, referred to as "word mismatch." In the realm of information retrieval, this poses a significant issue, as a document may not explicitly contain the terms present in a search query. Nevertheless, the document may be pertinent to the query's underlying information need. If a relevant document lacks the specific terms used in a query, it will not be retrieved. The purpose of *query expansion* is to address this query-document mismatch by broadening the search query through the inclusion of words or phrases with similar meanings or some other statistical relationship to the relevant document set.

To overcome this issue, users frequently attempt to refine their queries manually. In this section, we explore methodologies in which a system can assist with query refinement, either entirely automatically or with the involvement of the user.

2.5.1 Approaches

With the use of *query expansion*, users have the ability to provide additional input on the query terms themselves, rather than solely relying on the resulting information. Particularly for IR systems utilized on the internet, alternate but related queries are often suggested to the user. The generation of these alternate suggestions is commonly accomplished through the use of a thesaurus, which contains synonyms or semantically related subjects. Query expansion can also be automated by adding synonyms from the thesaurus to the query for each query term. This process can enhance the amount of relevant documents retrieved without the need for user input.

Manual and automatic methods exist to create a thesaurus. In a manual approach, editors can maintain a list of canonical terms for each concept, similar to traditional libraries where subject indices include a vocabulary of possible synonyms or related terms. This method is commonly used in well-resourced domains. Alternatively, a thesaurus can be automatically generated through the use of statistical data on term co-occurrences in a collection of documents from a specific domain. Another automatic approach to suggest query alternatives to a new user is to exploit the manual query reformulation attempts of users who have previously used the IR system. To perform this query log mining, a large number of generated queries and users are necessary, making it a viable method for web search systems [Manning et al., 2008, pp. 189-192].

2.5.2 Relevance Feedback

The concept of *relevance feedback* (RF) relies on the premise that users may struggle to create an effective query if they are not familiar with the collection. Therefore, they are given the opportunity to engage with the retrieval process to enhance the final list of results. This process involves several steps, as follows:

- The user submits a brief and straightforward query to the system.
- The system produces an initial list of results.
- The user provides feedback by marking some of the results as relevant or irrelevant.
- Based on the feedback received, the system generates an improved representation of the user's information needs.
- The system presents an updated result list to the user.

The relevance feedback process may be repeated through multiple iterations, following the last three steps to further refine the results. This approach is particularly beneficial in image searches where the user may find it challenging to express their needs in a few terms but can quickly evaluate the results. However, users may not find the relevance feedback process popular since they may not wish to invest additional time refining their query or providing feedback. Additionally, it may be challenging to understand why a specific feedback choice has impacted the next result set.

To address the issue of users not engaging explicitly, modified versions of relevance feedback have been developed. Blind relevance feedback assumes that the top-ranked documents in the result list are relevant and utilizes this assumption to generate feedback data that can feed the feedback loop. Implicit relevance feedback analyzes user actions such as clicking on a returned document to read it and adjusts the ranking based on that indirect feedback [Baeza-Yates, Ribeiro-Neto, et al., 1999, pp. 117-118], [Manning et al., 2008, pp. 178, 185-188].

The central concept behind relevance feedback involves the utilization of terms or expressions extracted from documents that have been marked as relevant, to refine the query. Conversely, irrelevant documents may also furnish negative cues that can be leveraged to facilitate query reformulation. Relevance feedback comes in three primary forms:

1. **Explicit relevance feedback**: Here, the system user explicitly designates a few top-ranked documents as either relevant or irrelevant to their information need.
2. **Implicit relevance feedback**: In this scenario, users do not overtly mark documents. Rather, documents viewed by users serve as feedback, and the information gleaned from such views is used to modify the query.
3. **Pseudo-relevance feedback**: In this mode, no user interaction is required. It is presumed that the k top ranked documents are relevant, and the IR system utilizes the reformulated query derived from these *pseudo-relevant* documents to optimize its performance.

2.5.3 Rocchio Algorithm

The Rocchio algorithm constitutes a practical application of relevance feedback for query expansion in the field of IR. Its theoretical foundation rests upon the vector space model (Figure 2.6). Its primary objective is to identify a query vector that optimizes the degree of similarity with pertinent documents and concurrently reduces the level of similarity with non-relevant documents [Rocchio Jr, 1971].

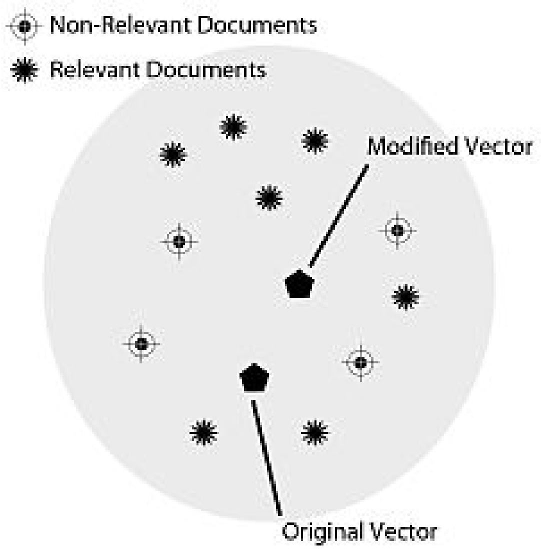


Figure 2.6: Rocchio algorithm illustration in vector space. (Image: Wikipedia, 2023b)

The Rocchio relevance feedback is mathematically written as equation 2.15:

$$\vec{Q}_m = \alpha \vec{Q}_0 + \beta \frac{1}{|D_r|} \sum_{D_j \in D_r} \vec{D}_j - \gamma \frac{1}{|D_{nr}|} \sum_{D_k \in D_{nr}} \vec{D}_k \tag{2.15}$$

Where $\vec{Q}_0$ represents the original query vector, $\vec{Q}_m$ is the modified query vector, and D_r and D_{nr} represent the sets of relevant and non-relevant documents, respectively, the values of the weights α, β, and γ determine the direction and distance of the modified query vector from the original query, relevant documents, and non-relevant documents. When a large number of judged documents are available, a higher weight for β and γ may be desired for the modified query vector to move closer to the centroid of relevant documents and further away from the centroid of non-relevant documents, starting from the original query vector $\vec{Q}_0$.

2.5.4 Relevance Based Language Model

For a considerable amount of time, statistical language models have been employed and investigated. These models serve as a means of generating text and have historically been extensively used in the realm of speech recognition.

RLM

Ponte and Croft introduced the Language Modelling framework of information retrieval in their paper "A language modeling approach to information retrieval" [Ponte and Croft, 2017]. The crux of this framework lies in its attempt to model the query generation process as a random sampling from one of the document models. The documents are then ranked based on the likelihood of observing the query as a random sample from the respective document model. Lavrenko et al's Relevance Based Language Modelling (RLM), presented in [Lavrenko and Croft, 2017], may be regarded as a query expansion technique that operates within the language modelling framework. Their approach proposes a novel method of estimating word probabilities in relevant documents without the use of any training data. The probability of observing a word w in the relevant set R, $P(w|R)$, is approximated by the probability of co-occurrence between the word w and the query, $P(w,Q)$. The top few words ranked according to $P(w|R)$ are then selected as the expansion terms.

The *probability ranking principle*, famously advocated by Robertson in [Stephen E Robertson, 1977], stipulates that optimal performance can be achieved by ranking the documents according to the posterior probability that they belong to the relevant class R. Robertson [Stephen E Robertson, 1977] also demonstrates that ranking the documents based on the odds of their being observed in the relevant class is equivalent to this method. We can, therefore, rank our documents using this principle by:

$$\frac{P(D|R)}{P(D|R)} \sim \prod_{w \in D} \frac{P(w|R)}{P(w|R)}$$

Additionally, an estimation of the probability of selecting a term w from the set R, represented as $P(w|R)$, is derived through approximation by

$$P(w|R) \sim \frac{P(w,Q)}{P(Q)}$$

The joint probability of observing word w along with query terms Q is denoted as $P(w,Q)$, while $P(Q)$ represents the query prior probability.

Under the assumption that the words q_1, q_2, q_3, and so on in the query, and the words w in the relevant documents, are sampled independently and identically from the top retrieved document, given a query $Q = \{q_1, q_2, ...\}$ and a set M of documents from the initial retrieval, then

$$P(w,Q) = \sum_{D \in M} P(D)P(w,Q|D)$$
$$P(w,Q|D) = P(w|D)\prod_{q \in Q} P(q|D)$$

$$P(w,Q) = \sum_{D \in M} P(D)P(w|D)\prod_{q \in Q} P(q|D) \tag{2.16}$$

where,

- $\prod_{q \in Q} P(q|D)$: Maximum likelihood estimate of q in D
- $P(w|D)$: Maximum likelihood estimate of w in D
- $P(D)$: Prior probability of selection of the document.

RM3

In practical application, the combined use of the Relevance-based Language Model and Query Likelihood Model, as proposed by [Abdul-Jaleel et al., 2004], is commonly employed. As noted by the authors, the Relevance model provides insight into the behavior of returned documents but does not take into account the original query. To preserve the information contained in the original query, the authors suggest linearly interpolating the Relevance model with the original query model.

RM3, a mixture model of Relevance based language model and query likelihood model:

$$P'(w|R) = \alpha P(w|R) + (1-\alpha)P(w|Q) \tag{2.17}$$
$$\text{where, } P(w|Q) = \frac{tf(w,Q)}{|Q|}$$

2.6 PageRank

PageRank is a link-based ranking algorithm developed by Brin and Page [Brin and Page, 1998], which determines the importance of a webpage based on the quantity and quality of links it receives from other webpages. The fundamental assumption underlying PageRank is that more important websites are likely to receive a greater number of links from other websites. The algorithm generates a probability distribution that represents the likelihood of a random surfer arriving at any given webpage by clicking on links.

Multiple link connections from one page to another are treated as a single link, and links from a page to itself are disregarded. The damping factor, typically set to around 0.85, plays a crucial role in the PageRank algorithm. This factor accounts for the likelihood that a random surfer may stop following links and instead jump to a random webpage. It prevents the algorithm from getting trapped in cycles of infinite link traversal.

The essence of PageRank can be understood through a model of a random surfer who, after several clicks, eventually reaches the desired webpage and then randomly transitions to another webpage. PageRank value assigned to a page reflects the probability that the random surfer will land on that page by clicking on a link. Conceptually, PageRank can be interpreted as a Markov chain where pages represent states, and links between pages represent transitions. Each transition is considered equally probable.

Mathematically, the PageRank equation can be expressed as follows:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

In this equation, $p_1, p_2, ..., p_N$ represents the pages under consideration. $M(p_i)$ represents the set of pages that link to p_i, and $L(p_j)$ denotes the number of outbound links on page p_j. N represents the total number of pages.

The PageRank values correspond to the entries of the dominant right eigenvector of the modified adjacency matrix. The matrix is rescaled to ensure that the sum of each column adds up to one, making PageRank an elegant metric. The eigenvector is represented as:

$$\mathbf{R} = \begin{bmatrix} PR(p_1) \\ PR(p_2) \\ \vdots \\ PR(p_N) \end{bmatrix}$$

where $\mathbf{R}$ is the solution of the equation:

$$\mathbf{R} = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ \vdots \\ (1-d)/N \end{bmatrix} + d \begin{bmatrix} \ell(p_1,p_1) & \ell(p_1,p_2) & \cdots & \ell(p_1,p_N) \\ \ell(p_2,p_1) & \ddots & & \vdots \\ \vdots & & \ell(p_i,p_j) & \\ \ell(p_N,p_1) & \cdots & & \ell(p_N,p_N) \end{bmatrix} \mathbf{R}$$

In this equation, $l(p_i, p_j)$ represents the ratio between the number of links outbound from page p_j to page p_i and the total number of outbound links on page p_j. If p_j does not link to p_i, the adjacency function $l(p_i, p_j)$ is 0. Additionally, the elements of each column in the matrix sum up to 1, ensuring the matrix is stochastic. This formulation of PageRank resembles a variant of eigenvector centrality commonly employed in network analysis.

One of the primary drawbacks of PageRank is its bias towards older pages. New pages, even if they are of high quality, tend to have fewer links unless they are part of an existing densely connected site, such as Wikipedia. This limitation can result in slower recognition and ranking for newer webpages, hindering their visibility in search engine results.

Chapter 3

Retrieval Bias and Retrievability

This chapter provides an introduction to the concept of Retrieval Bias in Information Retrieval (IR), along with a brief review of previous attempts to measure it. The chapter then transitions to the discussion of Retrievability, a more recent measure aimed at quantifying retrieval bias in IR systems. The thesis primarily employs Retrievability as a measure of retrieval bias, and thus, this chapter will provide a detailed analysis of its definition and associated analytical methods.

3.1 Retrieval Bias

Bias is the propensity to show partiality toward a particular individual or group, particularly for unjustifiable reasons, although this may also encompass biases that are viewed as legitimate. The definition of bias is multifaceted and can generate controversy within society, particularly when it pertains to racial or gender bias. Nonetheless, this topic is crucial to numerous fields beyond IR. For instance, the machine learning industry is currently examining the challenges presented by biases identified in algorithms that are influenced by the data they learn from [Zehlike et al., 2017]. The use of black box machine learning algorithms, which are becoming increasingly prevalent, can result in discrimination lawsuits when predictions are made based on characteristics such as sexuality, gender, or race [Hajian et al., 2016]. Similarly, information retrieval (IR) systems can face comparable consequences and accusations, as demonstrated by one of the Presidents of the United States, who accused Google of manipulating search results to malign his reputation, a task that Google appeared to have no difficulty in accomplishing.

The above discussed definition of bias can be applied to retrieval, where certain developments in the history of IR can be deemed as biased, but still have a positive impact on performance in specific contexts. These biases are integrated into the system's design, as observed in algorithms such as PageRank [Brin and Page, 1998] and HITS [Kleinberg, 1999], which utilize the hyperlink structure of the web to enhance the relevance score of web pages based on their popularity. Although discriminatory algorithms intentionally designed to discriminate have surfaced, they are typically used alongside traditional retrieval algorithms. When referring to retrieval bias, however, we do not consider these intentional biases from our definition. Hence, we define retrievability bias as unintentional bias that generally has an unfavorable effect on performance.

The issue of bias in IR is a long-standing concern, with evidence of underlying biases present in retrieval algorithms documented as far back as the earliest retrieval models, as shown by [Sparck Jones, 1972]. The goal of improving retrieval algorithms has been motivated, in part, by the need to eliminate biases from the process and improve system performance. Singhal's PTF.IDF model [Singhal et al., 1996] provides a prime example of this development, as it introduced a pivot to TF.IDF that allowed for parameterizing the length normalization of the model. By adjusting the model accordingly, the inherent length bias, which tends to favor long or short documents based on collection statistics, can be mitigated against, leading to improved performance. The development of less biased models has been an iterative process that has resulted in the creation of models that are provably less biased than their predecessors. However, demonstrating that a model or its configuration is less biased than others has been challenging due to the lack of a measure until 2008. Previous attempts to quantify bias, such as those focused on web search [Mowshowitz and Kawaguchi, 2005], were limited by the vastness of the web at that time. Their method involved pooling a selection of web retrieval systems and issuing a common set of queries to each, recording the results of each system for each query to create an ideal ranking. The authors then compared the systems' rankings with the ideal ranking to determine how far the system deviated from the ideal and, therefore, how biased the system was. Although this method had some merits, such as its focus on peer comparison, it did not address bias in a broader sense, as it assumed that a collection of systems could create an unbiased ranking, which may not be the case given that commercial search engines may be influenced by business and political interests. Thus, a measure for quantifying bias was still lacking.

Until the publication of Vinay et al.'s (2006) work [Vinay et al., 2006] on retrievability, there was no generally accepted method for quantifying bias in information retrieval (IR). Additionally, the process of evaluating IR performance was not immune to flaws. One example of this is when selecting and adjusting an algorithm for live use on a collection, which is commonly done without having any relevance judgements available for that specific collection. To address this, it is customary to first test and adjust the algorithm on a similar collection with relevance judgements before applying it to the working collection. However, this practice has the potential to lead to overfitting because the test collection may differ from the working collection in terms of size, domain, or structure, rendering any tuning efforts ineffective. Overfitting occurs when an algorithm is optimized for collection statistics that do not represent the working collection, resulting in poor performance when the algorithm is applied to the working collection. Unfortunately, this can be difficult to detect, as evaluating the algorithm is challenging and involves reviewing queries and rankings. However, if an IR algorithm were tuned to minimize retrievability bias (assuming a strong negative correlation between bias and performance), it would be possible to avoid the overfitting risk by tuning the algorithm on the working collection using retrievability analysis, which does not require a test collection.

In performance evaluation, biases can arise despite the risk of overfitting. Pool bias, in particular, is a significant concern in collection creation. This bias arises due to the process of system pooling, which involves creating a pool of documents to be judged for each query using past retrieval algorithms. This method helps reduce the number of documents that need to be reviewed by judges. However, using previous algorithms means that any biases they hold are now included in the pool. Consequently, judges may be subject to a biased pool, leading to biased judgments. Losada and Azzopardi investigated this phenomenon in terms of length biases and found that commonly used test collections exhibited strong length biases. This effect can be cumulative as each new collection is created, and successful algorithms from previous collections contribute to the pool, potentially contributing biases each year. Our study will delve further into this concept to determine if performance evaluations are genuinely biased from the outset.

3.2 Measuring Retrieval Bias

Retrievability can be compared to navigability metrics, which are frequently used to quantify the accessibility of nodes in a graph by traveling along its edges [Zhang et al., 2004], as it is rooted in transportation planning that is reliant on graph theory [Azzopardi and Vinay, 2008a]. One common usage of navigability metrics, which is similar to retrievability, is to evaluate how easily one can navigate through a web graph and visit pages by following existing links. To convey this information, measures such as PageRank and HITS were popular [Brin and Page, 1998; Kleinberg, 1999], but these were limited to collections with explicit links between documents that could be browsed. Azzopardi, Wilkie, and Russell-Rose attempted to quantify the findability of a webpage within its host site by employing a combination of navigability and retrievability measures [Azzopardi, Wilkie, et al., 2013; Wilkie and Azzopardi, 2013a]. The authors aimed to assess the structure of a website using this combination of measures, encompassing both browsing and searching aspects of website navigation. PageRank and HITS were utilized as navigability measures, while BM25 was employed as the retrieval model. The findings from these assessments were then combined, and the resulting findability scores were correlated with usage logs of the website being evaluated. While some correlation was observed, it appeared that the users' information needs were the primary driver behind website usage. However, the measure was noted to have the potential to identify pages that were challenging to locate but still received substantial traffic, which could serve as guidance for website restructuring. In a similar study, Azzopardi et al. [Azzopardi, English, et al., 2014] developed a tool that provided a rating of a page's retrievability based on the content of the page. The tool issued terms from the content to a retrieval system and calculated the page's retrievability from the results obtained.

