Thomas Mauerhofer, BSc

Using IMRaD Structure Features in Information Retrieval Raking Functions

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Roman Kern

Institute of Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, September 2020

This document is set in Palatino, compiled with pdfLaTeX2e and `Biber`.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Abstract

Today the internet is growing fast as users generate an increasing amount of data. Therefore, finding relevant information is getting more and more time-consuming. This happens as the internet consists of a larger amount of data that is distributed over various information sources. Search engines filter data, and reduce the time required to find relevant information. We focus on scientific literature search where search engines help to find scientific articles. An advantage of scientific articles is that they share a common structure to increase their readability. This structure is known as IMRaD (Introduction, Method, Results and Discussion). We tackle the question whether it is possible to improve the search result quality while searching for scientific works by leveraging IMRaD structure information. We use several state-of-the-art ranking algorithms, and compare them against each other in our experiments. Our results show that the importance of IMRaD chapter features depends on the complexity of the query. Finally, we focus on structured text retrieval and the influence of single chapters on the search result. We set out to tackle the problem to improve the quality of the results produced by state-of-the-art ranking algorithms for scientific literature research.

Keywords: Information Retrieval; Structured Text Retrieval; IMRaD; Term Frequency; TF-IDF; Ranked Boolean Retrieval; Okapi BM25; Divergence from Randomness

Contents

List of Figures

List of Tables

1 Introduction

Today a world without internet is unimaginable to many people. It affects almost all areas of our daily life. In a minute, for example, 3.7 million search queries are made on Google, 18 million messages are sent on Whatsapp, and 4.3 million videos are viewed on YouTube (see Figure 1.1). Furthermore, at the same time many websites are created, blog entries are written, videos and pictures are uploaded and shared. Hence, users generate a lot of data. Due to this vast amount of generated data, almost everything can be found on the internet.

In order to find relevant information, the challenge becomes to filter the data, because a typical user is usually interested in a specific piece of information. Applications which use information retrieval to find relevant information for the user are known as search engines. To be more specific, a search engine helps to reduce the time required to find a piece of information, and minimize the number of information sources that need to be searched. That sounds very simple at first, but each user has different subjective expectations, and therefore search engines have to fulfill different requirements. For example, a user uses a web search engine to search for ”restaurant”. Normally, the top recommendations would be restaurants near the user, or the best restaurants in town. But on the other hand the user could not be interested in eating, but wants to know the origin of the word.

During a single request search engines leverage different information channels. First, information provided by the user. For many known search engines, these are keywords used as a query. Another possibility is to search with files like pictures, articles, or scientific papers. When a search engine uses only this type of information it is called an explicit search. The explicit search process itself works in a way that stored data is examined first. After-

Figure 1.1: **User activity of the Internet in 2018.** The Internet has become a daily companion, and is an indispensable part of life. This figure gives an overview of what happens within the most popular services in a single minute. Retrieved July 19, 2019 from `https://www.allaccess.com/merge/archive/28030/2018-update-what-happens-in-an-internet-minute`

wards the results are ranked according to their relevance. Finally, the user gets an sorted list with the best results on top of the list.

However, there is also other implicit information available, which was not explicitly provided by the user. With this additional information, the results can be better adapted to the needs of the user. For example Agichtein et al. [ABD06] use machine learning approaches to take user actions during the search process into account. Leveraging this type of information in addition to information provided by the user is denoted implicit search.

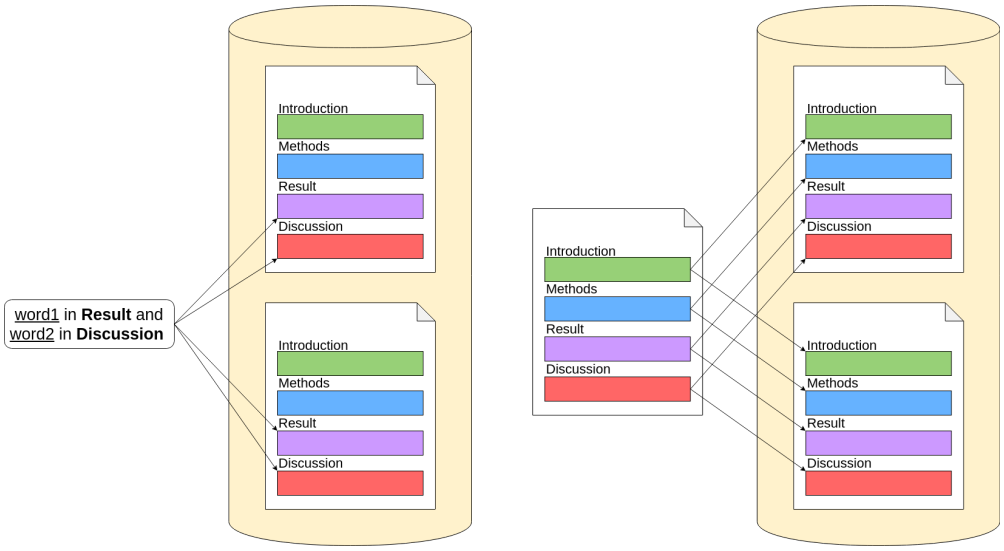
1.1 Motivation

Search engines are used in many different areas. One of them is in the field of science, where they simplify literature search. Before there were search engines on the Internet, literature search was only possible in libraries. The search was more complicated, since certain literature was only available in certain libraries. This resulted in a time expenditure, since literature had to be sent or picked up. In addition, all publications needed to be examined by the user itself. Overall the search of relevant literature was tedious for the user and a time consuming process.

Modern search engines made searching in the field of science more comfortable. With a single query all publications are searched for the query words. Normally, this takes only a few seconds. The ranking of the results also gives a first overview which literature is relevant for the user. To provide these publications to the user the search results typically contain a link to the publications.

The main goal of our thesis is to apply structural document information of scientific papers into the most common ranking algorithms of search engines. In general, a ranking algorithm ranks based on keywords in the text areas of scientific papers. Therefore, when two papers are related to each other they have similar keywords. We deduce that a more targeted search is possible by providing the information of the section where the keywords are located. For example, if the introduction is compared there is only a subset of keywords where the similarity depends on.

Sollaci and Pereira [SP04] describe in their work how the structure of a scientific paper changed over time. They concluded that the subdivision into introduction, method, result, and discussion became a common format in the course of the twentieth century. This subdivision is better known as the IMRaD structure, which we used as a base. This means we created a mapping from the chapter titles to this structure. Additionally, we adapted the ranking algorithms in a way that they take this structural information into account.



(a) Explicit Search (b) Implicit Search

Figure 1.2: **Explicit and Implicit Search using Document Structure Information.** Using explicit search (a) the user have to define where query words should occur. Then the specified areas of each scientific work in the database are searched through the word. The difference of implicit search (b) is that the user does not recognize the usage of structural information within the search engine. If a scientific work is used to search for other scientific works the words in the document gets structured by the search engine.

1.2 Research Questions

In general, we tackle the question whether it is possible to improve the search result quality while searching for scientific works by using IMRaD structure information. This means we evaluate if our adapted algorithms can achieve better results with the usage of IMRaD structure information than the standard version.

Hence, the following research questions arise:

1. Does the search result improve for explicit search using queries?

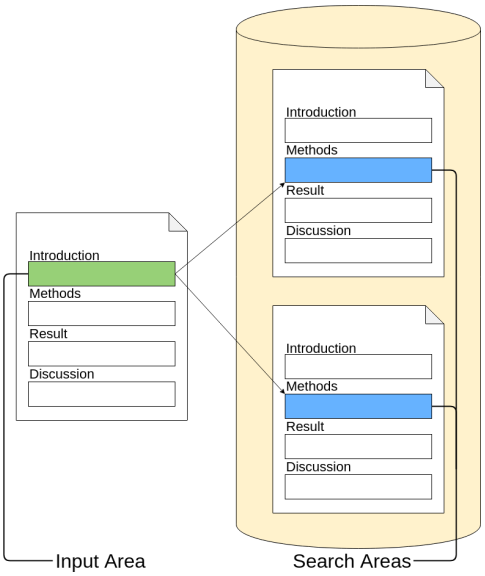


Figure 1.3: **Difference between Input Areas and Search Areas.** When using a scientific work to search implicit for other scientific works there are two different areas. The first is the input area which is part of the used scientific work. The second one is the search area where query words of the input area should occur. The used chapters of the input area do not have to be the same as the chapters of the search area.

For our first research question we focus on explicit search. Specifically, queries that contain the query keywords as well as the chapters in which they appear are used for the search. Our hypothesis is that used keywords vary in different search areas. Therefore, a more targeted search is possible if a user can specify where query words should occur (see Figure 1.2 a).

2. Does the search result improve for implicit search using complete scientific papers?

When a paper is used as a query, the query words and the structure information are available. Therefore, the query words can be mapped to their occurred chapter. (see Figure 1.2 b). Using the same assumption than in the previous question, the similarity of the scientific papers should be higher.

3. Does the search result improve if only a single chapter of the scientific

paper is used for searching?

The hypothesis is that each chapter has a different influence on the search result. Therefore, there are chapters with a higher impact than others, which could be positive for the resulting list, but also be negative. The idea is to improve the result by using only chapters with a positive effect.

Furthermore, keywords of a chapter can be used in other chapters (see Figure 1.3). For example, the Methods section of one paper would be referenced in the Related Work of another paper.

2 Related Work

2.1 Information Retrieval Models

Creating an information retrieval system is a complex process that has to be planned accordingly. To reach this goal models are used as a base, where the whole system is sketched. The model generation consists of two tasks. First, design a framework which represents the documents and the user queries. Second, create a ranking function, which generates a numeric rank for each document based on a query. Afterwards these ranks are used by the system to sort the documents.

One of the most common retrieval approaches is retrieval based on index terms. In this context an index term is a keyword, which appears in the document collection of the framework. This approach can be implemented efficiently as query words can be used as index terms with limited transformations. For example a user is interested in cooking, and searches for "Austrian dishes". The query words "Austrian", and "dishes" can directly used to search through the document collection since they do not need any transformation.

In general, information retrieval models consists of four parts. Ribeiro-Neto and Baeza-Yates [RB99] define them as a quadruple $[\mathbf{D}, \mathbf{Q}, \mathcal{F}, \mathcal{R}(q_i, d_j)]$, where:

1. **D** is a set composed of logical views of documents in a collection.
2. **Q** is a set composed of logical views of the user information needs. Such representations are called queries.