Additional metrics related to navigability that are comparable to retrievability are reachability and hubness. Sabetghadam et al. define reachability as the ease with which a document can be accessed using a graph traversal algorithm through a network of interconnected documents [Sabetghadam et al., 2015]. While retrievability measures the ease with which a document can be retrieved from a collection, reachability addresses the same issue in a different context. Reachability is a measure of whether a document can be accessed at all by following links within the graph, limiting the number of steps taken. Similarly, hubness also involves nodes in a graph [Taha, 2015]. Hubness is concerned with high-dimensional spaces, such as musical similarity, and the effect of minor variations in the graph on the outcomes [Gasser et al., 2010]. In high-dimensional spaces, a hub document is one that is frequently retrieved not because of its similarity but because it is closer to the mean. In such cases, all nodes should be located near the surface of a hypersphere, and a single node that is slightly positioned towards the mean becomes similar to a significant number of documents. Taha et al. suggest either eliminating hub documents [Taha, 2015] or combining features to reduce hubness [Flexer et al., 2010]. Azzopardi and Vinay have used the idea of eliminating hub documents to investigate retrievability [Azzopardi and Vinay, 2008b]. Instead of removing the most retrievable documents, they removed the least retrievable documents and analyzed the impact on performance. The authors discovered that a substantial portion of the collection could be removed from the index before any significant reduction in performance occurred. These measures, including retrievability, illustrate the notion that certain areas of a collection may be challenging to access.

3.3 Retrievability

Azzopardi and Vinay originally introduced the concept of "retrievability" as an analogy to public transport planning in their 2008 paper [Azzopardi and Vinay, 2008a]. They viewed the documents in a collection as destinations that the user wishes to reach, with the retrieval system serving as a central transport hub enabling access to all possible destinations. In the context of retrieval, the user poses a query relevant to their information need to reach their desired document, whereas in the transport planning analogy, the user takes a bus or train to reach their destination. The authors acknowledged that despite the ideal scenario of being able to retrieve any document with a relevant query, this was not always the case due to various factors.

Retrievability is an evaluation metric that focuses on individual documents within a collection, and quantifies how easily each document can be retrieved. It can also be used to determine the degree of bias present in a retrieval system towards a particular collection. To ensure the computation of retrievability is not influenced by external factors, the methodology is designed to focus on the collection, retrieval algorithm, and its configuration (such as parameters). The mathematical notation for calculating retrievability is provided as follows:

$$r(d) \propto \sum_{q \in Q} O_q \cdot f(k_{dg}, \{c, g\}) \tag{3.1}$$

where Q represents the set of all possible queries and q is a specific query within that set, O_q refers to the probability of selecting that query. Despite its importance, O_q has not been thoroughly investigated in existing literature. The retrieval of a document d for a query q is determined by its rank k_{dq}, and $f(k_{dq}, \{c, g\})$ represents the corresponding access function that indicates the degree of retrievability of d for q at rank cut-off c with discount factor g. To determine the overall retrievability of d, we sum the product of O_q and $f(k_{dq}, \{c, g\})$ over all queries in the set Q. Due to the impracticality of testing all possible queries, a vast array of queries is automatically generated from the collection. Essentially, the measure captures the idea that the higher the number of queries that retrieve d before the rank cut-off c, the more retrievable document d is considered to be.

The *cumulative scoring model* is the most simplified method for computing document retrievability. This model utilizes an access function, denoted by $f(k_{dq}, c)$, which assigns a score of 1 to a document d if it appears in the top c results for a query q, and a score of 0 otherwise. Alternatively, the *gravity-based model* incorporates a discount factor g to determine the score a document receives based on its rank. In this model, a document returned at a higher rank will receive a greater score than one returned at a lower rank. A cut-off parameter c can still be employed with the gravity-based model, which simulates the user's tendency to focus on the top results of a search query. This behavior can lead to position bias, where users are less likely to click on lower-ranked results as they scan the search results list. As an illustration, when a user searches for information, they may examine the first ten documents listed in the search results. However, as they scroll down the ranking, they are less likely to select a result that appears further down due to the position bias. This means that a document ranked number 10 is less easily retrievable than the one at rank 1. To obtain a cumulative measure, we can analyze a sufficiently large cross section of queries and count how many times a document d was returned above a certain rank, c. On the other hand, the gravity measure provides a more precise indication of a document's retrievability to a user.

The principal application of the retrievability's theory has been to measure the degree of bias that a system configuration creates over a collection of documents. Nevertheless, retrievability is assessed on a document-to-document basis, and as Equation 3.1 demonstrates, the outcome of performing a retrievability analysis is the retrievability scores for every document. To transform this set of retrievability scores into a single score indicating bias, approaches from political science have been employed. Azzopardi and Vinay summarized and quantified bias using the Gini Coefficient, which is an income inequality measure used to measure the inequality (in our sense, bias) of the distribution of income or wealth across the population of a country, region, or the group [Gastwirth, 1971]. Gini Coefficient calculates inequality by arranging a population in ascending order of their income and plotting the cumulative distribution of income over the ascending ordered population. The extent of deviation of this distribution from the Lorenz Curve reflects how unequal the distribution is. A distribution that approaches the Lorenz Curve indicates that the wealth/income is distributed equally throughout the population. However, the farther the distribution is from the Lorenz Curve, the more biased the distribution is. In the worst-case scenario, one individual would receive all of the wealth, while everyone else would receive nothing. In terms of retrievability bias, the population represents a collection of documents, and the total income or wealth represents the sum of the retrievability score of each document in the collection.

3.4 Retrievability Analysis Framework

Several studies have employed the concept of retrievability, leading to the establishment of a standard methodology for conducting retrievability analyses. This methodology comprises a set of fundamental stages that can be flexibly adapted to fit the specific context of the research question. The approach can be broken down into five primary stages, namely: generation of the query set, configuration of the system, execution of the query set, computation of document retrievability, and summarization of retrievability outcomes.

3.4.1 Query Set Generation

The initial step in a retrievability analysis is typically the generation of a query set, provided that the collections used have already been indexed. This step is of paramount importance, given that the estimation of retrievability hinges on it. As per Equation 3.1, the most ideal calculation of $r(d)$ would require the use of all possible queries, denoted by Q. However, issuing all conceivable queries is an impossible task; hence, Q is represented by an extensive set of queries [Azzopardi and Vinay, 2008b]. Frequently, this query set is automatically derived from the collection itself [Azzopardi and Vinay, 2008b; Bashir and Rauber, 2009a; Bashir and Rauber, 2009b; Chen et al., 2017; Ganguly et al., 2016; Lipani et al., 2015; Noor and Bashir, 2015; Pickens et al., 2010; Traub et al., 2016; Wilkie and Azzopardi, 2013a; Wilkie and Azzopardi, 2013b; Wilkie and Azzopardi, 2014; Wilkie and Azzopardi, 2015], using an extraction method similar to that employed by Jordan [Jordan et al., 2006], which identifies the terms contributing most to the set's entropy. A common approach involves the extraction of bigrams from a collection of text. This is done by sliding a window across the text, and saving each bigram that appears a specified number of times. The bigrams are then ranked and a predetermined number of the top-ranked bigrams are selected to create a query set of significant size [Azzopardi and Vinay, 2008b; Wilkie and Azzopardi, 2013b]. Azzopardi and Vinay's approach was initially based on the method outlined by Callan and Connel [Callan and Connell, 2001]. The authors generated a set of queries, which included single term queries, formed by selecting each term in the vocabulary that appeared at least 5 times and using it as a query, and bi-term queries, formed by selecting each bigram in the collection that appeared at least 20 times. The list of bigrams was truncated at 20 million, and each query was issued to the system to estimate its retrievability. This method created query sets of 1,797,520 and 2,881,230 queries for the Aquaint and .Gov collections, respectively. Azzopardi and Bache later followed a similar methodology to generate separate query sets for the AP and WSJ collections by ranking the top 100,000 collocations in each collection [Azzopardi and Bache, 2010].

A frequently employed approach involves the production of n-grams from a given collection. This method entails the identification of all terms that occur with a frequency exceeding a predetermined threshold in each document, followed by the creation of permutations from these terms [Bashir and Rauber, 2009a; Bashir and Rauber, 2009b; Bashir and Rauber, 2009c; Bashir and Rauber, 2010a; Bashir and Rauber, 2010b; Bashir and Rauber, 2011; Bashir and Rauber, 2017; Noor and Bashir, 2015]. Bashir's work is particularly focused on extracting queries from the claim section of patent documents, which functions as an abstract for each patent and provides a concise summary of its content. Bashir and Rauber employed the controlled query generation (CQG) technique [Jordan et al., 2006] in their research on identifying the most and least retrievable patents in a collection [Bashir and Rauber, 2009b], as well as in their investigation of pseudo relevance feedback [Bashir and Rauber, 2009b]. They employed two different methods of CQG to create two distinct sets of queries for the same collection. In their first approach, they emulated the method used by patent examiners when conducting a patent invalidation procedure. Specifically, they extracted all terms from the claims section of the patents and then combined the most frequent terms (up to a certain threshold) into two, three, and four-term queries. Their second method of query generation was centered on document relatedness. Similar to the first approach, they generated queries by extracting terms from the claims section of documents, but instead of focusing on a single document, they first clustered documents using k-nearest neighbors and then generated queries from the combined claims sections of all documents in the cluster.

In a later study by Bashir and Rauber [Bashir and Rauber, 2010a], the relationship between retrievability and recall was examined by generating four subsets of queries for evaluation purposes. To accomplish this, they employed a technique that involved extracting each term, bigram, trigram, and 4-term that appeared multiple times in a document. As a result, they obtained four subsets of queries, ranging from around 30,000 to slightly less than 2.5 billion queries. This extensive analysis was conducted on the TREC Chemical Retrieval Track, which involved using numerous queries. The query generation method used by the authors was designed to model how expert searchers generate queries. In the domain of prior art, expert searchers use the claims section of a patent to generate queries that retrieve any patent making similar claims, thus locating the most likely candidates for conflicts of interest. While this method is effective for prior art search, it may not be well-suited to other domains, as few have a section similar to the claims of a patent. In a subsequent study on the same track, [Bashir and Khattak, 2014; Bashir and Rauber, 2011] employed the same query generation method as their earlier work. However, they appear to have used a much smaller subset of the queries generated, with the largest set comprising approximately 116 million queries. They also excluded 2-term combination queries, only issuing 3 or 4-term queries. The authors additionally noted that they eliminated any terms (before the combinations) with a document frequency exceeding 25% of the collection [Bashir and Rauber, 2014].

In a distinct approach to retrievability, Bashir's research explores efficient techniques to estimate retrieval bias without the need for query generation. Bashir's study employs document features instead of issuing queries to estimate retrieval bias [Bashir, 2014]. To accomplish this, Bashir identifies specific document features such as the collective TF.IDF score of all the terms present in the document.

Azzopardi et al conducted an alternative approach to query extraction by extracting queries from a single page [Azzopardi, English, et al., 2014]. The purpose of this method was to assess the retrievability of individual pages by issuing a set of queries derived from one page to the system and measuring the frequency at which this page was returned. Samar *et al* combined Azzopardi's approach [Azzopardi and Vinay, 2008b] with their own novel technique, which involved using the anchor text of hyperlinked web archive documents they were analyzing to generate queries [Thaer Samar et al., 2018; TMH Samar, 2017]. Samar's method of generating query sets from page content differed slightly from previous methods [Azzopardi and Vinay, 2008b] in that they selected the most frequently occurring bigrams in the collection while ignoring the most frequently occurring bigrams, considering them as stop words that offer little or no discriminative value [Thaer Samar et al., 2018]. This query generation technique was also followed by Traub *et al* [Traub et al., 2016]. Samar's anchor text method is a new approach to query set generation that involves extracting anchor text terms for pages from external webpages. This method is based on the notion that anchor text is often a concise and descriptive piece of text about the destination page, thereby creating relevant queries for a page [Thaer Samar et al., 2018].

Traub *et al* [Traub et al., 2016], in their analysis, utilized a real user query log, which offers the advantage of reflecting the actual queries submitted by users in the collection. This feature is particularly beneficial for studies that require the use of query sets that resemble users' search behavior. However, in the absence of a real user log, it is often acceptable to generate queries automatically. Therefore, the majority of studies have relied on automatically generated queries. Traub compared the retrievability estimates provided by real queries with those generated by simulation and found notable differences between the two query sets, particularly in terms of how many unique terms are present and the use of named entities in both the query sets. The real queries contained considerably more of both.

Pickens *et al.* [Pickens et al., 2010] present a noteworthy contribution in the form of inventing reverted index, utilizing a simplified approach to query generation. The authors establish a set of base queries for retrievability by extracting all terms that appear in more than one document, as indicated by a document frequency greater than one ($df > 1$).

The various techniques used for generating queries illustrate the diversity in approaches taken to the initial stage of a retrievability analysis, highlighting the absence of a well-defined structure for such an analysis. The query generation phase is a critical aspect of retrievability work, with two primary concerns: the quality and size of the query set. In terms of query quality, it is imperative that the generated queries are neither overly discriminatory nor not sufficiently discriminatory, and are generally suitable for the given collection. Failure to do so can lead to skewed results and confounding of biases during the analysis. Furthermore, the query set must be of sufficient size to yield a reasonably stable estimation of retrievability. Research conducted by Wilkie and Azzopardi indicates that eliminating too many queries from the generated set in an attempt to increase efficiency can lead to skewed results, particularly when the model has little bias to begin with [Wilkie and Azzopardi, 2014]. Therefore, the query set must be large enough to counteract the biases that can arise due to automatic query generation.

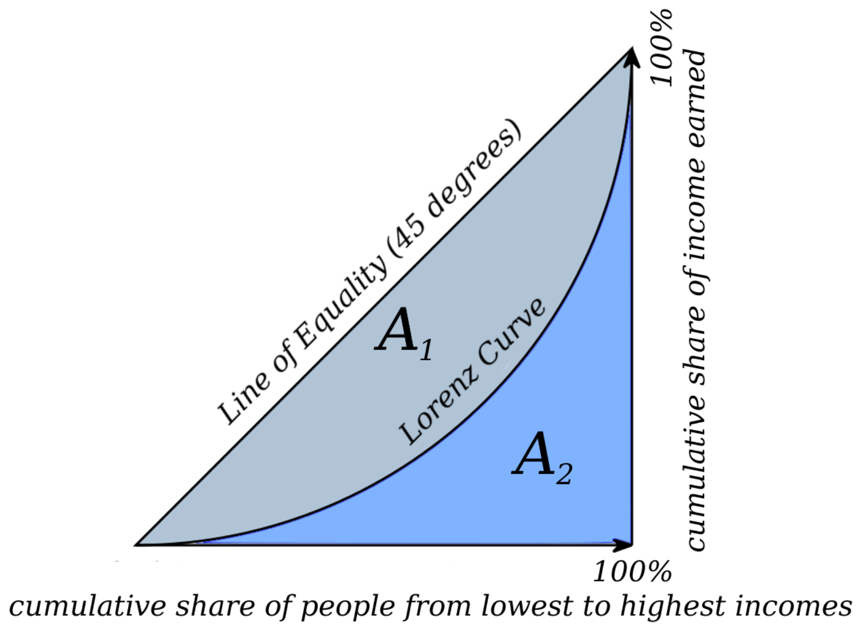
3.4.2 IR System Configuration

After generating a suitable query set, the system under evaluation must be configured. This process involves selecting a model and determining the hyper-parameters associated with the model that align with the analytical requirements. Researchers frequently conduct a retrievability analysis by sweeping the length normalization parameter of a given retrieval algorithm [Azzopardi and Vinay, 2008b; Wilkie and Azzopardi, 2013a; Wilkie and Azzopardi, 2013b]. This stage is elementary and primarily specifies the instantiating the system that is being evaluated. Once the system is set up, queries are issued one by one, which is a time-consuming step because of the large number of queries involved. The outcomes of each query are recorded, and most systems allow for setting a rank cutoff that specifies the depth of the ranking. Any documents ranked beyond the cutoff are disregarded. It is crucial to set the cutoff high enough to compute retrievability at the specified cutoff. As a result of this process, a ranked list of results is generated for each query in the query set used.

3.4.3 Calculating document Retrievability and global Retrievability

Upon recording the results of each query up to rank n, the process of calculating document retrievability r(d) commences. A decision must be made regarding the type of utility function to be employed to compute r(d), which is typically either cumulative or gravity based. In the case of the cumulative based measure, only the cutoff needs to be determined, specifying the rank at which a document must appear in the rankings to accumulate more r(d) score. Conversely, in the case of a gravity based measure, a decay function must also be established in addition to the cutoff, which determines how much score is gained at each successive rank, thereby affecting the measure's sensitivity to results appearing further down the ranking. Research by Wilkie and Azzopardi has revealed that these measures are highly correlated [Wilkie and Azzopardi, 2013b], and that the selection and configuration of the function primarily impacts the magnitude of differences between r(d). This study also demonstrated that a variety of utility function configurations all concurred on the setting that minimized retrievability bias when tuning a system's length normalization parameter to minimize bias. Similarly, a study by Bashir and Rauber also confirmed Wilkie and Azzopardi's findings that a cumulative measure's cut-off had a minimal impact on the overall Gini Coefficient [Bashir and Rauber, 2010b]. Consequently, most subsequent studies report only one utility function's findings regarding r(d). Upon selection and configuration of the utility function, the retrievability score, denoted by r(d), is computed for every document in the collection. Subsequently, a list is generated, comprising of the document and its corresponding retrievability score(s). This compilation can facilitate the identification of documents with either excessive or inadequate retrievability [Bashir and Rauber, 2009a; Bashir and Rauber, 2009b]. Moreover, it may be employed to enhance the efficacy of pseudo relevance feedback [Bashir and Rauber, 2010a; Bashir and Rauber, 2010b].

A visual or graphical approach to examining the imparity of accessibility within document collections is to employ Lorenz Curves, a graphical representation



100%
cumulative share of income earned
Line of Equality (45 degrees)
A_1
Lorenz Curve
A_2
100%
cumulative share of people from lowest to highest incomes

Figure 3.1: Lorenz Curve illustrated [Image: Wikipedia, 2023a]

commonly utilized in economics to depict the distribution of income among populations, as demonstrated in figure 3.1.