3. $\mathcal{F}$ is a framework for modeling document representations, queries, and their relationships.
4. $\mathcal{R}(q_i, d_j)$ is a raking function that associates a real number with a query representation $q_i \in \mathbf{Q}$ and a document representation $d_j \in \mathbf{D}$. The ranking function generates an order over all documents $\mathbf{D}$ with respect to a query q_i.

Hence, the model is used to define the framework $\mathcal{F}$ and the ranking function $\mathcal{R}(q_i, d_j)$. For example, for textual documents the document representation is a set of all terms within the document. To keep the collection smaller without losing any information stop words should be removed in a preprocessing step. The set of index terms within a document collection is called vocabulary. According to our document representation the query representation is a set of all terms within the query. There can also be an additional preprocessing step in the query creation. An example for such an preprocessing step would be synonyms which are added to the query set.

After the design of the framework, a ranking function is created. It should be constructed in a way that it fits to the requirements of the user. This means for a given query, the ranking function determines a numeric rank to each document in the collection, which represents the relevance for the user. For example, the ranking function counts how many query terms appear in the term set of a document.

Another example is to use term frequency as ranking function. Term frequency itself denotes how often a term occurs in a document. To be able to use it, the document representation is adapted from a set with all terms to a bag of words. In a bag of words each term is represented as a pair of term and term frequency. The ranking function sums the frequencies over all query terms. To be able to compare the documents using the term frequency, the ranks are normalized.

Information retrieval is used in several fields where the underlying models have to fulfill different requirements. Therefore, they are separated into text-based models, link-based models, and multimedia objects-based models. Furthermore, text-based models can be categorized in unstructured, and

semi-structured text models. Unstructured text models are used for text documents where the content is represented as sequence of words. Semi-structured text models contain structure such as title, sections, paragraphs, in addition to unstructured text.

The web is rapidly growing, and as a consequence has a huge number of web pages (i.e. documents). Therefore, additional information has to be leveraged as well. This means that the content of documents, and furthermore the links between those documents are take into account. Models which use those additional link information are called Link-based models where PageRank [BP98] and Hyperlink-Induced Topic Search [Kle99] are important parts of the models.

Information retrieval for multimedia objects differs according their underlying data from the first 2 types. For example, when thinking on a image it can be seen as a matrix of color values. Detecting similarities between images requires the calculation of more complex features, such as shapes. The representation of the query has to be adapted as well. The user can use words, or use images to define a query. One of the simplest forms of multimedia-based retrieval is image retrieval. Audio and video retrieval are more complicated since there is also a time value which have to be taken into account.

2.2 Unstructured Text Retrieval

In unstructured text retrieval, documents can be seen as sequence of words. The 3 classical models are boolean-, vector-, and probabilistic model. First, in the boolean model, documents and queries are represented as sets. Terms are stitched together with boolean operators to formulate user queries. Second, in the vector model, documents and queries are represented as a vector in a t-dimensional space. The size of t is defined by the number of words in the vocabulary of the collection. Third, in the probabilistic model, documents and queries are represented based on probability theory. Specifically by estimating the probability of a term appearing in a relevant

document. Gudivada et al. [Gud+97] advice in their work to denote boolean models as set theoretic, vector models as algebraic, and probabilistic models as probabilistic.

2.2.1 The Boolean Model

The boolean model is a well-known information retrieval model in the area of unstructured text retrieval. It was proposed as a paradigm for accessing large-scale systems since the 1950s [Melo9]. The model uses boolean operators and set theory to find relevant documents.

The classic boolean model can only decide if a document is relevant for the user, or not. It does not provide a rank, which is used to sort the documents. Salton et al. [SFW83] introduce in their work an extension where documents are sorted according their relevance.

Index terms are combined with the 3 boolean operators NOT($\neg$), AND($\wedge$), OR($\vee$) to formulate user queries. The disjunctive normal form of the query shows which areas of the sets are relevant. For example, for query $q = t_1 \wedge (t_2 \vee \neg t_3)$, and vocabulary V = $\{t_1, t_2, t_3\}$, q_{DNF} is:

$$q_{DNF} = (t_1 \wedge t_2 \wedge t_3) \vee (t_1 \wedge t_2 \wedge \neg t_3) \vee (t_1 \wedge \neg t_2 \wedge \neg t_3) \tag{2.1}$$

This representation of the query highlight that 3 areas are relevant for the user. First, all 3 query terms occur. Second, the first and the second term occur, but not the third. Finally, the first term occurs, but not the second and the third. Figure 2.1 displays the example query represented in a Venn diagram, where the 3 areas can be seen in a graphical representation.

The boolean model works also if not all terms of the vocabulary are part of the user query. Considering a vocabulary V = $\{t_1, t_2, t_3, t_4\}$, and the previous example query, the disjunctive normal form is:

$$\begin{aligned} q_{DNF} =& (t_1 \wedge t_2 \wedge t_3 \wedge \neg t_4) \vee (t_1 \wedge t_2 \wedge t_3 \wedge t_4) \vee \\ & (t_1 \wedge t_2 \wedge \neg t_3 \wedge \neg t_4) \vee (t_1 \wedge t_2 \wedge \neg t_3 \wedge t_4) \vee \\ & (t_1 \wedge \neg t_2 \wedge \neg t_3 \wedge \neg t_4) \vee (t_1 \wedge \neg t_2 \wedge \neg t_3 \wedge t_4) \end{aligned} \tag{2.2}$$