This involves sorting the populace in an ascending manner based on their wealth, followed by the generation of a graph that displays the cumulative distribution. In a scenario where the distribution is equal, the line on the graph would be akin to the Line of Equality. The measure of deviation of the distribution from the Line of Equality is an indicator of its skewness. By substituting wealth with r(d), this concept can be implemented in the analysis of retrievability, allowing for the visualization of an IR system's bias.

The next follow-up step involves assessing the degree of retrievability bias in a system through the use of the Lorenz Curve is by estimating the *Gini Coefficient*. However, Wilkie and Azzopardi investigated the impact of alternative inequality metrics on retrievability bias calculations, since Gini Coefficient is just one of many inequality metrics available [Gastwirth, 1971; Wilkie and Azzopardi, 2015]. Prior to their study, Gini Coefficient was the only metric used to determine overall retrievability bias. Therefore, the objective of their research was to determine whether Gini Coefficient was a suitable metric, and whether other metrics offered additional perspectives on bias. Using three parameterized retrieval models (BM25, PL2, and LMD) on the Aquaint and .Gov collections, Wilkie and Azzopardi calculated retrievability scores for each model and setting, then estimated system retrievability bias using six inequality metrics. The authors found that these metrics mostly agreed on which system and other settings minimized retrievability bias, although there were differences in magnitude. Specifically, the Palma Index [Palma, 2011] and the 20:20 Ratio highlighted the magnitude difference between settings, but were consistent with the Gini Coefficient and other measures. These findings were replicated across the combinations of retrieval models and document collections. Therefore, the authors recommended the continued use of the Gini Coefficient to summarize retrievability bias, as the other metrics explored provided no compelling reason to do otherwise. This work supported Azzopardi and Vinay's choice of inequality metric [Azzopardi and Vinay, 2008b], and indicated that prior research was not compromised by its use of the Gini Coefficient.

The *Gini-Coefficient* G serves as a concise measure to encapsulate the information presented by the Lorenz Curve. From the figure 3.1, G can be expressed as the ratio of the regions enclosed by the curve, as written in equation 3.2. This value can be determined using equation 3.3, where D represents the total number of documents included within the collection [Gastwirth, 1971].

$$G = \frac{A_1}{A_1 + A_2} \tag{3.2}$$

$$G = \frac{\sum_{i=1}^{|D|}(2 \cdot i - |D| - 1) \cdot r(d_i)}{(|D| - 1)\sum_{j=1}^{|D|} r(d_j)} \tag{3.3}$$

The value of G lies within the range of 0 to 1. A value of 0 indicates an absence of bias and implies that all documents have an equal chance of being retrieved. Conversely, as the value of G increases, the level of bias in the retrieval system towards the document collection utilized also increases, as stated in [Azzopardi and Vinay, 2008b].

Chapter 4

Retrievability Experiments on TREC 678 corpus

The main objective of the first semester's experimentation for the masters thesis is to investigate the retrievability bias for standard retrieval models using an improved query set, then look into retrievability disparity between documents in Relevance Judgement and documents otherwise, and finally explore the impact of query expansion on retrievability of collection documents.

For the reproducibility and comparison with the original results of Azzopardi and Vinay (2008) [Azzopardi and Vinay, 2008b], three different retrieval models are chosen for the retrievability experiment which were part of their study as well. To check the consistency of result for the retrieval models across collections, a different corpus is selected. Selection of the corpus is partly based on availability of topic file and relevance judgement, and similarity of the corpus with the corpus used by Azzopardi and Vinay (2008) [Azzopardi and Vinay, 2008b].

The retrievability values of documents is used to investigated if the documents selected for the relevance judgement is more retrievable than rest of the documents. This can reveal bias in the relevance judgement. Finally, a selection of query expansion method is made to do retrievability analysis on, to find out whether query expansion increase or decrease retrieval bias.

This chapter covers the main experimental work performed in first semester of this thesis and works on two of the goals are covered, goal-1 and goal-2. All the methods, experiments and results are presented in this chapter itself.

4.1 Experimental Setup

4.1.1 Hardware and Operating System

For the experiments as well as document preprocessing, indexing, evaluations and any other experiment related computation, two different systems with following main specifications are used:

Desktop PC

- Intel Core i7-12700 12th Gen @ 2.1GHz
- 16 GB main memory
- Ubuntu 22.04.1 LTS (Jammy Jellyfish) - 5.15.0-56-generic Kernel

IISER Kolkata Dirac Supercomputer

- Intel Xeon Gold 6148 CPU @ 2.40GHz
- 128 GB main memory
- Rocks 7.0 Manzanita (CentOS 7.4)

All additional software and packages used are identical:

- Python 3.10
- PyLucene 8.8.1
- trec eval 9.0.7
- Pyserini 0.12.0

The packages listed above are not exhaustive and several other python packages are used, which will be mentioned wherever they will be used. Details about the use of these packages and any particular settings will be discussed along with the experiment details in their respective subsections.

4.1.2 Dataset

TREC style collection is especially suitable because they feature Topics and their relevance judgement Qrels, which is required to do a contrast study between retrievability r(d) values of judged documents and rest of the documents.

The collection selected is the document collection used in TREC 2004 Robust Track, often referred to as **TREC 678** corpus. The document collection for the Robust track is the set of documents on both TREC Disks 4 and 5 minus the the Congressional Record on disk 4. [1]

Source	# Docs	Size (MB)
Financial Times	210,158	564
Federal Register 94	55,630	395
FBIS, disk 5	130,471	470
LA Times	131,896	475
Total Collection:	528,155	1904

The size of the document collection is close to 2 GB with 528,155 text documents. Vocabulary size of this corpus is close to 1.5 million.

For preparing the corpus for retrievals, it needs to be indexed first. Before that, document IDs (DOCID) are identified using the XML tags and contents of a document is rest of the text in the document except the XML tags. Now having the contents of the documents and their doc-ids, Lucene (PyLucene[2]) is used with the analyzer set as “EnglishAnalyzer” (which performs basic text preprocessing and porter stemming) to index the corpus. Resulting index of the corpus created by lucene is of the size 1.9 GB (notice that the corpus size was also about 1.9 GB) and can be easily accessed using Lucene’s IndexReader and IndexSearcher classes.

[1] https://trec.nist.gov/data/robust/04.guidelines.html

[2] PyLucene is a python wrapper around the Java Lucene (https://lucene.apache.org/pylucene/). Lucene is a Java library which provides indexing and search features, as well as other related advanced functionalities (https://lucene.apache.org/)

4.1.3 Retrieval Models

TFIDF, Okapi BM25, Language Model with Dirichlet Smoothing (LMDir) with $\mu = 1000$ are selected for retrievability study of standard retrieval models, as these retrieval models were also used by Azzopardi and Vinay in their 2008 research article on Retrievability Azzopardi and Vinay, 2008b.

TFIDF is an old and classic IR model with no hyper-parameters. It is often included in studies alongside with other models for standard comparison.

BM25 builds upon TFIDF way of scoring with better term-frequency and document length normalization and is considered a strong baseline. BM25 has two paramters: k_1 and b. Optimal paramters is selected from evaluations against 250 topics of 678-robust: $k_1 = 0.7$, $b = 0.35$.

LMDir Language Modelling using Dirichlet Smoothing is one of commonly used IR models from language model retrieval functions. LMDir has a smoothing parameter μ that is selected to be $\mu = 1000$ in order to match with the prior study Azzopardi and Vinay, 2008b.

All three retrieval algorithms are pre-implemented in Lucene. Therefore, respective Lucene functions from *Similarities* class are used to perform the retrievals for these retrieval models.

4.1.4 Model Effectiveness

Performance of each retrieval model on TREC 678 corpus with respective TREC topics 6,7,8 and robust (250 queries) is evaluated using *trec eval*. In the table 4.1, a few key performance metrics for the 3 algorithms are reported.

TREC 678	MAP	MRR	P@5	P@20
TFIDF	0.1561	0.5257	0.3598	0.2488
BM25	0.2596	0.6766	0.4988	0.3679
LMDir	0.2526	0.6774	0.4747	0.3600

Table 4.1: Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), Precision at 5 and 20 documents (P@5, P@20) with respect to their TREC queries

TFIDF performs significantly poorer than BM25 and LMDir. BM25 and LMDir provides equivalent performance for TREC 678 corpus with respect to the same testing query set. Later sections in the chapter show that despite similar effectiveness, BM25 and LMDir differ in terms of their retrieval bias.

4.2 Large Scale Retrieval Simulation

Estimation of $r(\mathbf{d})$ values for each document require a number of approximations to be made. Same as the original study Azzopardi and Vinay, 2008b, generalized utility/cost fucntion $f(.)$ is taken as the simple binary function which just indicates the presence or absence of a document in top ranks with cutoff c and the query weight o_q to be equal and constant by setting $o_q = 1$. For each retrieval model, $r(\mathbf{d})$ values are computed over all the documents in the collection for 5 different rank cutoffs: $c = 10, 20, 30, 50, 100$.

The next choice is the set of queries to be used. Since TREC collections are test collections, availability of a user query log is not possible. Therefore, a set of artificial queries need to be generated by some means. Azzopardi and Vinay (2008) Azzopardi and Vinay, 2008b, for creating their query set, used uni-grams and bi-grams sampled from the documents in the collection. All unigram queries were terms in the vocabulary which occurred at least 5 times. All two term queries were bigrams which occurred at least 20 times (if number of bigrams after frequency-thresholding more than 2 million, bigrams are sorted by their frequencies and the list is truncated at 2 million). Union of these two exclusive subsets of one term and two term queries was their query set.

Using the above method, queries were generated for TREC 678 collection but were found to be consisting a lot of undesirable terms, which are unlikely to be issued by a user to an IR system, creating a noisy query set. Therefore, for constructing a set of more realistic[3] queries, a modified method is employed to filter out undesirable queries as much as possible and will be discussed in the next subsection in detail.

4.2.1 Modified Query Generation Method

Types of queries (one term and two term queries), occurrence thresholds, and truncation prescription is kept same as done by Azzopardi and Vinay (2008) Azzopardi and Vinay, 2008b without exploring the impact of variation in these choices to query set and retrievability analysis results.

Unigram Query Generation Method

Following are the steps:

1. All the tokens from lucene index of TREC 678 corpus is taken and all non-alphabetical tokens are removed, giving only alphabetical words.
2. All words are then converted to lowercase and stopwords[4] are removed.
3. This is the main step responsible for filtering unlikely queries. Part-of-Speech tagging[5] is done on words and then only Nouns (tag 'NOUN') and uncategorized (tag 'X' for others) are selected and rest of the words with other tags are removed. The reasoning behind such selection is that nouns tend to represent majority of realistic queries and words such as of, which, its, would often do not add meaning to the query or the information need of the user. Below is a table of tags which are removed and their examples[6] to help put things in perspective.

[3]Here, a realistic query is such a query which looks close to what a real person might enter in a search engine. For example, a query "said to" is considered less realitic than a query "United Nations" . A more concrete notion could be derived from comparison with query logs, but is not done here.

[4]NLTK English Stopword set

[5]nltk.tag.pos_tag for universal tagset

[6]Taken from the NLTK book website: https://www.nltk.org/book/ch05.html

Tag	**Meaning**	**English Examples**
PRT	particle	*out, per, with, on, that, up, at, over*
ADJ	adjective	*high, new, special, good, big, local*
NUM	numeral	*fourth, 14:24, twenty-four, 1991*
ADP	adposition	*with, by, under, at, on, of, into*
VERB	verb	*would, told, playing, given, is, say*
DET	determiner, article	*no, the, a, every, which, some, most*
PRON	pronoun	*its, my, her, us, I, he, their*
CONJ	conjunction	*but, or, although, while, if, and*
ADV	adverb	*already, really, early, still, now*

4. Frequency of each unique word is counted and words with frequency less than 5 are removed.

5. Words with only one character i.e. all alphabets are removed from list.

6. Now, if the number of words left in the set are more than 2 million, words are sorted by their frequencies in descending order and truncated the list at 2 million.

All the above steps are followed for TREC 678 collection and the constructed set of words is considered to be the unigram queries, which will be posed to the IR system as one term queries. Some examples from the set of unigram queries is presented in Figure 4.1.

Bigram Query Generation Method

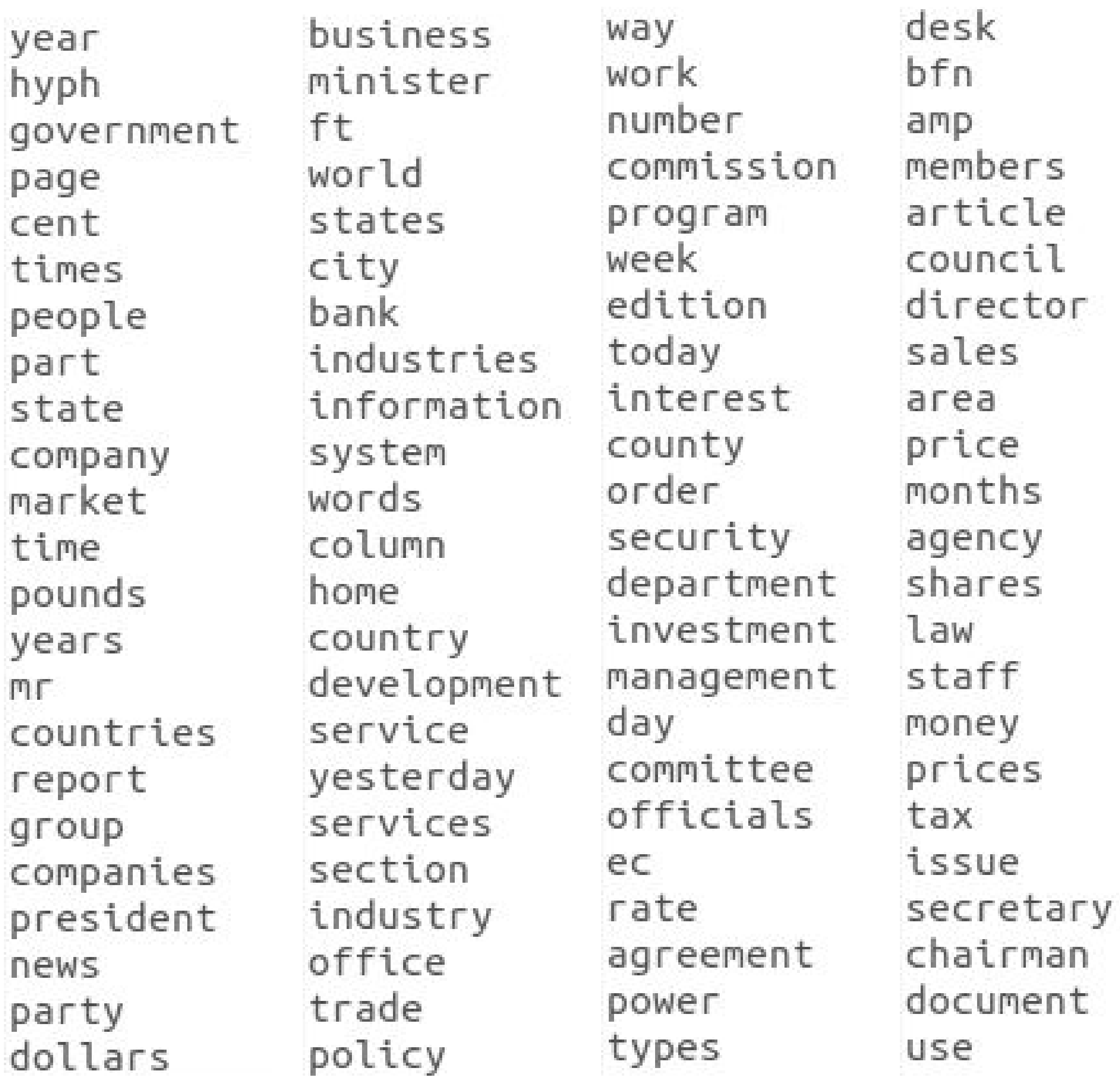
Following are the steps:

1. Content of each document are first blank-line tokenized[7] (to avoid two blank-line separated sentences, with the first sentence ending with no punctuation, getting considered one sentence) and then Punkt sentence tokenization[8] is done to get the sentences from each document.

2. Each sentence from all the documents are then word tokenized. Same as unigram query generation method, non-alphabetical tokens and stopwords are

[7]nltk.tokenize.regexp.blankline_tokenize

[8]nltk.tokenize.sent_tokenize



year	business	way	desk
hyph	minister	work	bfn
government	ft	number	amp
page	world	commission	members
cent	states	program	article
times	city	week	council
people	bank	edition	director
part	industries	today	sales
state	information	interest	area
company	system	county	price
market	words	order	months
time	column	security	agency
pounds	home	department	shares
years	country	investment	law
mr	development	management	staff
countries	service	day	money
report	yesterday	committee	prices
group	services	officials	tax
companies	section	ec	issue
president	industry	rate	secretary
news	office	agreement	chairman
party	trade	power	document
dollars	policy	types	use

Figure 4.1: Examples from the generated unigram query set.

removed. All the remaining words are lowercased.

3. Pairs of consecutive words from each sentences from all the documents in the collection are extracted as bigrams.

4. Again, as previously done for unigram query generation, Part-of-Speech tagging is done for both the words in bigrams and then the bigrams having both words tagged only as either 'NOUN' or 'X' is retained and rest of the bigrams are discarded.[9]

5. Bigrams whose one of the word is just single character (an alphabet) is removed.

6. Occurrence count of each bigram is done and bigrams with frequency less than 20 is removed.

7. Again, same as in the last step of unigram query generation method, if the number of bigrams left in the set are more than 2 million, bigrams are sorted by their frquencies in descending order and list is truncated at 2 million bigram queries.

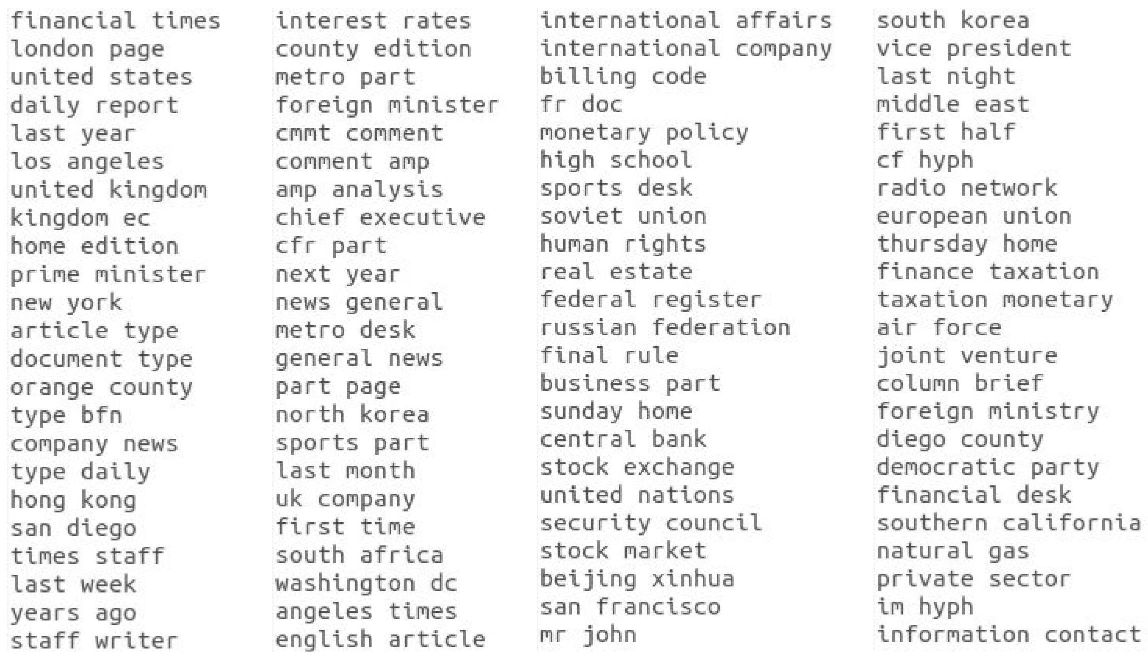
Following the above steps, bigram query set is generated from TREC 678 collection, which will be posed to the IR system as two term queries. Some examples of bigrams in this set is given in Figure 4.2.

Query Set

Unigram queries and bigram queries together form the query set that is used for retrievals. Below is the number of queries in the query set and in each subset:

	No. of queries
Unigram queries	137,029
Bigram queries	447,183
Query Set (Unigram + Bigram)	584,212

[9]When this step was omitted, bigrams like "said to", "company of" (which is equivalent to 'company' and is part of unigram queries anyway), "come here" appeared which are considered to be undesirable in this thesis.



financial times	interest rates	international affairs	south korea
london page	county edition	international company	vice president
united states	metro part	billing code	last night
daily report	foreign minister	fr doc	middle east
last year	cmmt comment	monetary policy	first half
los angeles	comment amp	high school	cf hyph
united kingdom	amp analysis	sports desk	radio network
kingdom ec	chief executive	soviet union	european union
home edition	cfr part	human rights	thursday home
prime minister	next year	real estate	finance taxation
new york	news general	federal register	taxation monetary
article type	metro desk	russian federation	air force
document type	general news	final rule	joint venture
orange county	part page	business part	column brief
type bfn	north korea	sunday home	foreign ministry
company news	sports part	central bank	diego county
type daily	last month	stock exchange	democratic party
hong kong	uk company	united nations	financial desk
san diego	first time	security council	southern california
times staff	south africa	stock market	natural gas
last week	washington dc	beijing xinhua	private sector
years ago	angeles times	san francisco	im hyph
staff writer	english article	mr john	information contact

Figure 4.2: Examples from the generated bigram query set.

4.2.2 Retrievals and document retrievability

A large scale retrieval simulation is conducted on the Desktop PC mentioned in Section 4.1.1 by posing all the queries from the above query set and retrieving top 100 documents for each retrieval model. From the retrieval results, frequency of each document for all the search results in top c = 10, 20, 30, 50 & 100 rank is counted which gives the estimate of $r(\mathbf{d})$ for each document $\mathbf{d}$ in the collection for each retrieval model. Low c threshold would represent web search by users more accurately, whereas, high c threshold would represent expert users in prior art search or precedent retrieval jobs.

4.3 Trends in document retrievability

First observation from examining the $r(\mathbf{d})$ values corresponding to any retrieval model, is that some documents have retrievability score disproportionately high, and on the contrast, many documents are not retrieved even once.

Following the Retrievability Analysis Framework described in Section 3.4, Lorenz curve and Gini-coefficient is used to study the inequality in retrievability scores in this huge collection of documents. Lorenz curve visualize the inequality whereas Gini coefficient summarises the inequality in one single metric.

In the following subsections, Lorenz curve and Gini coefficient of $r(\mathbf{d})$ for each three retrieval models (TFIDF, BM25, LMDir) and each rank cutoff values c (10, 20, 30, 50 & 100) for TREC 678 collection is presented.

4.3.1 Lorenz Curves

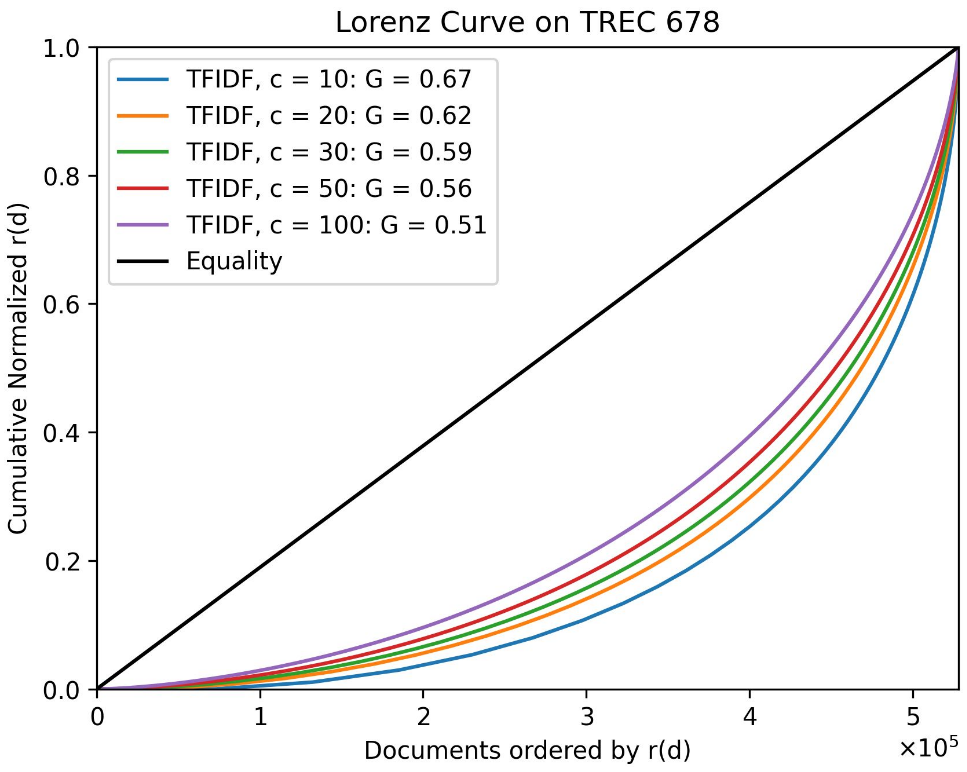


Figure 4.3: Lorenz Curve for TFIDF model.

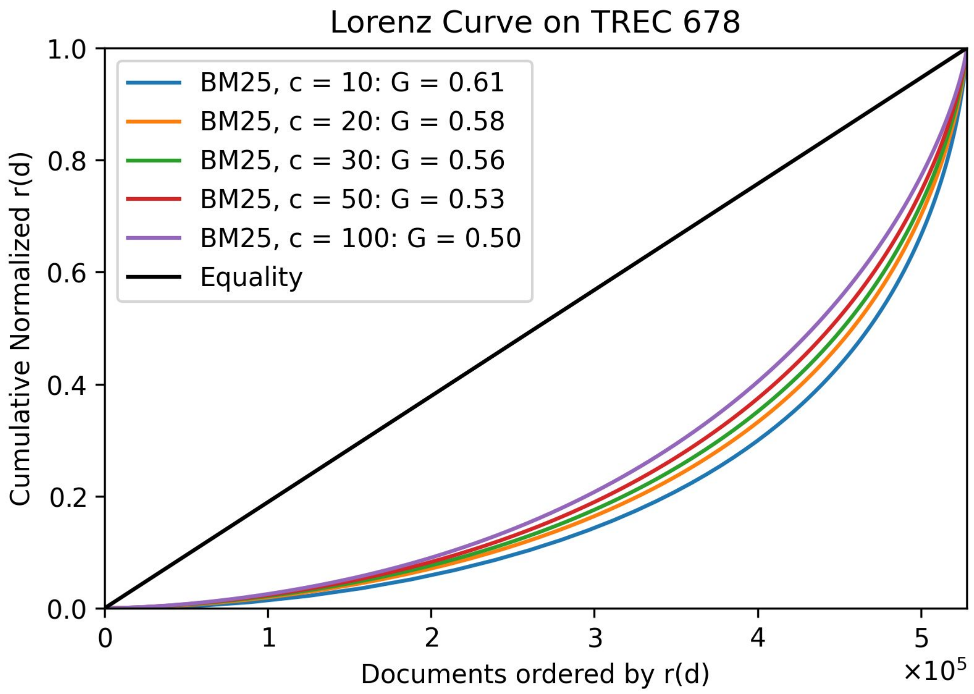


Figure 4.4: Lorenz Curve for BM25 model.

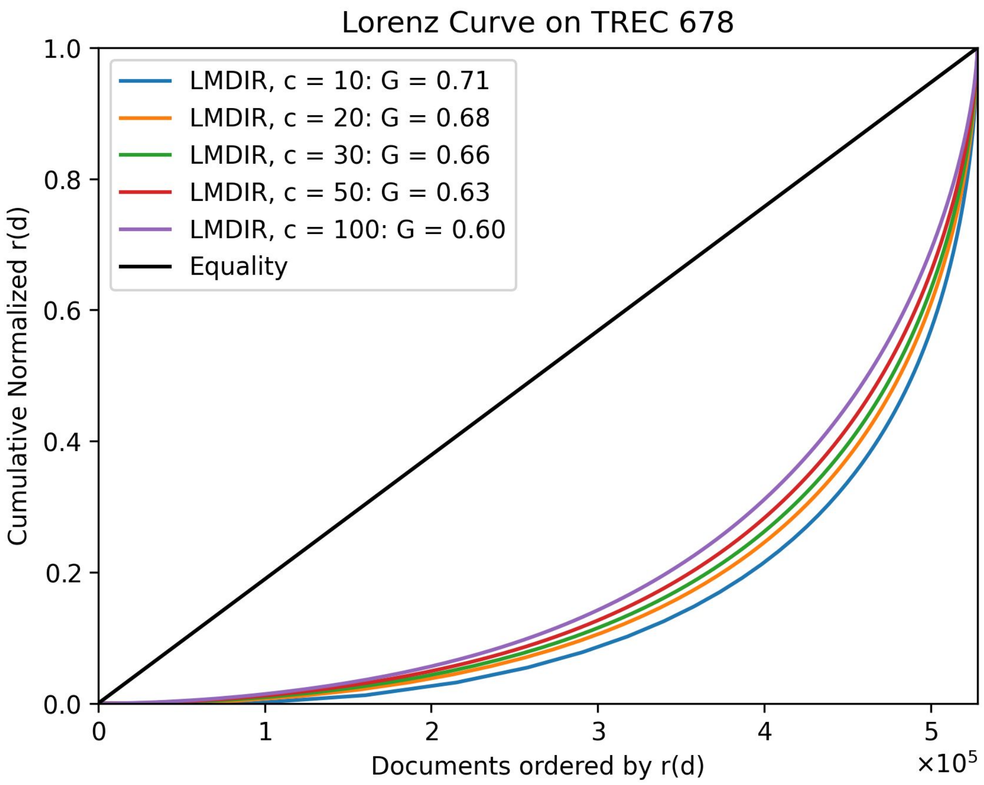


Figure 4.5: Lorenz Curve for LMDir model.

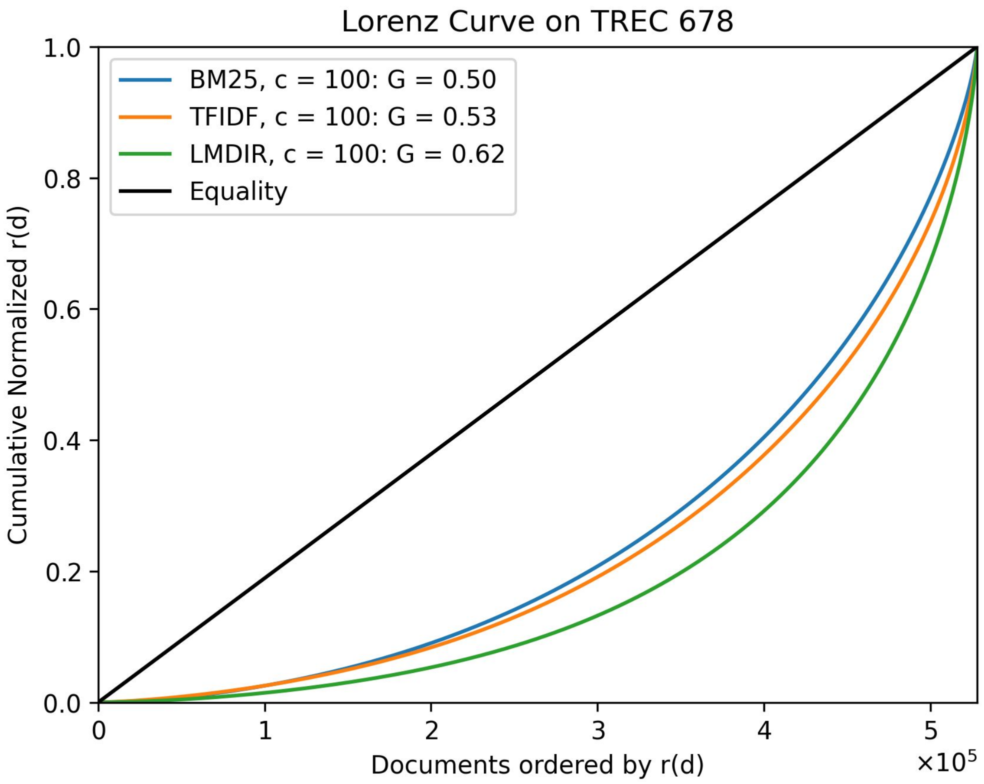


Figure 4.6: Lorenz Curve for TFIDF, BM25 and LMDir models for c = 100.

4.3.2 Gini Coefficient

Retrieval Model		c				
		10	20	30	50	100
TFIDF	G	0.67	0.62	0.59	0.56	0.51
	ρ		0.95	0.91	0.84	0.75
BM25	G	0.61	0.58	0.56	0.53	0.50
	ρ		0.97	0.95	0.92	0.87
LMDir	G	0.71	0.68	0.66	0.63	0.60
	ρ		0.98	0.91	0.88	0.85

Table 4.2: Gini coefficient for all the retrieval models with different values of c for TREC 678 collection. Pearson's correlation coefficient ρ is calculated between $r(d)$ values of $c = 10$ and all other values of c. The relationship between all the pairs of $r(d)$ values is found to be statistically significant, thus verifying the stability of retrievability measure with respect to the choice of c.

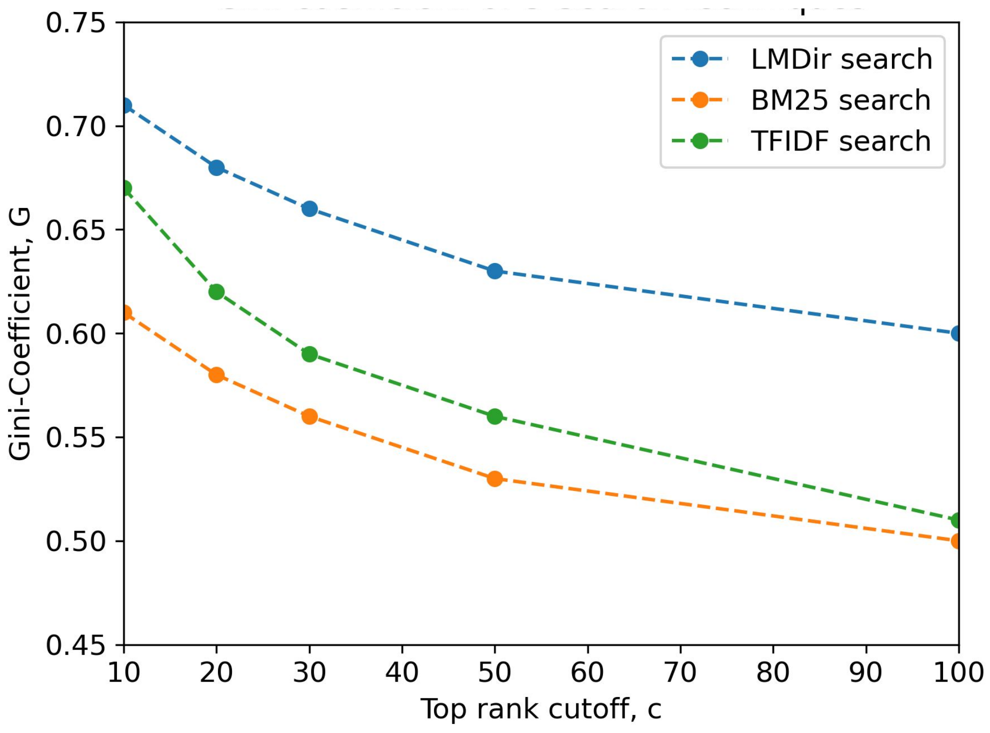


Figure 4.7: Plot of G vs c from the above table.

4.3.3 Observations

For all the retrieval models, Gini coefficient values slowly decrease as the value of c increase. This suggests that the exposure of user to the search result bias decrease as they look further down the rank list of search result. When $c = N$, where N is the no. of document in the collection, Gini coefficient becomes zero (absolutely no bias), because all the documents are retrieved all the time leading to equal $r(\mathbf{d})$ scores equal to number of queries. Whereas, if a user is only looking at a few top documents returned by a retrieval model, then the user is exposed to greater bias irrespective of the retrieval model.

BM25 is observed to induce least bias among the three retrieval models, whereas LMDir induced highest bias. Previously in Section 4.1, BM25 and LMDir was found to have equivalent performance; but now from looking at the Gini coefficient values of both models, BM25 and LMDir are not at all similar in terms of the inequality in retrievability of documents each of the model is inducing.

Another observation is that, as the c is increasing, Gini coefficient of TFIDF is catching up with the lower G value of BM25. For $c = 100$, Gini coefficient of TFIDF and BM25 is almost same, with BM25 still having a slightly lower value.