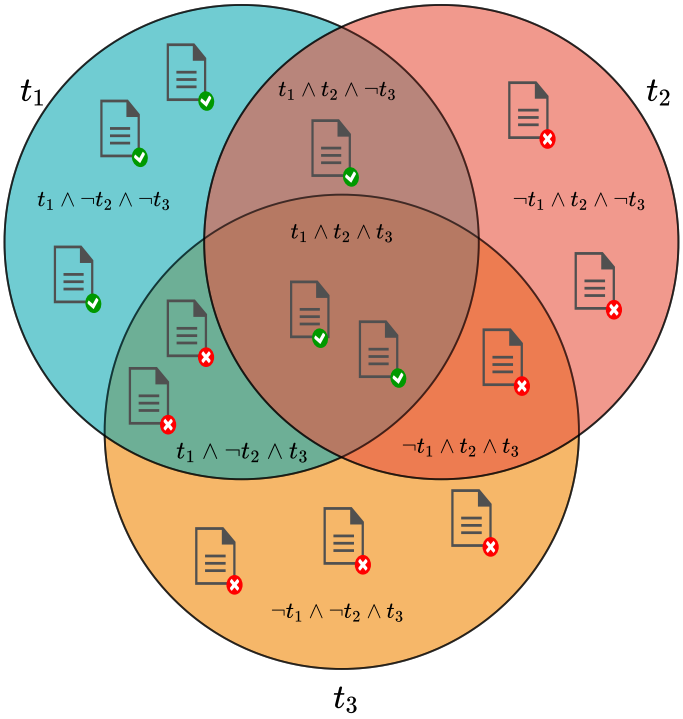


Figure 2.1: **Example query in the boolean model with 3 terms.** For the boolean model documents in the collection are represented as sets of terms. In this example the vocabulary of the document collection is given by V = $\{t_1, t_2, t_3\}$. Furthermore, documents can be separated according to the terms they are containing. Given a query $q = t_1 \wedge (t_2 \vee \neg t_3)$ all documents which satisfy this query are marked with an green hook. This means that they are relevant for the user. All other documents which does not satisfy the query are marked with a red cross.

The last term (i.e., t_4), which is not part of the query, is also considered in the disjunctive normal form. It is added once as present, and once as absent to the other 3 terms.

The main advantages of the boolean model are the clean formalism, and its simplicity [RB99]. These advantages comes with the usage of binary operators, and binary index term weighing. One of the main disadvantages is the exact matching of documents. This means that a document can only be relevant, or not relevant to the user, without any ranking. As a result, users receive too few or too many documents.

2.2.2 The Vector Space Model

The vector space model was introduced by Salton et al. [SWY75]. In the model documents and queries are represented as vectors in an t-dimensional space, whereas t is the number of words in the vocabulary.

The vector space model is more advanced than the Boolean model, as it contains partial matching. Partial matching means, that a degree of similarity between user queries and documents in the system are calculated. To accomplish this, non-binary weights are used in combination with index terms.

The idea of non-binary weights is based on the assumption that some index terms are more important than others to describe the content of a document. The calculation of such term weights is a challenging task, as they have to reflect the subjective expectation of a user. As a result, terms that appear in a few documents have a higher weight than terms that occur in many documents.

For example, a collection consists of 3 documents, given by $D_1 = \{$"cooking", "appetizer"$\}$, $D_2 = \{$"cooking", "main", "dish"$\}$, and $D_3 = \{$"cooking", "dessert"$\}$. A user is interested to prepare a dessert. Therefore he searches for "cooking dessert". The first query word "cooking" is part of each document. This means it is not very useful to define the users requirements. The second query word "dessert" is only part of one document. Therefore, it has more expressiveness than the first term. As a result, the weight of the first term will be smaller than the weight of the second term.

The combination of index terms and weights allows to calculate a numeric rank for each document. These ranks are used to sort the documents ranked by a user query. The resulting list contains the best results on top according to their relevance.

When having a closer look on index terms and their correlations it can be assumed that they are mutually independent. This means knowing $w_{i,j}$, where $w_{i,j}$ is the weight of an index term t_i in a document d_j, tells nothing

about the next weight $w_{i+1,j}$. The assumption does not hold, as terms in a document are always related to each other. For example, the words *mobile* and *phone*, which often occur together. Therefore, when a document contains the word *mobile* it is probable that the document also contains the word *phone*. This correlation is reflected in their term weights. The term-term correlation matrix described by Ribeiro-Neto and Baeza-Yates [RB99] model such relations. It is defined as:

$$\mathbf{C} = \mathbf{M} \cdot \mathbf{M}^T, \tag{2.3}$$

where $\mathbf{M}$ is a term-document matrix with t rows, and N columns. Each row contains a term of the vocabulary. As a result, the number of rows is equal to the size of the vocabulary. The columns represent the documents collection, where each column contains a single document. Each entry in the matrix $\mathbf{M}$ is given a weight $w_{i,j}$ associated with index term t_i and document d_j. In the term-term correlation matrix $\mathbf{C}$ each element $c_{u,v} \in \mathbf{C}$ describes the correlation between the terms t_u and t_v, which is given by:

$$c_{u,v} = \sum_{d_j} w_{u,j} \times w_{v,j}. \tag{2.4}$$

Therefore the relation between any two terms t_u and t_v is in the matrix. It is based on the joint co-occurrence of the two terms within all documents of the collection. For example, when having a collection of 2 documents, and the vocabulary of the collection is given by V = $\{t_1, t_2, t_3\}$. The term-term correlation matrix is calculated as follows:

$$\begin{array}{c} \begin{array}{cc} & \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{array}{c} t_1 \\ t_2 \\ t_3 \end{array} & \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} \end{array} \\ \mathbf{M} \end{array} \times \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{array}{c} d_1 \\ d_2 \end{array} \begin{bmatrix} w_{1,1} & w_{2,1} & w_{3,1} \\ w_{1,2} & w_{2,2} & w_{3,2} \end{bmatrix} \\ \mathbf{M}^T \end{array}$$

$$\Downarrow$$

$$\begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{array}{c} t_1 \\ t_2 \\ t_3 \end{array} \begin{bmatrix} w_{1,1}w_{1,1} + w_{1,2}w_{1,2} & w_{1,1}w_{2,1} + w_{1,2}w_{2,2} & w_{1,1}w_{3,1} + w_{1,2}w_{3,2} \\ w_{2,1}w_{1,1} + w_{2,2}w_{1,2} & w_{2,1}w_{2,1} + w_{2,2}w_{2,2} & w_{2,1}w_{3,1} + w_{2,2}w_{3,2} \\ w_{3,1}w_{1,1} + w_{3,2}w_{1,2} & w_{3,1}w_{2,1} + w_{3,2}w_{2,2} & w_{3,1}w_{3,1} + w_{3,2}w_{3,2} \end{bmatrix} \end{array}$$

It can be seen that the term-document matrix $\mathbf{M}$ has size 3 x 2, where rows consists of the terms, and columns consists of the documents. For example,

the weight in the first row, and in the first column $w_{1,1}$ contains the weight of t_1 in document d_1. The transposed matrix consists of the terms in the columns, and the documents in the rows. By multiplying these 2 matrices the term-term correlation matrix **C** is generated. It has size 3 x 3, where rows and columns consists of the terms in the vocabulary. For example, the entry in the first row, and in the second column contains the joint co-occurrence of the terms t_1 and t_2.

The vector space model is just one information retrieval model which takes advantage of term-term correlation. Other models are the set-based model, fuzzy information retrieval models, and language models.

In the vector space model documents, and queries are represented as vectors in an t-dimensional space. The size of t is defined by the size of the vocabulary. As a result, each index term represent one dimension in the vector. Ribeiro-Neto and Baeza-Yates [RB99] define the document representation d_j, and the query representation q as:

$$\vec{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j}) \tag{2.5}$$

$$\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{t,q}). \tag{2.6}$$

In both representations the vector consists of term weights which describes their content. Each document in the collection is represented by a document vector $\vec{d}_j$. Thereby $w_{i,j}$ is defined as the weight of term t_i in document d_j. It has to be non-negative, and non-binary. The term weights of the query $w_{i,q}$ consists the weight of term t_i, which occurs in query q. It has to be non-negative.

The degree of similarity between a document d_j, and a query q is calculated by the cosine of the angle between their corresponding vectors (see Figure 2.2). Specifically,

$$\begin{aligned} sim(d_j, q) &= \frac{\vec{d}_j \bullet \vec{q}}{|\vec{d}_j| \times |\vec{q}|} \\ &= \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}, \end{aligned} \tag{2.7}$$

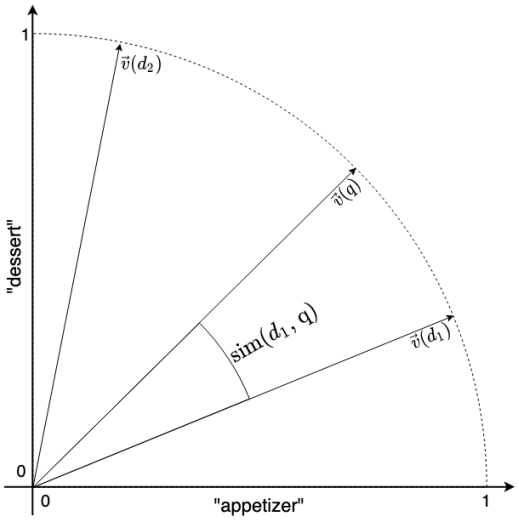


Figure 2.2: **Example of the similarity between query and documents in the vector space model.** In this example, the vocabulary of the document collection is given by V = {"*appetizer*", "*dessert*"}. The 2 documents d_1, d_2, and the query q are represented as vectors according their term weights. The similarity of 2 vectors is given by the cosine of the angle between them.

where $\vec{d}_j \bullet \vec{q}$ is the internal product of the 2 vectors. The factor $|\vec{d}_j|$ is the norm of the document vector, and $|\vec{q}|$ is the norm of query vector. These norms define the document length, and the query length. The norm of the query vector does not affect the ranking result since it is the same for all documents in the collection. Singhal et. al [SBM96] discuss in their work more advanced document length normalization for vector space models.

TF-IDF Weighting Scheme

Term weighting was first discussed by Luhn [Luh57]. The Author observed that terms that occurs more often in a document are important to describe the content of the document. Therefore, these terms can be seen as keywords. As a result, he assumed that the term frequency $f_{i,j}$, where term t_i occurs in document d_j, is relative to the term frequency weight $TF_{i,j}$. Hence, a high term frequency leads to a high term frequency weight. This assumption

leads to the following formulation of term frequency weights

$$tf_{i,j} = f_{i,j}. \tag{2.8}$$

In this formula the raw term frequency is used as term weight. However, Salton and Yang [SY73] observed in their work that in some cases term frequency weights are an improvement according binary weights. Furthermore, they state inconsistence in their test results when they changed their test collection, and query set.