4.4 Bias in Relevance Judgement

The question is asked that whether the relevance judgement is also rigged with retrievability bias. That is to say, if the documents present in the relevance judgement have higher retrievability scores as compared to the documents which have not been included in the relevance judgement.

To address this question, relevance judgement of TREC 678 topics 678-robust (250 queries) is used. A set of document IDs present in the relevance judgement is formed, and the set of rest of the document IDs is considered as non-judged documents. Descriptive statistics of retrievability values $r(d)$ for documents in the relevance judgement and rest of the non-judged documents is calculated (see Table 4.3) and then inferences are drawn from them.

From the Table 4.3 and distribution plot , several observations can be made. First, even though the number of judged documents are about half of the number of

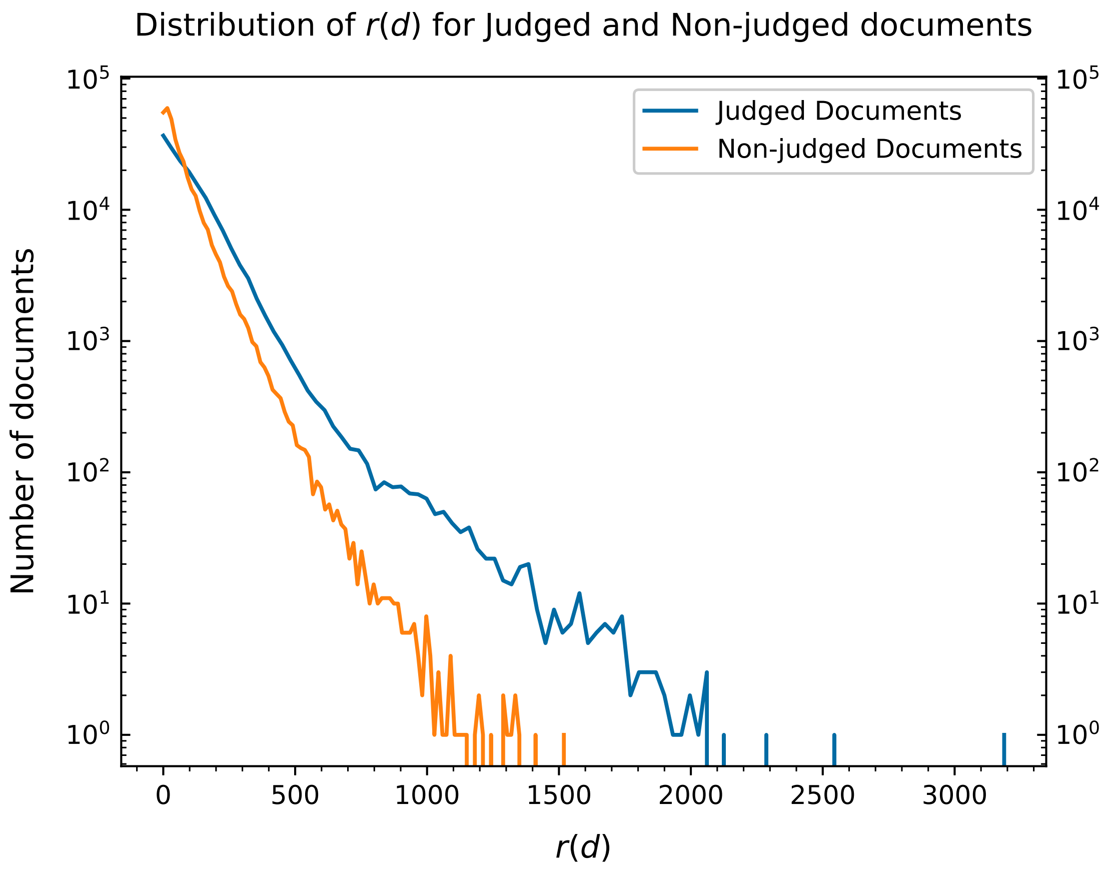


Figure 4.8: Plot of count of documents on log-scale with $r(d)$ values for set of judged documents and set of non-judged documents. It can be observed that the distribution for the judged documents in more dominated by numerous documents of higher $r(d)$ values, which clearly shows that the set of judged documents has predisposition towards higher $r(d)$ values.

	Judged documents	Non-judged documents
Count	174787	353368
Mean r(d)	**131.03**	**80.15**
Std r(d)	139.21	86.21
Min r(d)	0	0
Max r(d)	3220	1534
25%	39	24
50%	93	52
75%	177	106

Table 4.3: Descriptive Statistics for $r(d)$ values (for BM25 with $c = 100$) of Judged and Non-judged documents.

non-judged documents, mean $r(\mathbf{d})$ of judged documents are significantly higher than non-judged documents. This suggests that there are some documents in relevance judgement which have very high retrievability scores which can be responsible for increasing the mean. Max $r(\mathbf{d})$ of judged documents is the overall highest $r(\mathbf{d})$, whereas Max $r(\mathbf{d})$ of non-judged documents is almost half of Max $r(\mathbf{d})$ of judged documents. The minimum $r(\mathbf{d})$ of both categories are zero, suggesting that judged documents do not necessarily contain only those documents which are retrieved by BM25. From the percentile information, it can be said that, although distribution of $r(\mathbf{d})$ of both judged and non-judged documents start from zero, $r(\mathbf{d})$ values' distribution is right shifted, with the shift increasing with increasing $r(\mathbf{d})$.

The relevance judgement is tending to be biased towards more retrievable documents, irrespective of the queries. This is a cause of concern because retrieval model evaluations rely upon relevance judgements in the Cranfield paradigm. The favoritism of relevance judgement towards documents with higher retrievability scores means that retrieval models preferring highly retrievable documents will get higher evaluation scores, due to which not only a retrieval model with higher bias will be deemed better but also the performance metrics calculated using such relevance judgement will be inaccurate; subsequently, if such a retrieval model is used to pool documents for creating any new relevance judgement, then the bias in relevance judgements will add up over time. Therefore, this observation of retrievability bias in constituent documents of relevance judgement has implications for the document pooling strategies devised or used to create relevance judgements.

4.5 Retrievability after RM3 Query Expansion

Investigation of the impact of query expansion on retrievability of documents and overall retrievability bias is carried out for **RM3** query expansion technique.

Initial retrieval is done using BM25 (with k_1 and b parameters same as before). RM3 is used to expand the queries with top 10 docs as pseudo-relevant docs, 10 expansion terms and original query weight $= 0.4$. Re-retrieval is performed again using BM25 and final retrieval results are presented.

For large scale retrieval simulation for RM3, same query set is used as before. Due to longer retrieval time taken by RM3, retrievals are run on IISER-Kolkata

Dirac Supercomputer and document retrievability values are estimated for all the 5 *c* rank cutoff values (10, 20, 30, 50 & 100).

Lorenz curve is plotted and Gini coefficient is calculated for all *c* values.

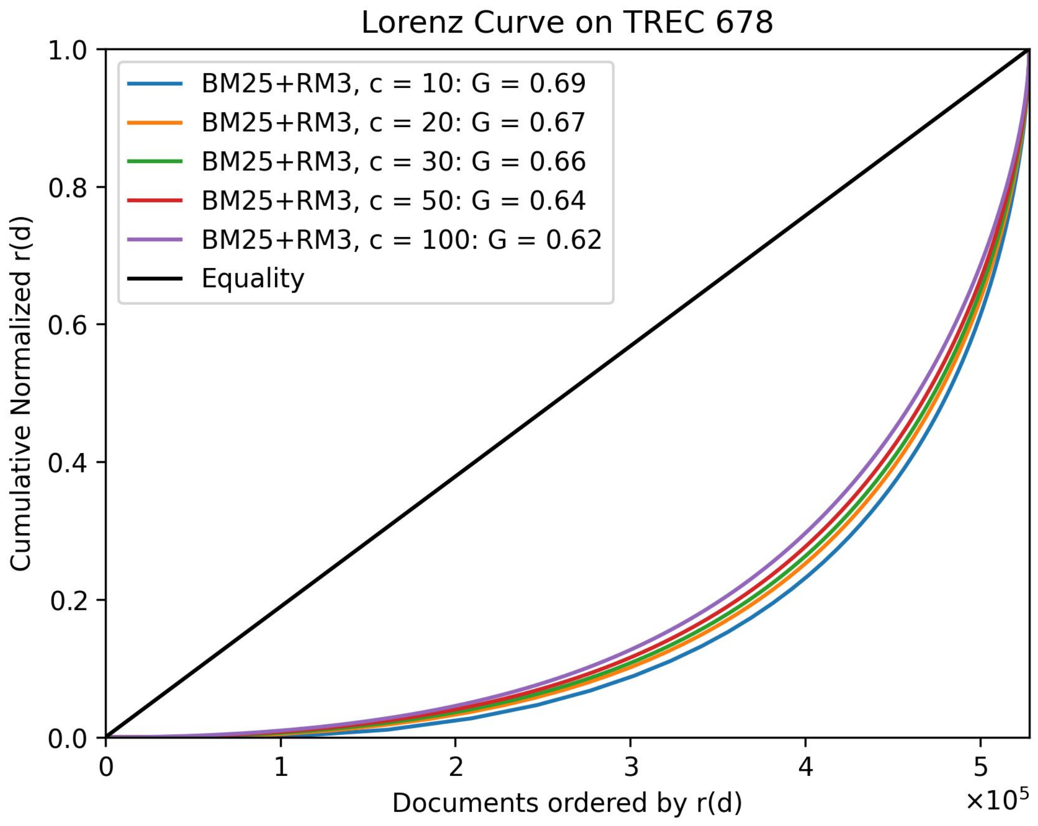
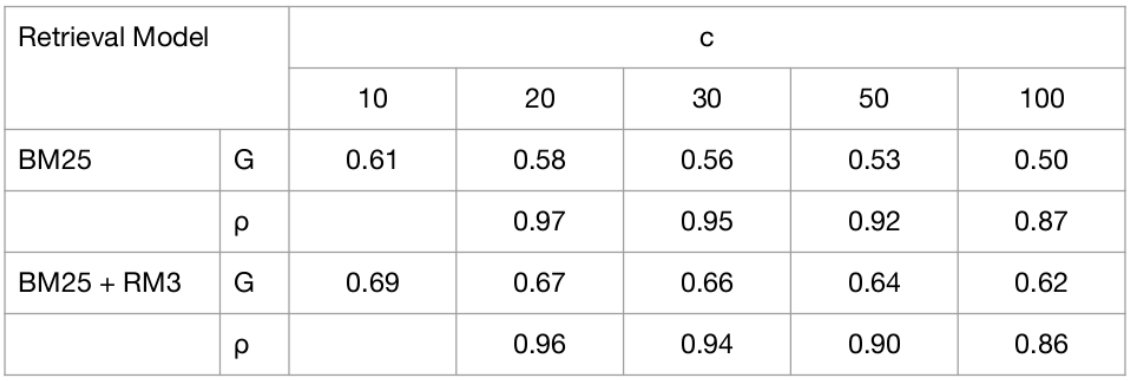


Figure 4.9: Lorenz Curve for BM25 + RM3 on TREC 678.



Retrieval Model		c				
		10	20	30	50	100
BM25	G	0.61	0.58	0.56	0.53	0.50
	ρ		0.97	0.95	0.92	0.87
BM25 + RM3	G	0.69	0.67	0.66	0.64	0.62
	ρ		0.96	0.94	0.90	0.86

Table 4.4: Comparison between Gini coefficient calculated for BM25 and BM25+RM3 on TREC 678. Pearson's correlation coefficient ρ is also reported in the similar combination as done in Table 4.2.

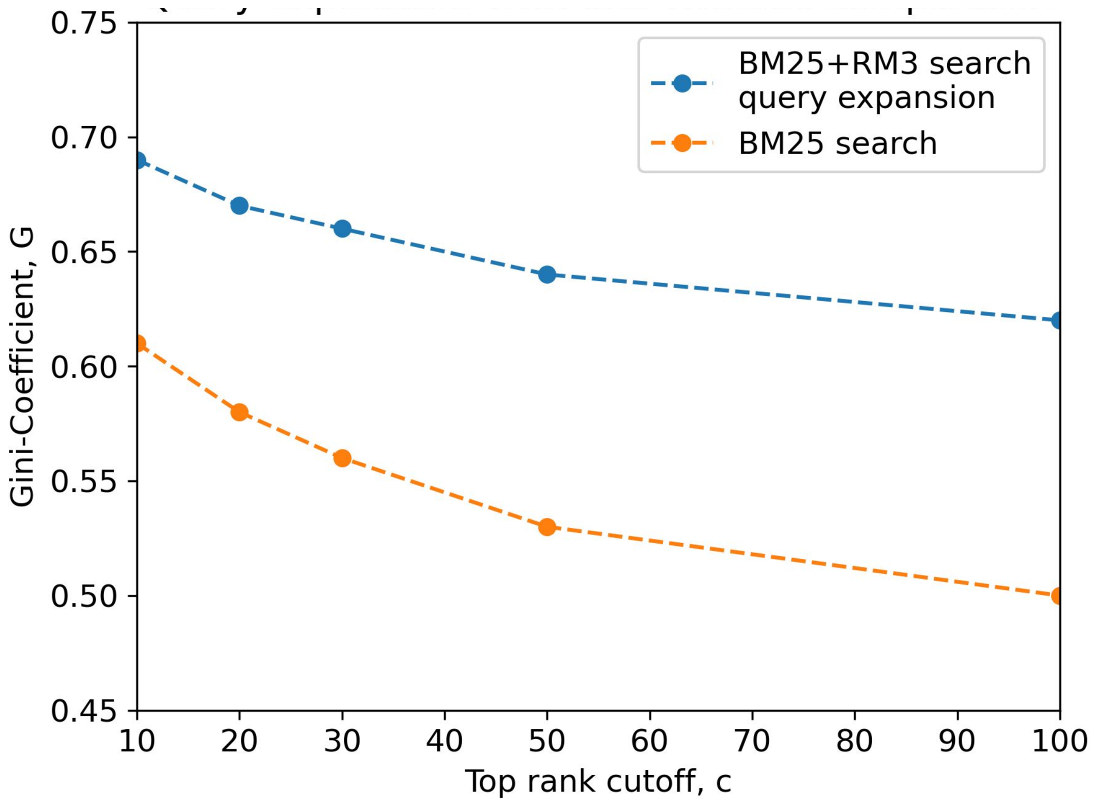


Figure 4.10: Plot of G vs c for BM25+RM3 search and BM25 search.

RM3 query expansion has increased retrievability bias. This suggests that the process of adding more terms from top documents is leading to reduction in retrievability of documents. Also, the decrease in Gini coefficient with c is slower for BM25+RM3 than BM25. Although the RM3 is known for boosting performance very well, the boost in retrievability bias that is coming along with it is concerning. This observation also highlights that increase in bias doesn't always correlate with decrease in performance, and hence the relationship between effectiveness and retrievability bias is non-trivial and not easily reducible to positive or negative correlation.

Chapter 5

Comparing Query Sets for Retrievability Analysis: Artificial vs. Query Log-Based Queries

This chapter focuses on conducting a retrievability analysis on two corpora using the BM25 retrieval model, employing different query sets. The first query set is constructed using the AOL query log, while the second query set is generated using the query generation method proposed by the original proposers of the retrievability measure [Azzopardi and Vinay, 2008b] due to its widespread use in retrievability literature. Additionally, a modified approach is employed to construct the last query set, building upon the aforementioned method [Azzopardi and Vinay, 2008b]. Subsequently, a large-scale retrieval simulation is performed for each query set, and retrievability scores are computed for all documents in the two collections corresponding to those query sets.

The objective is to determine the correlation between the retrievability scores obtained from artificially generated query sets and those derived from the query set constructed from the query log. This analysis aims to ascertain whether artificially generated queries yield similar retrievability scores to query log-based retrievability scores, and whether artificial queries can serve as an acceptable substitute for real user queries in retrievability analysis.

The results indicate a low or almost non-existent correlation between the retrievability scores obtained from artificially generated queries and those derived from the query log. Consequently, it can be inferred that artificially generated queries do not yield retrievability scores similar to those derived from a query log. Furthermore, the proposed modified approach demonstrates a higher correlation with the query log-based retrievability scores, thereby representing a contribution towards a more acceptable query generation method for retrievability analysis on IR systems in situations where a historical query log is not available.

In summary, this chapter presents a retrievability analysis conducted on two corpora using different query sets, employing the BM25 retrieval model. The study compares retrievability scores derived from artificially generated queries with those obtained from a query log, aiming to determine the suitability of artificial queries for retrievability analysis. The findings reveal a lack of correlation between the retrievability scores obtained from artificially generated queries and those derived from the query log. Moreover, the proposed modified approach exhibits a higher correlation with the query log-based retrievability scores, indicating its potential as a more acceptable query generation method in the absence of a historical query log for retrievability analysis on IR systems.

5.1 Dataset

We utilized two distinct datasets for our research: English Wikipedia articles and the TREC WT10g collection. Additionally, we incorporated a publicly released three-month query log from AOL dating back to the year 2006.

The English Wikipedia articles were obtained from a 2023 dump, providing us with a larger and more recent dataset. This choice was made to ensure that we had access to up-to-date information and a comprehensive representation of knowledge present on the English Wikipedia platform.

In contrast, we included the TREC WT10g collection as it serves as an older web corpus compared to the AOL query log. By incorporating this dataset, we aimed to investigate whether the findings remain consistent when examining a smaller and older corpus in comparison to a larger and more recent dataset. It is worth noting that a specific overlap filter, which will be discussed later in the section 5.3, was applied to ensure appropriate alignment between the corpus and query log.

The inclusion of both datasets allows us to explore potential variations in correlations while accounting for the disparities between a modern, extensive corpus and an older, more limited dataset. By establishing similar correlations across these datasets, we can infer that our results are robust to the differences in corpus characteristics and query logs.

Next, we will elaborate on the procedures and methodologies employed to preprocess the selected datasets in order to extract required texts and draw reliable conclusions in our study.

5.2 Corpus Preparation

In order to utilize the available corpora effectively for indexing and generating queries, it is necessary to perform a process of cleaning and preprocessing. The original form of both corpora is not suitable for direct use as plain English texts due to the presence of noise and other inconsistencies. Therefore, in this section, we will describe the specific preprocessing steps carried out for each collection. By undertaking these necessary preparations, we aim to obtain noise-free English texts that can be used for indexing purposes, as well as for generating queries from the artificial query set. The subsequent subsections will provide detailed explanations of the preprocessing methods employed for each corpus.