To improve the results of term frequency weighting, inverse document frequency weights are used additionally. The concept of inverse document frequency was introduced by Spärck Jones [Jon72], and is one of the foundations of term weighting. Therefore, it is used in every modern information retrieval system. The inverse document frequency weight is approximated using Zipf's Law [Zip32]

$$IDF_i = \log \frac{N}{n_i}. \tag{2.9}$$

It is called inverse document frequency as n_i/N is the relative document frequency. Therefore, is n_i the number of documents, where a term k_i occurs in the document collection, and N is the size of the document collection. The inverse document frequency represents the importance of a term regarding the whole document collection. It is small if a term occurs in almost every document, and it is high if the term appears just in a few documents.

To use term frequency weighting in combination with inverse document frequency weighting the raw term frequency weight (see Equation (2.8)) has to be adopted to the logarithmic term frequency weight:

$$tf_{i,j} = \begin{cases} 1 + \log f_{i,j}, & \text{if } f_{i,j} > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

The log term frequency weight is one of the most frequently used term frequency weighing schemes as the inverse document frequency term weight is also a logarithmic function. Therefore, they can be combined directly as

defined by Salton and Yang [SY73] to the TF-IDF weighing scheme:

$$w_{i,j} = \begin{cases} 1 + \log f_{i,j} \times \log \frac{N}{n_i}, & \text{if } f_{i,j} > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{2.11}$$

where $w_{i,j}$ is the term weight of term k_i, which occurs in document d_j. On the one hand, this includes the term frequency, which represents the importance of the term within the document. On the other hand, it is composed of the inverse document frequency, which represents the importance of the term within the whole document collection. The combination of these two weights leads to an effective term weighting scheme. Therefore, it is the base for term weighting that is used in almost every modern information retrieval system.

As the TF-IDF weighting scheme is one of the most popular weighting schemes in information retrieval several variants where proposed over time. Ribeiro-Neto and Baeza-Yates [RB99] describe in their work the most important forms of term frequency weighting (see Table 2.1), and inverse document weighting (see Table 2.2). Furthermore, they discuss the best

Weighting scheme	**TF Weight**
Binary	$\{0, 1\}$
Raw Frequency	$f_{i,j}$
Log Normalization	$1 + \log f_{i,j}$
Double Normalization 0.5	$0.5 + 0.5 \frac{f_{i,j}}{max f_{i,j}}$
Double Normalization K	$K + (1 - K) \frac{f_{i,j}}{max_i f_{i,j}}$

Table 2.1: **Variants of TF weight.** There exist 5 important variants of term frequency weighting. First, *Binary* weight is the simplest form, and only captures if a term occurs in a document. Second, *Raw Frequency* uses the term frequency directly, and can be seen as base for term frequency weighting. Third, *Log Normalization* is an extension of *Raw Frequency*, and uses the logarithm of the (raw) frequency. Forth, *Double Normalization* 0.5 rescales the weights to be in the range between 0.5 and 1.0. Therefore, the weight is always normalized. Fifth, the *Double Normalization K* is a generalization of the *Double Normalization* 0.5, where K can take values in the range between 0.0 and 1.0.

combinations of them to compile different TF-IDF weighting schemes for document term weighting, as well as query term weighting.

For example, given a document $d_1 = \{"apple", "apple", "apple", "pear", "pear"\}$. When searching for *"pear"*, the individual variants of term frequency weighting (see Table 2.1) yield different results. First, *Binary* weight captures if the term occurs in the document. Therefore, the weight of d_1 is 1. Second, *Raw Frequency* denotes how often a term occurs in a document. As a result, the weight is 2. Third, *Log Normalization* uses 1 for each query term in the document, and adds the logarithmic *Raw Frequency* of the term. Hence, the weight is approximately 1.7. Fourth, *Double Normalization* 0.5 leverages the constant factor 0.5, the *Raw Frequency* of a term, and the maximum frequency over all terms to normalize weights in the range between 0.5 and 1.0. The result of the document weight is $0.8\overline{3}$. Fifth, *Double Normalization K* is a generalization of the *Double Normalization* 0.5. For example, if K is 0.1, the weighting value lies between 0.1 and 1.

Weighting scheme	**IDF Weight**
Unary	1
Inverse Frequency	$\log \frac{N}{n_i}$
Inverse Frequency Smooth	$\log(1 + \frac{N}{n_i})$
Inverse Frequency Max	$\log(1 + \frac{max_i n_i}{n_i})$
Probabilistic Inverse Frequency	$\log(\frac{N-n_i}{n_i})$

Table 2.2: **Variants of inverse document frequency weight.** There exist 5 important variants of inverse document frequency weight. First, the *Unary* form is used to ignore the inverse document frequency. Second, the *Inverse Frequency* is the standard variant as it was initial introduced. Third, the *Inverse Frequency Smooth* adds 1 to the fracture result. This produces a more representative weight for extreme values of n_i. Fourth, the *Inverse Frequency Max* uses the term with the largest document frequency instead of the number of documents in the collection. Therefore the computed weights are relative to the term with the highest document frequency. Fifth, the *Probabilistic Inverse Frequency* subtract the document frequency from the number of documents in the collection. It is the basic form for the probabilistic model, as described in the next section

To illustrate inverse term frequency, in addition to to d_1, the document collection also contains d_2 = {"*apple*", "*peach*"}. When searching for "*apple*" all documents of the collection contain the term. Therefore, the term is not useful to differentiate documents of the collection. As a result, the inverse document frequency weight is the minimum possible outcome for each weighting scheme (see Table 2.2). When searching for "*pear*" 1 of 2 documents contain the term. Hence, the individual variants of inverse document frequency weighting lead to different results. First, the *Unary* variant is always 1. Second, the *Inverse Frequency* variant denotes N as the number of all documents in the collection. Additionally, n_i is the document frequency of the query term. The logarithm of these to values leads approximately to 0.7. Third, *Inverse Frequency Smooth* also uses N and n_i. In contrast to *Inverse Frequency*, 1 is added inside the logarithm, which results of a weight of approximately 1.1. Fourth, the *Inverse Frequency Max* is similar to the *Inverse Frequency Smooth*, however uses the term with the largest document frequency instead of N. In our example that is "*apple*", which occurs in every document. Therefore, the result is also approximately 1.1. Fifth, the *Probabilistic Inverse Frequency* is similar to the *Inverse Frequency*, however subtracted N by n_i inside the logarithm. This leads to an inverse term frequency weight of 0.0.

There exist several forms of TF-IDF weighting schemes. The best performing variant could vary given different document collections. In Table 2.3 are 3 recommended combinations, which were proposed by Salton [Sal71].

Document term weight	**Query term weight**
$f_{i,j}$	$(0.5 + 0.5\frac{f_{i,q}}{max_i f_{i,q}}) \times \log \frac{N}{n_i}$
$1 + \log f_{i,j}$	$\log(1 + \frac{N}{n_i})$
$(1 + \log f_{i,j} \times \log \frac{N}{n_i})$	$(1 + \log f_{i,q}) \times \log \frac{N}{n_i}$

Table 2.3: **Recommended variants of the TF-IDF weighting scheme.** There exist 3 recommended variants of the TF-IDF weighting scheme. Given different document collections the best performing form could vary. They combine term frequency weighing variants with inverse frequency weighing variants. Additionally, they distinguish between document term weights, and query term weights.

The author distinguishes between document term weights, and query term weights.

1. The first variant combines the *Raw Frequency* of the term frequency weighing variants in Table 2.1, and the *Inverse Frequency* of the inverse frequency weighing variants in Table 2.2 for the document term weight. The associated query term weight uses *Double Normalization* 0.5 and also *Inverse Frequency*.
2. The second variant differs, as document terms consist only of a term frequency weight, and query terms consists only of a inverse frequency weight. It uses the *Log Normalization* as term frequency weight, and the *Inverse Frequency Smooth* as inverse frequency weight.
3. The third variant is the initial variant, as it was defined by Salton and Yang [SY73]. It combines the *Log Normalization*, and the *Inverse Frequency* for document-, and query terms.

The third variant is the most common used form in the vector space model. Ribeiro-Neto and Baeza-Yates [RB99] describe in their work an extension for this variant to retrieve better results. There the document term weight, and the query term weight should only be used if the term frequency is greater than 0, otherwise the corresponding weight is 0. In the web it is common that the query term frequency is 1. Therefore, it should be reduced to the inverse document frequency. If this is done, the similarity (see Equation (2.7)) captures $TF \times IDF^2$. To bring it back to the $TF \times IDF$ form, the document term weight is reduced to the term frequency weight TF. The second variant is based on the same basic principle.

The TF-IDF weighting scheme can also be used as a ranking function with limited knowledge about vectors. Manning et. al [MRS08] describe in their work such a TF-IDF ranking function:

$$\begin{aligned} \text{Score}(q,d) &= \sum_{i \in q} \text{tf-idf}_{i,d} \\ \text{tf-idf}_{i,d} &= f_{i,q} \times \log \frac{N}{n_i}, \end{aligned} \tag{2.12}$$

where the query q, and a document d are passed into the ranking function. The query and the document can be seen as sets of terms. To calculate

the rank, the TF-IDF weights are summed up. For every query term that does not occur in the document the weight is 0. For the calculation the *Raw Frequency* term frequency weighing, and the *Inverse Frequency* inverse frequency weighing are used (see Table 2.1, and Table 2.2).

The TF-IDF weighting scheme is one of the most popular weighting schemes in information retrieval as it is leveraged, adapted and optimized for a wide array of different use-cases.

2.2.3 The Probabilistic Model

The first probabilistic model was introduced by Robertson and Sparck Jones [RJ76]. In their work they defined the Probabilistic Relevance Model as framework for future models. Given a user query the task is to find the set of documents that contain all relevant documents. This set of documents is called the ideal answer set.

Properties are defined to receive the ideal answer set. The challenge is to define these properties with index terms. An additional challenge is, that the properties of the ideal answer set are not known at query time. Therefore, they are estimated to generate the initial answer set.

Afterwards, the answer set is improved by user interaction. For example, the user has to choose which documents of the answer set are relevant. With this user decisions the known properties are adapted, and a new answer set is generated. With each iteration of this process, the resulting answer set converges to the ideal answer set.