Wikipedia

Extracting plain text from Wikipedia dumps poses a significant challenge due to the presence of wiki markdown syntax, Lua modules, and other complex syntactic structures. Parsing through these intricacies to obtain the plain text content of each article requires robust preprocessing techniques.

A commonly used tool for extracting plain text from regular Wikipedia dumps is `attardi/wikiextractor`, which is publicly available on the Github; also other such extractors are available: `Wiki2Plain`, `WikiPrep`, `wikipedia2text`, `mwparserfromhell`, `tatuylonen/wikiextract`, `libmwparser` and others listed at mediawiki.org/wiki/Alternative_parsers. However, even with the aid of `wikiextractor` tool, the resulting output still contains considerable noise and omits certain text contents due to inherent limitations in the extractor. Another type of Wikipedia dump utilized by Cirrus Search of Wikipedia itself is already extensively parsed and cleaned. Nevertheless, this dump lacks representative delimiters to indicate the end of a heading or the start of a new paragraph, resulting in its own limitations. Consequently, sections such as 'References' and 'See also' are included in the extracted text, despite not belonging to the main content of the article.

To overcome these challenges and obtain the cleanest possible plain text extracts from Wikipedia, we adopt an innovative approach. We combine both types of unclean plain text extracts in a novel manner, leveraging one extract to eliminate noise from the other extract and vice versa. By integrating these two sources, we achieve a refined and noise-free plain text extract of Wikipedia articles. This approach arguably yields the cleanest extraction method's outputs when compared to those of other publicly available extractors.

The method involves first cleaning Cirrus Wikipedia dump for equations `{\displaystyle...}`, `[...]`, `{{...}}` and navigation text from title. Then wikiextractor's extracted output was used to insert headings into cleaned Cirrus output and remove texts starting sections like 'References' and 'See also' which are not desired to be the part of the final clean plain text of the articles. The implementation details are available for open public use and reproducibility purposes at the Github repository of this project. The final clean output of the Wikipedia articles were indexed using PyLucene with `EnglishAnalyzer()`.

WT10g

For the web collection WT10g, all the tags `<...>` and `</...>` were removed from the documents within `<DOC>` and `</DOC>`. The cleaned output of the WT10g documents were indexed using PyLucene with `EnglishAnalyzer()`.

5.3 Real Query Set: AOL query log

The AOL query log refers to the dataset containing the search queries made by users on the AOL search engine. It was a collection of anonymized user search data collected over a three-month period in 2006. The AOL query log holds significant advantages for the IR community. As one of the largest and most comprehensive collections of search queries, it provides valuable insights into user behavior and search patterns. Researchers and practitioners in the IR field can leverage this dataset to analyze user intent, study query reformulation strategies, and improve search engine algorithms. The AOL query log offers a rich source of data for training and evaluating information retrieval models, enabling researchers to develop more accurate and effective search techniques. Additionally, it allows for the identification of emerging trends and the discovery of novel research directions, enhancing the overall understanding of user information needs and preferences. In essence, the AOL query log serves as a valuable resource that fuels advancements in the IR community and contributes to the development of more efficient and user-centric search systems.

To prepare the real query set, the initial step involved extracting all unique queries from the AOL query log. Queries containing periods were eliminated to effectively filter out website links. Subsequently, to ensure that only queries answerable by the selected collection were included, queries were selected if all of their terms were present in the corpus vocabulary. Upon analysis, it was observed that queries with 20 or more tokens often comprised a substantial number of nonsensical queries. Consequently, only queries with a token count below 20 were considered for inclusion. These refined criteria resulted in the formation of the final query set consisting of authentic queries. Table 5.1 presents the distribution of query token counts and the corresponding number of queries for the final filtered queries sourced from the AOL query log. Additionally, the table provides the total number of queries encompassed within the final real query set.

5.4 Artificial Query Sets: Query Generation

The process of generating a comprehensive query set, denoted as Q, can be theoretically defined as an exhaustive enumeration of all possible queries for a given collection. However, due to the immense size of such a set, it becomes practically infeasible to perform retrieval simulations using the available computational resources. Moreover, the automatic generation of all potential queries for a collection remains an unsolved problem in the existing literature. To address this challenge, researchers [Azzopardi and Vinay, 2008b] in the field of retrievability measures have

Number of Tokens in the Query	**Frequency**	
	Wikipedia	**WT10g**
1	254647	236684
2	1411221	1384250
3	1519506	1506522
4	861411	855424
5	336884	334607
6	112751	111882
7	35281	34986
8	11585	11498
9	4282	4218
10	1693	1675
11	788	784
12	412	415
13	184	181
14	115	122
15	61	66
16	68	75
17	40	44
18	36	38
19	11	12
Total	**4550976**	**4483483**

Table 5.1: Distribution of Query Token Counts in the Final Filtered Queries from AOL Query Log

proposed an alternative approach wherein they strive to generate queries for the collection automatically. This query generation method has since been widely adopted in subsequent studies [Bashir and Rauber, 2009a; Bashir and Rauber, 2009b; Pickens et al., 2010; Wilkie and Azzopardi, 2013a; Wilkie and Azzopardi, 2014; Wilkie and Azzopardi, 2015], enabling the construction of query sets used for computing retrievability scores. In this study, we aim to investigate the impact of using artificially-generated query sets, as opposed to real query sets, on the retrievability scores obtained. By examining these differences, we can gain insights into the efficacy of artificial query sets in substituting real query sets for retrievability evaluation purposes. For this study, we generate a query set using original method

[Azzopardi and Vinay, 2008b] and then another artificial query set using a modified method newly proposed in this thesis.

5.4.1 Azzopardi's Query Generation Method

One commonly used approach for generating queries from a text collection is the Azzopardi's Query Generation Method, which involves the extraction of bigrams followed by ranking and selection of the top bigrams to create a comprehensive query set. This method was originally introduced by Azzopardi and Vinay in 2008, building upon the approach outlined by Callan and Connel in 2001 [Azzopardi and Vinay, 2008b; Callan and Connell, 2001].

The initial step in Azzopardi's Query Generation Method is the extraction of bigrams from the text collection. This is accomplished by implementing a sliding window technique that traverses the text, identifying pairs of consecutively occurring terms. Each bigram that appears a certain number of times is saved for further processing. The specific thresholds for determining the minimum frequency of appearance may vary depending on the application and dataset.

Once the bigrams have been extracted, they are then ranked using their frequencies. After the ranking stage, the top x bigrams are selected to form a sizeable query set. The value of x depends on various factors such as the size of the collection, the intended purpose of the queries, and the available computational resources. The selection of the top bigrams ensures that the generated queries encompass the most relevant and significant terms or combinations of terms within the collection.

In the original implementation by Azzopardi and Vinay, the query set consists of both single-term queries and bi-term queries. Single-term queries are constructed by taking each term in the vocabulary that occurs at least five times and posing the term itself as a query. On the other hand, bi-term queries are constructed by considering every pair of consecutively occurring terms (bigrams) in the collection that occur at least 20 times. It is worth noting that the list of bigrams is typically truncated at a certain threshold to control the size of the query set. In the specific case mentioned, the truncation limit is set at 20 million.

Azzopardi's Query Generation Method offers a systematic approach to generate queries from a text collection. By leveraging the extraction of bigrams, frequency ranking, and selection of top bigrams, this method provides a means to create a comprehensive query set. The method can be customized and adapted by adjusting the frequency thresholds, and the number of selected bigrams, allowing for flexibility in its application.

In summary, Azzopardi and Vinay uses ‘Query-based Sampling’ Method for query generation in the following manner [Azzopardi and Vinay, 2008b]:

- All Unigrams with $tf \geq 5$
- 2 million Bigrams with $tf \geq 20$

These unigrams and bigrams remaining upon filtration are used as the query set for retrievability analysis.

5.4.2 Improved Query Generation Method

The identification of collocations within a text corpus can be achieved through a straightforward counting approach, whereby the frequency of co-occurring word pairs is examined. This technique provides evidence of words that possess a distinct function, one that extends beyond a mere combination of their individual meanings. However, the process of selecting the most frequent bigrams alone often yields uninteresting results. Upon closer inspection, it becomes apparent that a substantial portion of these bigrams consists of function words, offering limited insights. To enhance the quality of identified collocations, a simple yet effective heuristic was proposed by Justeson and Katz [Justeson and Katz, 1991]. This heuristic involves subjecting candidate phrases to a part-of-speech filter, which selectively retains patterns likely to represent genuine “phrases” rather than random word combinations. Justeson and Katz specifically recommend certain tag patterns for both bigrams and trigrams, enhancing the meaningfulness of the resulting collocation identification process.

Therefore, to enhance the process of identifying collocations, we propose to integrate Justeson and Katz’s part-of-speech filtering method into the query generation approach. By incorporating this technique, we aim to selectively extract the most "query-like" N-grams, thereby improving the relevance and effectiveness of the generated queries. Our improved method introduces the following additional steps to the existing procedure:

1. Perform Part-of-Speech (POS) tagging on the collection documents: Initially, we employ POS tagging on all the documents within the collection. This step assigns appropriate grammatical tags to each word, facilitating the subsequent identification of N-grams.
2. Extract N-grams from the POS-tagged documents: N-grams, where N represents the desired length of the word sequences, are extracted from the POS-tagged documents. In our case, we consider N to range from 1 to 4, enabling the identification of unigrams, bigrams, trigrams, and quadgrams.
3. Select N-grams with "query-like" POS tag patterns: From the pool of extracted N-grams, we apply Justeson and Katz's recommended POS tag patterns to filter and retain N-grams that exhibit patterns resembling queries. The specific POS tag patterns for each N-gram type are provided in Table 5.3. Tag patterns for Quadgrams are proposed by us heuristically from our observations.

Subsequently, the resulting list of N-grams is sorted in descending order based on their occurrence frequencies. To ensure a manageable and relevant set of queries, we truncate the list at specific thresholds. These thresholds are determined by drawing inspiration from the frequency distribution of queries found in the AOL query set. We aim to maintain a proportional ratio between the selected N-grams and the query frequencies observed in the AOL based real query set, thus preserving a close alignment with real-world query usage patterns. Refer to the table 5.2 for the number of queries taken from each N-grams.

N-grams		Unigrams	Bigrams	Trigrams	Quadgrams	Total
Frequency	**Wikipedia**	250,000	1,500,000	1,500,000	1,300,000	4,550,000
	WT10g	250,000	1,400,000	1,500,000	1,350,000	4,500,000

Table 5.2: The frequency distribution for each N-grams in the Query Set created using Improved Query Generation Method

Table 5.3: POS Tag Patterns for N-grams

Unigram	**Bigram**	**Trigram**	**Quadgram**
NOUN	ADJ NOUN	ADJ ADJ NOUN	NOUN VERB ADP NOUN
	NOUN NOUN	ADJ NOUN NOUN	ADJ NOUN ADJ NOUN
		NOUN ADJ NOUN	NOUN ADP ADJ NOUN
		NOUN NOUN NOUN	NOUN NOUN ADP NOUN
		NOUN ADP NOUN	NOUN VERB NOUN NOUN
			ADV ADJ NOUN NOUN
			ADJ NOUN VERB NOUN
			NOUN ADJ NOUN NOUN

5.5 Retrievability Experiment

The retrievability experiment was conducted using the BM25 retrieval model [Stephen E. Robertson et al., 1992] on both corpora. The BM25 retrieval model is a widely used information retrieval model that has shown effective performance in various IR studies. For this experiment, default parameters were employed, namely $k1 = 1.2$ and $b = 0.75$, in accordance with common practice in the field.

Retrievals were carried out separately for queries from the three query sets summarized in the table 5.4. For each collection, the rank cutoffs were set at $c = 10, 20, 30, 50,$ and 100. Subsequently, the retrievability scores of all documents in the collection were computed for each query set. The cumulative scoring model of retrievability scoring was utilized to calculate these scores.

	Real Query Set	**Artificial Query Sets**	
	AOL queries	Azzopardi's method	Proposed method
Wikipedia	4,550,976	5,320,001	4,550,000
WT10g	4,483,483	3,902,575	4,500,000

Table 5.4: Number of queries in each query set for Wikipedia and WT10g collections.

5.5.1 Parallelization in Lucene

Apache Lucene is used to perform BM25 retrievals on a large query set consisting of millions of queries. Apache Lucene is an open-source information retrieval library written in Java. It provides powerful indexing and search capabilities, making it a widely used tool in various applications that require efficient and accurate searching of large collections of documents. Lucene forms the core of many popular search engines, including Elasticsearch and Solr. To speed up the experiment, the retrieval process is parallelized, resulting in significant speed improvements when utilizing a 20-core i7 processor machine.

The parallelization is implemented using Java's Fork/Join framework, specifically the `ForkJoinPool` class. The `ForkJoinPool` is created with the number of available processors minus one, allowing for efficient distribution of query retrieval tasks across multiple threads. The main steps involved in the parallelization process are as follows:

1. Loading the index and setting up the search environment:
    - The Lucene index is opened using the IndexReader and DirectoryReader classes, providing access to the indexed documents.
    - An IndexSearcher is created, which will be used to perform the query retrievals.
    - The BM25 similarity model is set on the IndexSearcher using the setSimilarity method, specifying the ranking formula parameters.
2. Splitting the query set into chunks:
    - The list of queries is loaded from a JSON file.
    - A method is used to divide the list of queries into smaller chunks, aiming to distribute the workload evenly across threads.
3. Parallel retrieval of query chunks:
    - A Progress Bar is initialized to track the progress of the query retrievals.
    - The query chunks are processed in parallel using the `parallelStream` method, which enables concurrent execution of the retrieval tasks.

- Each query in the chunk is processed by the retrieve method, which performs the actual retrieval using Lucene.
- The progress bar is updated after each query retrieval using the a step() method.

4. Saving the results:

   - After all query chunks have been processed, the `ForkJoinPool` is shut down and the program waits for the termination of all threads.
   - The retrieved document counts are stored in a `ConcurrentHashMap` called allRd, which keeps track of the retrieval results for different values of parameter c (from the cList).
   - The allRd object is serialized and saved to a JSON file using the Gson library, allowing for further analysis and evaluation.

This parallelization approach in Lucene provides significant speed improvements for processing large query sets. By distributing the retrieval tasks across multiple threads, the code takes advantage of parallel processing capabilities, reducing the overall execution time. The use of a `ForkJoinPool` enables efficient workload distribution and synchronization, ensuring the proper execution and completion of the parallel retrievals.

Anserini, a popular open-source information retrieval toolkit, also supports parallel retrievals to improve performance. Anserini utilizes a different method compared to the code provided. Instead of using the Fork/Join framework, Anserini leverages multi-threading using Java's ExecutorService and Callable interface. In Anserini, the retrieval process involves creating a pool of worker threads using the ExecutorService. Each worker thread is responsible for executing a retrieval task, which involves querying the index and processing the results. The ExecutorService handles the distribution of these tasks among the available threads. Both approaches have their advantages and are suitable for different scenarios. Anserini's method with ExecutorService provides greater control over thread management, making it more adaptable to complex retrieval scenarios. On the other hand, the Fork/Join framework used in the provided code simplifies the parallelization process and is well-suited for scenarios with uniform subtasks.

5.6 Correlation between Retrievability scores from different Query Sets

In this section, we investigate the correlation between retrievability scores obtained from two distinct query sets: the real query set based on the AOL query log, and the artificial query set generated using Azzopardi's method. Furthermore, we explore the correlation between the retrievability scores of all documents from the real queries and the scores derived from the artificial queries, employing a newly proposed modified method. To examine these correlations, we employ various correlation coefficients, which are presented in detail in result tables 5.5 and 5.6. By studying the relationships between the scores obtained from different query sets, we aim to gain insights into the robustness of the prior query generation method to act as an substitute to query log.

Collection	Pearson's correlation coefficient r	Spearman's rank correlation coefficient ρ	Kendall's rank correlation coefficient τ
Wikipedia	**0.2379**	0.5121	0.3603
WT10g	**0.1100**	0.5303	0.3731

Table 5.5: Correlation Coefficients between r(d) from AOL queries and Azzopardi's method generated artificial queries, both for c = 100.

Collection	Pearson's correlation coefficient r		Spearman's rank correlation coefficient ρ		Kendall's rank correlation coefficient τ	
	Q_{Azz}	Q_{new}	Q_{Azz}	Q_{new}	Q_{Azz}	Q_{new}
Wikipedia	0.2379	**0.3065**	0.5121	0.5893	0.3603	0.4194
WT10g	0.1100	**0.3135**	0.5303	0.5837	0.3731	0.4126

Table 5.6: Correlation Coefficients between r(d) from AOL queries vs Azzopardi method generated artificial queries (Q_{Azz}) and r(d) from AOL queries vs proposed method generated artificial queries (Q_{new}), both for c = 100.

5.7 Discussion

The findings of this study reveal a low correlation between the retrievability scores of documents in the collection when using real-world AOL queries compared to artificially generated queries using Azzopardi's query generation method. This indicates that Azzopardi's method should not be employed as an approximation for real-world queries in a retrievability analysis.

The low correlation between retrievability scores is an important observation, as it raises doubts about the effectiveness and reliability of using artificially generated queries to estimate document retrievability. Azzopardi's method, although theoretically sound, fails to capture the nuanced characteristics and complexities of real-world queries, leading to a discrepancy in retrievability scores. Consequently, relying solely on artificially generated queries may yield inaccurate estimations and compromise the estimate of retrieval bias.

To address this limitation, the study proposes an improved method for generating artificial queries that demonstrate an enhanced correlation with real-world queries in terms of retrievability scores. This improvement suggests that there is potential for refining query generation techniques to better align with the complexities inherent in real-world queries. By adopting the proposed method, researchers and practitioners can generate artificial queries that more accurately reflect the characteristics of real-world queries, leading to improved retrievability estimations due to higher correlation with r(d) from real queries.

While our improved method shows promise in enhancing the correlation between retrievability scores, it is important to acknowledge that real-world query logs remain the most reliable option for estimating retrievability. Real-world query logs capture the actual queries issued by users in authentic information-seeking contexts, providing a more accurate representation of retrieval performance.

In conclusion, this study highlights the limited correlation between retrievability scores obtained from real-world queries and artificially generated queries using Azzopardi's method. This finding underscores the need for caution in relying solely on artificial query sets to estimate retrievability accurately. While the proposed improved method shows an enhanced correlation, real-world query logs remain the gold standard for reliable retrievability estimation. Future research should continue to explore and refine techniques for generating artificial queries, aiming to bridge the gap between artificial and real-world queries to improve robustness of the retrievability analysis and retrievability bias estimation.

Chapter 6

Correlation between Retrievability and PageRank

This chapter delves into the investigation of the correlation between PageRank and Retrievability, to achieve the final goal of this thesis [1.2]. The objective of this undertaking is to explore the relationship between these two measures, necessitating the selection of an appropriate corpus with an inter-page link structure that is also topically diverse enough to facilitate variations in retrievability.

To fulfill this requirement, two distinct document collections have been chosen. Firstly, Wikipedia, an extensive repository of knowledge covering a wide range of topics, has been selected. Notably, Wikipedia articles contain references to other Wikipedia articles, rendering it suitable for PageRank analysis. Secondly, the TREC Web Corpus from the year 2000, specifically WT10g, has been utilized. This corpus represents a snapshot of a subset of the World Wide Web (WWW) from the year 2000, making it an ideal candidate for conducting PageRank analysis on webpage link graphs.

The methodology employed in obtaining the necessary in-links and out-links information from these two corpora will be elucidated in this chapter. Additionally, the computational challenges associated with performing a PageRank computation on such large datasets will be discussed. Furthermore, the chapter will present the PageRank algorithm employed, including the steps taken to address the aforementioned challenges.

Finally, this chapter will conclude with a brief discussion on the observations derived from the correlation of PageRank scores and the retrievability scores. These findings will shed light on the relationship between PageRank and Retrievability, contributing to a deeper understanding of the interplay between link analysis algorithms and the retrievability of information in large-scale document collections.

6.1 Motivation

PageRank [Brin and Page, 1998], renowned for enhancing the quality of search results in web search, has played a pivotal role in the success of Google as the "default" search engine of choice for the masses since the early 2000s. This algorithm, which assigns a measure of importance or usefulness to webpages, has significantly influenced the way information is retrieved from the World Wide Web (WWW).

Given the effectiveness of PageRank in improving search outcomes, it is worth exploring whether Retrievability scores can serve as an alternative measure of usefulness for documents within a corpus, much like PageRank does. By investigating the correlation between PageRank document ranks and Retrievability document ranks, this thesis aims to shed light on the potential interchangeability of PageRank and Retrievability.

If a significant correlation is discovered between these two measures, it would imply that Retrievability could serve as a viable replacement for PageRank. On the other hand, a moderate correlation could suggest that the ranks obtained from both PageRank and Retrievability may complement each other, potentially enhancing search effectiveness when used together within the retrieval model. Understanding the relationship between PageRank and Retrievability has the potential to refine the retrieval models employed in search engines, leading to improved search outcomes and a more efficient retrieval process. By evaluating the correlation between these measures, valuable insights can be gained regarding their utility as measures of document usefulness and their implications for enhancing search effectiveness.

Thus, the motivation behind this study lies in the desire to explore the potential of Retrievability as a measure of document usefulness and to assess its correlation with PageRank. The findings of this investigation contribute to the ongoing efforts aimed at optimizing search effectiveness in large-scale document collections.

6.2 Methodology

In order to investigate the correlation between retrievability and PageRank in Wikipedia, we employed a methodology that involved extracting relevant data from publicly available monthly data dumps provided by Wikipedia. The data dumps provide comprehensive information about the structure and relationships among articles within the Wikipedia ecosystem. Specifically, we utilized three key tables from the data dumps: the Page table, the Pagelinks table, and the Redirect table. Each of these tables plays a crucial role in understanding the interlinking and redirection patterns within Wikipedia.

Page table The Page table serves as the central repository of information regarding individual pages within the MediaWiki installation. It contains entries for every page, identifying them by their titles and providing essential metadata. Notably, the actual content of each page is stored separately in the text table. To retrieve the text of a specific article, the MediaWiki system initially searches for the page's title in the Page table. Upon locating the entry, it then utilizes the associated page_latest value to search the revision table for the corresponding revision_id. Subsequently, the revision_text_id is obtained, which is used to search the text table and retrieve the actual text content. It is important to note that when a page is deleted, its revisions are transferred to the archive table for archival purposes.

Pagelinks Table The Pagelinks table plays a critical role in tracking all internal links within the wiki. For every internal link present in a source page, an entry is recorded in the Pagelinks table. Each entry includes the source page's ID and namespace, along with the article name and namespace of the target page being linked to. While multiple instances of the source page's ID may exist, corresponding to the number of internal links within it, the table ensures a unique entry for each internal link associated with any given page ID. It should be noted that the target page referenced in the Pagelinks table may or may not exist, and due to changes such as renames or deletions, it may refer to different page records over time. This table was introduced in version 1.5 and combines the information previously stored in the links table and the brokenlinks table from version 1.4.

Redirect Table The Redirect table contains information about pages that currently serve as redirects. It includes the ID of the source page and details about the target page. It is worth mentioning that the database dumps provided by Wikipedia and other Wikimedia projects, available at https://dumps.wikimedia.org/, may have incomplete data in this table. Specifically, only redirect pages created or edited after the summer of 2007 are present. For redirects created before that time, we relied on the Pagelinks table to identify the appropriate target page.

6.2.1 Challenges and Resolutions

The computation of PageRank using the standard algorithm on the Wikipedia dataset poses significant challenges due to the size of the dataset. The standard algorithm loads the entire graph structure into primary memory, making it infeasible to perform the computation on a normal personal laptop or desktop with RAM capacity up to 16GB. The large graph leads to memory overflow, hindering the efficient execution of the PageRank algorithm. To address this challenge, this thesis proposes a solution that enables PageRank computation without the need to load the entire graph into RAM.

The proposed solution involves the utilization of left and right column sorted link files stored in secondary memory. These link files contain information about which article (pageID) points to which other articles, with each entry recorded in a single line. By employing this approach, the algorithm can iterate through the sorted file one line at a time, updating the PageRank scores of the documents in an iterative fashion.

This solution effectively overcomes the memory limitations of a typical personal laptop or desktop by eliminating the need to load the entire graph into primary memory. Instead, the algorithm accesses the necessary information from the sorted link files stored in secondary memory. By processing the data in a sequential manner, the PageRank computation can be performed with minimal memory requirements, enabling the utilization of hardware resources with limited RAM capacity.

Furthermore, this approach offers scalability, as it allows the PageRank algorithm to handle increasingly large datasets. By employing sorted link files, the algorithm can efficiently process the dataset regardless of its size, as long as the available secondary memory is sufficient to accommodate the sorted files.

It is worth noting that the utilization of sorted link files introduces additional I/O operations compared to the standard algorithm. However, the overall performance impact is mitigated by the avoidance of memory overflow and the ability to perform the computation on hardware with limited RAM capacity. The trade-off between increased I/O operations and improved memory utilization is deemed acceptable in this context.

6.3 Algorithm

The given algorithm-1 implements the iterative PageRank algorithm for computing the PageRank scores of nodes in a graph. The algorithm takes a left-sorted file and a right-sorted file as input, which represent the edges of the graph. The algorithm consists of two main functions: `makeDictWithOutlinksCounts()` and `pagerank()`.

The `makeDictWithOutlinksCounts()` function reads the left-sorted file and creates a dictionary called `nodes_dict`, which stores information about each node in the graph. The dictionary contains the number of outlinks for each node, the starting PageRank value, the current PageRank value, and a flag indicating whether the node has incoming links. The function iterates over the lines of the file, extracts the source and target nodes from each line, and updates the `nodes_dict` accordingly.

The `pagerank()` function performs the actual PageRank computation. It receives the `nodes_dict`, which contains information about each node in the graph, the right-sorted file that represents the edges of the graph, the damping factor (d), the starting PageRank value (`start_value`), the precision for convergence (`precision`), and the maximum number of iterations (`max_iterations`). This function performs the iterative calculation of the PageRank scores for each node. The function begins by initializing a tolerance value based on the specified precision, which determines the level of convergence required for the algorithm to terminate. It then enters a loop that executes for a maximum of `max_iterations` times or until convergence is achieved. Within each iteration, the function traverses the lines of the right-sorted file, extracting the incoming link and the corresponding target node. This link represents an incoming connection to the target node from another node in the graph. The function calculates the PageRank score for the target node by summing the previous PageRank values of the incoming links and dividing it by the number of outlinks of the source node.

To facilitate the alternating update of the PageRank values, the function employs two indices: `prev_pagerank_index` and `curr_pagerank_index`. These indices determine the position of the previous and current PageRank values within the `nodes_dict` dictionary, allowing for the appropriate updating of the values. During the iteration process, the function also checks for convergence by comparing the difference between the current and previous PageRank values of each node with the tolerance value. If the difference exceeds the tolerance, indicating that the PageRank scores have not converged sufficiently, the flag variable is set to False. The function further handles special cases related to the first iteration. It marks nodes with incoming links as "touched" by setting the corresponding flag in the `nodes_dict` to True. Additionally, it initializes nodes that do not have any outgoing links by adding them to the `nodes_dict` with appropriate default values. Upon completing the iteration over the right-sorted file, the function sets the PageRank score of "untouched" nodes (nodes without incoming links) to a value of $(1 - d)$. This step ensures that nodes without incoming links receive a fraction of the damping factor as their PageRank score. At the end of each iteration, the function updates the `nodes_dict` with the newly calculated PageRank values for each target node. It then evaluates the convergence status by checking the flag variable. If convergence has been achieved for all nodes (i.e., if the flag is True), the iteration process is terminated. Finally, the function returns the updated `nodes_dict` and the number of iterations performed.

In summary, the algorithm reads the input files, initializes the necessary data structures, performs iterative PageRank computations, and outputs the final PageRank scores for each node. It employs a dictionary to store and update the relevant information for each node, and it utilizes the left-sorted and right-sorted files to calculate the PageRank scores based on incoming and outgoing links.

Algorithm 1: PageRank Computation (Part 1)

```
Function
 PageRank($leftSortedLinkFile, rightSortedLinkFile, d, startValue, precision, maxIterations$):
    Input  : $leftSortedLinkFile$, $rightSortedLinkFile$, $d$, $startValue$,
             $precision$, $maxIterations$
    Output: $nodesDict$, $i$
    $nodesDict \leftarrow$
     makeDictWithOutlinksCounts($leftSortedLinkFile$, $startValue$)
    $tol \leftarrow 10^{-precision}$
    for $i \leftarrow 1$ to $maxIterations$ do
        $prevPagerankIndex \leftarrow (i-1) \mod 2 + 1$
        $currPagerankIndex \leftarrow i \mod 2 + 1$
        $flag \leftarrow$ True
        $prevToNode \leftarrow$ None
        forall $line$ in $rightSortedLinkFile$ do
            $inLink, currToNode \leftarrow$ parseLine($line$)
            if $prevToNode \neq currToNode$ then
                if $prevToNode \neq None$ then
                    $pagerank \leftarrow (1-d) + d * pagerank$
                    $nodesDict[prevToNode][currPagerankIndex] \leftarrow$
                     $pagerank$
                    if $|nodesDict[prevToNode][currPagerankIndex] -$
                     $nodesDict[prevToNode][prevPagerankIndex]| \geq tol$
                     then
                        $flag \leftarrow$ False
                    end
                    if $i == 1$ then
                        $nodesDict[prevToNode][3] \leftarrow$ True
                    end
                end
                $pagerank \leftarrow 0$
                if $currToNode \notin nodesDict$ then
                    $nodesDict[currToNode] \leftarrow$
                     $[0, startValue, startValue, False]$
                end
            end
            $prevPagerank \leftarrow nodesDict[inLink][prevPagerankIndex]$
            $numOutlinks \leftarrow nodesDict[inLink][0]$
            $pagerank \leftarrow pagerank + prevPagerank/numOutlinks$
            $prevToNode \leftarrow currToNode$
        end
        if $prevToNode \neq None$ then
            $pagerank \leftarrow (1-d) + d * pagerank$
            $nodesDict[prevToNode][currPagerankIndex] \leftarrow pagerank$
            if
             $nodesDict[prevToNode][currPagerankIndex] - nodesDict[prevToNode][prevPagerankIndex] >$
             $tol$ then
                $flag \leftarrow$ False
            end
            if $i == 1$ then
                $nodesDict[prevToNode][3] \leftarrow$ True
            end
        end
        if $i == 1$ then
            forall $node$ in $nodesDict$ do
                if $nodesDict[node][3] == False$ then
                    $nodesDict[node][prevPagerankIndex] \leftarrow (1-d)$
                    $nodesDict[node][currPagerankIndex] \leftarrow (1-d)$
                end
            end
        end
        if $flag == True$ then
            break
        end
    end
    return $nodesDict, i$
```

```
Algorithm 2: PageRank Computation (Part 2)
Function makeDictWithOutlinksCounts($leftSortedLinkFile, startValue$):
    Input  : $leftSortedLinkFile$, $startValue$
    Output: $nodesDict$
    $nodesDict \leftarrow \emptyset$
    $prevFromNode \leftarrow$ None
    $outlinksCount \leftarrow 1$
    forall $line$ in $leftSortedLinkFile$ do
        $fromNode, toNode \leftarrow$ parseLine($line$)
        if $prevFromNode == None$ then
            $prevFromNode \leftarrow fromNode$
            continue
        if $fromNode == prevFromNode$ then
            $outlinksCount \leftarrow outlinksCount + 1$
        else
            $nodesDict[prevFromNode] \leftarrow$
              $[outlinksCount, startValue, startValue, False]$
            $outlinksCount \leftarrow 1$
        $prevFromNode \leftarrow fromNode$
    if $prevFromNode \neq None$ then
        $nodesDict[prevFromNode] \leftarrow$
          $[outlinksCount, startValue, startValue, False]$
    return $nodesDict$
Function parseLine($line$):
    Input  : $line$
    Output: $fromNode, toNode$
    $fromNode, toNode \leftarrow line.split(''\,\widehat{}\,)[: 2]$
    return $fromNode, toNode$
```

6.4 Results

Collection	Kendall's rank correlation coefficient τ		Rank Biased Overlap measure RBO	
	Q_{real}	Q_{art}	Q_{real}	Q_{art}
Wikipedia	0.0896	0.1748	**0.5343**	**0.5753**
WT10g	0.0361	0.0391	**0.5126**	**0.5095**

Table 6.1: Correlation Coefficients between PageRank of documents vs $r(d)$ from AOL queries (Q_{real}) and the proposed improved query generation method created artificial queries (Q_{art}) for c = 100.

6.5 Discussion

The analysis conducted in this chapter aimed to investigate the correlation between the ranks of documents based on PageRank scores and Retrievability scores within the Wikipedia and WT10g corpora. The findings reveal a weak correlation between these two measures for both datasets.

The weak correlation observed suggests that the ranks obtained from PageRank and Retrievability do not align closely with each other. This implies that the two measures capture different aspects of document importance or usefulness within the corpus. While PageRank emphasizes the popularity and link structure of webpages, Retrievability focuses on the accessibility and ease of retrieving information from documents. The weak correlation between PageRank and Retrievability ranks indicates that they are not directly interchangeable measures. Therefore, replacing PageRank with Retrievability alone may not yield comparable results in terms of search effectiveness. However, the modest correlation suggests that there may be situations where the combination of PageRank and Retrievability ranks could potentially enhance the overall search performance.

It is important to note that the weak correlation observed does not diminish the value of either measure. PageRank continues to be an effective algorithm for assessing the importance of webpages and improving search results. Similarly, Retrievability scores provide insights into the accessibility and retrieval potential of documents. The weak correlation merely suggests that these measures capture different dimensions of document quality and usefulness. The observed weak correlation may be attributed to various factors. Firstly, the underlying metrics used in computing PageRank and Retrievability are distinct. PageRank relies on the link structure and connectivity of webpages, while Retrievability is influenced by factors such as document length, term frequency, and term distribution. These contrasting factors contribute to the disparity in ranks obtained from the two measures.

Despite the weak correlation, the findings of this study provide valuable insights into the interplay between PageRank and Retrievability. The contrasting perspectives offered by these measures underscore the importance of considering multiple dimensions of document importance and usefulness in search systems. Future research can delve deeper into the specific factors influencing the weak correlation and explore ways to leverage the combination of PageRank and Retrievability to improve search effectiveness. In conclusion, the analysis conducted in this chapter reveals a weak correlation between the ranks of documents based on PageRank and Retrievability scores in both the Wikipedia and WT10g corpora. This suggests that these measures capture different aspects of document importance and usefulness. While they may not be directly interchangeable, their combination holds potential for enhancing search effectiveness.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In conclusion, this thesis has addressed several critical issues in the field of Information Retrieval (IR) systems and provided valuable insights into the limitations and potential improvements of existing methodologies. Through extensive research and analysis, a number of significant findings have been uncovered, shedding light on the challenges and opportunities in this domain.

Firstly, a major flaw in the artificial query generation method for Retrievability analysis has been identified. This flaw highlighted the need for a more robust and reliable approach to accurately assess the retrievability of documents in IR systems. In response to this, an improved method was proposed, which demonstrated a notable enhancement in correlation, thus offering a more accurate measure of document retrievability.

Moreover, an important discovery was made regarding a new kind of bias in the Relevance Judgement process, which significantly affected the evaluation of IR systems. This bias was found to favor highly retrievable documents, thereby potentially skewing the results of relevance-based evaluations. Recognizing this bias is crucial for ensuring fair and unbiased assessments of IR systems, and future research should focus on developing strategies to mitigate its impact.

The thesis also explored the impact of employing the RM3 technique on IR system performance. While RM3 demonstrated the ability to boost overall performance, it was revealed that this enhancement came at the cost of making unique relevant documents less findable. This finding emphasizes the importance of carefully considering the trade-offs associated with different retrieval techniques and highlights the need for further investigation into optimizing retrieval effectiveness without sacrificing the retrieval of unique and relevant documents.

Additionally, a slight correlation was observed between document ranks from PageRank and Retrievability measures. This finding suggests a potential relationship between the two metrics and provides a basis for exploring their interplay and mutual reinforcement in future research. Understanding the connections between different evaluation methods can contribute to the development of more comprehensive and effective IR systems.

Overall, this thesis has made contributions to the field of information retrieval by uncovering important limitations and proposing improvements to existing methodologies. The findings presented here underscore the complexity of IR systems and emphasize the need for ongoing research and innovation to address the challenges and enhance the effectiveness of these systems. By addressing the identified flaws and biases, and by leveraging the potential correlations between different metrics, future advancements in IR can lead to more reliable and efficient systems capable of meeting the evolving needs of users in the ever-expanding world of information retrieval.

7.2 Future Work

While this thesis has made recognisable progress in identifying and addressing limitations in the current methods used for retrievability analysis and relevance judgment-based evaluations of IR systems, there are several avenues for further research and exploration. The following suggestions outline potential future work that can build upon the findings and contribute to the improvement of search result quality:

Investigating the Impact of the Proposed Improved Method

The thesis proposed an improved method for artificial query generation, which demonstrated an enhancement in correlation. However, further investigation is needed to thoroughly evaluate the effectiveness of this method across different IR systems, datasets, and retrieval tasks. This analysis can provide insights into the generalizability and robustness of the proposed approach.

Addressing Bias Towards Highly Retrievable Documents

The identification of bias towards highly retrievable documents in relevance judgment has shed light on a previously overlooked aspect of IR evaluation. Future work can focus on developing strategies to mitigate this bias and improve the fairness and accuracy of relevance judgment-based evaluations. This may involve the design of alternative relevance judgment mechanisms or the incorporation of bias-aware models into the evaluation process.

Evaluating the Impact of RM3 on Finding Unique Relevant Documents

The observation that the use of RM3 can boost performance but potentially hinder the findability of unique relevant documents warrants further investigation. Future studies can delve deeper into understanding the underlying reasons for this trade-off and explore modifications or alternative approaches to RM3 that can mitigate its negative impact on finding unique relevant documents.

Exploring Retrievability for Search Result Quality Enhancement

Inspired by the success of PageRank in improving search result quality, future work can focus on exploring how retrievability measures can be effectively incorporated into ranking algorithms. Some correlation between document ranks from PageRank and Retrievability has made us think that perhaps Retrievability may be utilized as document information enrichness measure. This can involve leveraging the retrievability as a usefulness measure to develop modified retrieval models that enhance search result quality.

Bibliography

[Abdul-Jaleel, Allan, Croft, Diaz, Larkey, Li, Smucker, and Wade, 2004]
Abdul-Jaleel, N., Allan, J., Croft, W. B., Diaz, F., Larkey, L., Li, X., Smucker, M. D., & Wade, C. (2004). Umass at trec 2004: Novelty and hard. *Computer Science Department Faculty Publication Series*, 189.

[Azzopardi and Bache, 2010]
Azzopardi, L., & Bache, R. (2010). On the relationship between effectiveness and accessibility. *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 889–890.

[Azzopardi, English, Wilkie, and Maxwell, 2014]
Azzopardi, L., English, R., Wilkie, C., & Maxwell, D. (2014). Page retrievability calculator. *Advances in Information Retrieval: 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings 36*, 737–741.

[Azzopardi and Vinay, 2008a]
Azzopardi, L., & Vinay, V. (2008a). Document accessibility: Evaluating the access afforded to a document by the retrieval system. *Workshop on Novel Methodologies for Evaluation in Information Retrieval*, 52–60.

[Azzopardi and Vinay, 2008b]
Azzopardi, L., & Vinay, V. (2008b). Retrievability: An evaluation measure for higher order information access tasks. *Proceedings of the 17th ACM conference on Information and knowledge management*, 561–570.

[Azzopardi, Wilkie, Russell-Rose, Azzopardi, and Wilkie, 2013]
Azzopardi, L., Wilkie, C., Russell-Rose, T., Azzopardi, L., & Wilkie, C.

(2013). Towards measures and models of findability. *Clarke et al.[19]*, 3–4.

[Baeza-Yates, Ribeiro-Neto, et al., 1999]
Baeza-Yates, R., Ribeiro-Neto, B., et al. (1999). *Modern information retrieval* (Vol. 463).

[Bailey, Craswell, and Hawking, 2003]
Bailey, P., Craswell, N., & Hawking, D. (2003). Engineering a multi-purpose test collection for web retrieval experiments. *Information Processing & Management*, *39*(6), 853–871.

[Bashir, 2014]
Bashir, S. (2014). Estimating retrievability ranks of documents using document features. *Neurocomputing*, *123*, 216–232.

[Bashir and Khattak, 2014]
Bashir, S., & Khattak, A. S. (2014). Producing efficient retrievability ranks of documents using normalized retrievability scoring function. *Journal of Intelligent Information Systems*, *42*, 457–484.

[Bashir and Rauber, 2009a]
Bashir, S., & Rauber, A. (2009a). Analyzing document retrievability in patent retrieval settings. *Database and Expert Systems Applications: 20th International Conference, DEXA 2009, Linz, Austria, August 31–September 4, 2009. Proceedings 20*, 753–760.

[Bashir and Rauber, 2009b]
Bashir, S., & Rauber, A. (2009b). Identification of low/high retrievable patents using content-based features. *Proceedings of the 2nd international workshop on Patent information retrieval*, 9–16.

[Bashir and Rauber, 2009c]
Bashir, S., & Rauber, A. (2009c). Improving retrievability of patents with cluster-based pseudo-relevance feedback documents selection. *Proceedings of the 18th ACM conference on Information and knowledge management*, 1863–1866.

[Bashir and Rauber, 2010a]

Bashir, S., & Rauber, A. (2010a). Improving retrievability and recall by automatic corpus partitioning. *Transactions on large-scale data-and knowledge-centered systems II*, 122–140.

[Bashir and Rauber, 2010b]

Bashir, S., & Rauber, A. (2010b). Improving retrievability of patents in prior-art search. *Advances in Information Retrieval: 32nd European Conference on IR Research, ECIR 2010, Milton Keynes, UK, March 28-31, 2010. Proceedings 32*, 457–470.

[Bashir and Rauber, 2011]

Bashir, S., & Rauber, A. (2011). On the relationship between query characteristics and ir functions retrieval bias. *Journal of the American Society for Information Science and Technology*, *62*(8), 1515–1532.

[Bashir and Rauber, 2014]

Bashir, S., & Rauber, A. (2014). Automatic ranking of retrieval models using retrievability measure. *Knowledge and information systems*, *41*, 189–221.

[Bashir and Rauber, 2017]

Bashir, S., & Rauber, A. (2017). Retrieval models versus retrievability. *Current Challenges in Patent Information Retrieval*, 185–212.

[Belew, 2000]

Belew, R. K. (2000). *Finding out about: A cognitive perspective on search engine technology and the www*. Cambridge University Press.

[Belkin, 1980]

Belkin, N. J. (1980). Anomalous states of knowledge as a basis for information retrieval. *Canadian journal of information science*, *5*(1), 133–143.

[Bennett, Scholer, and Uitdenbogerd, 2008]

Bennett, G., Scholer, F., & Uitdenbogerd, A. (2008). A comparative study of probabilistic and language models for information retrieval. *Database Technologies 2008: Proceedings of the Nineteenth Australasian Database Conference (ADC 2008)*, 65–74.

[Brin and Page, 1998]

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, *30*(1-7), 107–117.

[Buckley, Dimmick, Soboroff, and Voorhees, 2006]

Buckley, C., Dimmick, D., Soboroff, I., & Voorhees, E. (2006). Bias and the limits of pooling. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 619–620.

[Buckley, Dimmick, Soboroff, and Voorhees, 2007]

Buckley, C., Dimmick, D., Soboroff, I., & Voorhees, E. (2007). Bias and the limits of pooling for large collections. *Information retrieval*, *10*, 491–508.

[Buckley and Voorhees, 2004]

Buckley, C., & Voorhees, E. M. (2004). Retrieval evaluation with incomplete information. *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 25–32.

[Callan and Connell, 2001]

Callan, J., & Connell, M. (2001). Query-based sampling of text databases. *ACM Transactions on Information Systems (TOIS)*, *19*(2), 97–130.

[Chen, Azzopardi, and Scholer, 2017]

Chen, R.-C., Azzopardi, L., & Scholer, F. (2017). An empirical analysis of pruning techniques: Performance, retrievability and bias. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2023–2026.

[Cleverdon, 1991]

Cleverdon, C. W. (1991). The significance of the cranfield tests on index languages. *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, 3–12.

[Croft, Metzler, and Strohman, 2010]

Croft, W. B., Metzler, D., & Strohman, T. (2010). *Search engines: Information retrieval in practice* (Vol. 520). Addison-Wesley Reading.

[Dewey, 1876]

Dewey, M. (1876). *A classification and subject index, for cataloguing and* *arranging the books and pamphlets of a library*. Brick row book shop, Incorporated.

[Flexer, Schnitzer, Gasser, and Pohle, 2010]
Flexer, A., Schnitzer, D., Gasser, M., & Pohle, T. (2010). Combining features reduces hubness in audio similarity. *Children*, *15*(15.95), 4–07.

[Furnas, Landauer, Gomez, and Dumais, 1987]
Furnas, G. W., Landauer, T. K., Gomez, L. M., & Dumais, S. T. (1987). The vocabulary problem in human-system communication. *Communications of the ACM*, *30*(11), 964–971.

[Ganguly, Bandyopadhyay, Mitra, and Jones, 2016]
Ganguly, D., Bandyopadhyay, A., Mitra, M., & Jones, G. J. (2016). Retrievability of code mixed microblogs. *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 973–976.

[Gasser, Flexer, and Schnitzer, 2010]
Gasser, M., Flexer, A., & Schnitzer, D. (2010). Hubs and orphans-an explorative approach. *Proceedings of the 7th Sound and Music Computing Conference (SMC'10)*.

[Gastwirth, 1971]
Gastwirth, J. L. (1971). A general definition of the lorenz curve. *Econometrica: Journal of the Econometric Society*, 1037–1039.

[Hajian, Bonchi, and Castillo, 2016]
Hajian, S., Bonchi, F., & Castillo, C. (2016). Algorithmic bias: From discrimination discovery to fairness-aware data mining. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2125–2126.

[D. Harman, 1993]
Harman, D. (1993). Overview of trec-1. *Human Language Technology: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.

[D. K. Harman, 1993]

Harman, D. K. (1993). *The first text retrieval conference (trec-1)* (Vol. 500). US Department of Commerce, National Institute of Standards; Technology.

[Jones, Walker, and Robertson, 2000]

Jones, K. S., Walker, S., & Robertson, S. E. (2000). A probabilistic model of information retrieval: Development and comparative experiments: Part 2. *Information processing & management*, *36*(6), 809–840.

[Jordan, Watters, and Gao, 2006]

Jordan, C., Watters, C., & Gao, Q. (2006). Using controlled query generation to evaluate blind relevance feedback algorithms. *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, 286–295.

[Justeson and Katz, 1991]

Justeson, J. S., & Katz, S. M. (1991). Co-occurrences of antonymous adjectives and their contexts. *Computational linguistics*, *17*(1), 1–20.

[Kleinberg, 1999]

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, *46*(5), 604–632.

[Lavrenko and Croft, 2017]

Lavrenko, V., & Croft, W. B. (2017). Relevance-based language models. *ACM SIGIR Forum*, *51*(2), 260–267.

[Lipani, Lupu, Aizawa, and Hanbury, 2015]

Lipani, A., Lupu, M., Aizawa, A., & Hanbury, A. (2015). An initial analytical exploration of retrievability. *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, 329–332.

[Losada and Azzopardi, 2008]

Losada, D. E., & Azzopardi, L. (2008). An analysis on document length retrieval trends in language modeling smoothing. *Information Retrieval*, *11*, 109–138.

[Manning, Schütze, and Raghavan, 2008]

Manning, C. D., Schütze, H., & Raghavan, P. (2008). *Introduction to information retrieval* (Vol. 39). Cambridge University Press Cambridge.

[Mowshowitz and Kawaguchi, 2005]

Mowshowitz, A., & Kawaguchi, A. (2005). Measuring search engine bias. *Information processing & management*, *41*(5), 1193–1205.

[Noor and Bashir, 2015]

Noor, S., & Bashir, S. (2015). Evaluating bias in retrieval systems for recall oriented documents retrieval. *International Arab Journal of Information Technology (IAJIT)*, *12*(1).

[Palma, 2011]

Palma, J. G. (2011). Homogeneous middles vs. heterogeneous tails, and the end of the 'inverted-u': It's all about the share of the rich. *development and Change*, *42*(1), 87–153.

[Pickens, Cooper, and Golovchinsky, 2010]

Pickens, J., Cooper, M., & Golovchinsky, G. (2010). Reverted indexing for feedback and expansion. *Proceedings of the 19th ACM international conference on Information and knowledge management*, 1049–1058.

[Ponte and Croft, 2017]

Ponte, J. M., & Croft, W. B. (2017). A language modeling approach to information retrieval. *ACM SIGIR Forum*, *51*(2), 202–208.

[Stephen E Robertson, 1977]

Robertson, S. E. [Stephen E]. (1977). The probability ranking principle in ir. *Journal of documentation.*

[Stephen E. Robertson, Walker, Hancock-Beaulieu, Gull, and Lau, 1992]

Robertson, S. E. [Stephen E.], Walker, S., Hancock-Beaulieu, M., Gull, A., & Lau, M. (1992). Okapi at trec. *TREC*, 21–30.

[Rocchio Jr, 1971]

Rocchio Jr, J. J. (1971). Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing.*

[Roelleke, 2013]

Roelleke, T. (2013). Information retrieval models. *Foundations and Relationships. Morgan & Claypool Publishers, USA.*

[Roy, 2022]

Roy, D. (2022). *Introduction to information retrieval* [Course slide]. IISER Kolkata.

[Sabetghadam, Lupu, Bierig, and Rauber, 2015]

Sabetghadam, S., Lupu, M., Bierig, R., & Rauber, A. (2015). Reachability analysis of graph modelled collections. *European Conference on Information Retrieval*, *9022*, 370–381.

[Salton and Yang, 1973]

Salton, G., & Yang, C.-S. (1973). On the specification of term values in automatic indexing. *Journal of documentation*.

[Thaer Samar, Traub, van Ossenbruggen, Hardman, and de Vries, 2018]

Samar, T. [Thaer], Traub, M. C., van Ossenbruggen, J., Hardman, L., & de Vries, A. P. (2018). Quantifying retrieval bias in web archive search. *International Journal on Digital Libraries*, *19*, 57–75.

[TMH Samar, 2017]

Samar, T. [TMH]. (2017). *Access to and retrievability of content in web archives* (Doctoral dissertation). [Sl: sn].

[Sanderson, 2008]

Sanderson, M. (2008). Ambiguous queries: Test collections need more sense. *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 499–506.

[Sanderson and Joho, 2004]

Sanderson, M., & Joho, H. (2004). Forming test collections with no system pooling. *Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval*, 33–40.

[Singhal et al., 2001]

Singhal, A., et al. (2001). Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, *24*(4), 35–43.

[Singhal, Salton, Mitra, and Buckley, 1996]

Singhal, A., Salton, G., Mitra, M., & Buckley, C. (1996). Document length normalization. *Information Processing & Management*, *32*(5), 619–633.

[Sparck Jones, 1972]

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, *28*(1), 11–21.

[Taha, 2015]

Taha, A. A. (2015). Bias, effizienz und hubness: Herausforderungen in der anwendbarkeit von metriken. *Ausgezeichnete Informatikdissertationen 2015*.

[Traub, Samar, Van Ossenbruggen, He, de Vries, and Hardman, 2016]

Traub, M. C., Samar, T., Van Ossenbruggen, J., He, J., de Vries, A., & Hardman, L. (2016). Querylog-based assessment of retrievability bias in a large newspaper corpus. *2016 IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, 7–16.

[Turtle and Croft, 1992]

Turtle, H. R., & Croft, W. B. (1992). A comparison of text retrieval models. *The computer journal*, *35*(3), 279–290.

[Van Rijsbergen, 1979]

Van Rijsbergen, C. (1979). Information retrieval: Theory and practice. *Proceedings of the joint IBM/University of Newcastle upon tyne seminar on data base systems*, *79*.

[Vinay, Cox, Milic-Frayling, and Wood, 2006]

Vinay, V., Cox, I. J., Milic-Frayling, N., & Wood, K. (2006). Measuring the complexity of a collection of documents. *Advances in Information Retrieval: 28th European Conference on IR Research, ECIR 2006, London, UK, April 10-12, 2006. Proceedings 28*, 107–118.

[Wikipedia, 2023a]

Wikipedia. (2023a). Lorenz curve — Wikipedia, the free encyclopedia [[Online; accessed 08-May-2023]].

[Wikipedia, 2023b]

Wikipedia. (2023b). Rocchio algorithm — Wikipedia, the free encyclopedia [[Online; accessed 08-May-2023]].

[Wikipedia, 2023c]

Wikipedia. (2023c). Vector space model — Wikipedia, the free encyclopedia [[Online; accessed 07-May-2023]].

[Wilkie and Azzopardi, 2013a]

Wilkie, C., & Azzopardi, L. (2013a). An initial investigation on the relationship between usage and findability. *Advances in Information Retrieval: 35th European Conference on IR Research, ECIR 2013, Moscow, Russia, March 24-27, 2013. Proceedings 35*, 808–811.

[Wilkie and Azzopardi, 2013b]

Wilkie, C., & Azzopardi, L. (2013b). Relating retrievability, performance and length. *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 937–940.

[Wilkie and Azzopardi, 2014]

Wilkie, C., & Azzopardi, L. (2014). A retrievability analysis: Exploring the relationship between retrieval bias and retrieval performance. *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 81–90.

[Wilkie and Azzopardi, 2015]

Wilkie, C., & Azzopardi, L. (2015). Retrievability and retrieval bias: A comparison of inequality measures. *Advances in Information Retrieval: 37th European Conference on IR Research, ECIR 2015, Vienna, Austria, March 29-April 2, 2015. Proceedings 37*, 209–214.

[Wilkie and Azzopardi, 2017]

Wilkie, C., & Azzopardi, L. (2017). An initial investigation of query expansion bias. *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, 285–288.

[Witten, Witten, Moffat, Bell, Bell, Fox, and Bell, 1999]

Witten, I. H., Witten, I. H., Moffat, A., Bell, T. C., Bell, T. C., Fox, E., & Bell, T. C. (1999). *Managing gigabytes: Compressing and indexing documents and images*. Morgan Kaufmann.

[Zehlike, Bonchi, Castillo, Hajian, Megahed, and Baeza-Yates, 2017]

Zehlike, M., Bonchi, F., Castillo, C., Hajian, S., Megahed, M., & Baeza-Yates,

R. (2017). Fa* ir: A fair top-k ranking algorithm. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1569–1578.

[Zhang, Zhu, and Greenwood, 2004]
Zhang, Y., Zhu, H., & Greenwood, S. (2004). Web site complexity metrics for measuring navigability. *Fourth International Conference onQuality Software, 2004. QSIC 2004. Proceedings.*, 172–179.

[Zobel, 1998]
Zobel, J. (1998). How reliable are the results of large-scale information retrieval experiments? *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 307–314.