Robertson [Rob97] proposed is his work the probability ranking principle, which denotes that documents that are relevant for the user can be influenced by properties outside the system. Therefore, the ideal answer set generated by the system is not necessarily the ideal answer set for the user.

In the probabilistic model documents are represented as vectors in a t-dimensional space:

$$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j}), \tag{2.13}$$

where the size of t is the size of the vocabulary. As a result, each index term represents one dimension in the vector. Furthermore, the weights of the vector are binary. Hence, $w_{i,j} = 1$ if term k_i occurs in document d_j, and $w_{i,j} = 0$ otherwise. The query q is a subset of index terms.

Using the document representation d_j, and the query representation q, the similarity can be calculated as follows:

$$sim(d_j, q) = \frac{P(R|\vec{d_j}, q)}{P(\overline{R}|\vec{d_j}, q)}, \tag{2.14}$$

where R is a set of documents that is initially estimated, or known to be relevant for the user given a query q. $\overline{R}$ describes the complement of R (i.e., the set of documents that are not relevant for the user). Furthermore, $P(R|\vec{d_j}, q)$ defines the probability that a document d_j is relevant with respect to query q, and $P(\overline{R}|\vec{d_j}, q)$ defines the probability that the document is not relevant given this query. The equation can be approximated, as denoted by Ribeiro-Neto and Baeza-Yates [RB99], using:

$$sim(d_j, q) \sim \sum_{k_i \in q \wedge k_i \in d_j} \log\left(\frac{P(k_i|R, q)}{1 - P(k_i|R, q)}\right) + \log\left(\frac{1 - P(k_i|\overline{R}, q)}{P(k_i|\overline{R}, q)}\right), \tag{2.15}$$

where $P(k_i|R, q)$ is the probability that index term k_i is present in a document selected randomly from the set of relevant documents R. Furthermore, $P(k_i|\overline{R}, q)$ is the probability that index term k_i is not present in a document selected randomly from R. Equation (2.15) can be seen as foundation for ranking in probabilistic models.

The set of relevant documents R is not known initially. Therefore, an additional method is necessary to calculate $P(k_i|R, q)$, and $P(k_i|\overline{R}, q)$ for the initial answer set. One possible solution is a contingency table as proposed by Robertson and Sparck Jones [RJ76] (see Table 2.4).

With the contingency table $P(k_i|R,q)$, and $P(k_i|\overline{R},q)$ can be defined as:

$$P(k_i|R,q) = \frac{r_i}{R} \tag{2.16}$$

$$P(k_i|\overline{R},q) = \frac{n_i - r_i}{N - R}, \tag{2.17}$$

where the first equation captures the relevant documents that contain term k_i in relation to all relevant documents. The second equation captures the non-relevant documents that contain term k_i in relation to all relevant documents. By combining these two equations with Equation (2.15) the similarity can be computed:

$$sim(d_j,q) \sim \sum_{k_i \in q \wedge k_i \in d_j} \log \left(\frac{r_i(N - n_i - R + r_i)}{(R - r_i)(n_i - r_i)} \right). \tag{2.18}$$

To improve numerical stability of the equation, a constant of 0.5 is added ro r_i:

$$sim(d_j,q) \sim \sum_{k_i \in q \wedge k_i \in d_j} \log \left(\frac{(r_i + 0.5)(N - n_i - R + r_i + 0.5)}{(R - r_i + 0.5)(n_i - r_i + 0.5)} \right). \tag{2.19}$$

This is the classic probabilistic ranking equation, called the Robertson-Sparck Jones equation. However, it is still necessary to know the relevant documents, as R and r_i are part of the equation. One solution is to set R,

Case	Relevant	Non-relevant	Total
Documents containing k_i	r_i	$n_i - r_i$	n_i
Documents not containing k_i	$R - r_i$	$N - n_i - (R - r_i)$	$N - n_i$
All documents	R	$N - R$	N

Table 2.4: **Contingency table for probabilistic ranking.** The contingency table represents the relationships between documents and relevance. N is the number of all documents in the collection, n_i defines the number of documents that contain an index term k_i. Furthermore, R is the number of documents that are relevant for a user with respect to a query q, r_i defines the number of relevant documents that contain an index term k_i.

and r_i to be 0:

$$sim(d_j, q) \sim \sum_{k_i \in q \wedge k_i \in d_j} \log \left(\frac{N - n_i + 0.5}{n_i + 0.5} \right). \tag{2.20}$$

This equation is used to calculate the initial set of relevant documents. Robertson [Robo4] proposed is his work that the equation has negative terms if $n_i > N/2$. Therefore, he defined an alternative form by removing the subtraction of n_i in the numerator:

$$sim(d_j, q) \sim \sum_{k_i \in q \wedge k_i \in d_j} \log \left(\frac{N + 0.5}{n_i + 0.5} \right). \tag{2.21}$$

As a result, the term of the sum is zero if the index term occurs in each document. Furthermore, it can be determined that the calculation for one term is similar to the calculation of the inverse document frequency (see Equation (2.9)).

In theory, the probabilistic model is optimal, as documents are ordered by the probability of being relevant for the user. In practice the system is affected by external factors. Therefore, finding the ideal answer set is only theoretically possible. An additional disadvantages is that the initial relevant-, and not-relevant sets have to be estimated. Furthermore, term frequency is not used, as document weights are binary. More advanced extensions of the probabilistic model tackle some of the outlined issues. For example, BM25, which will be discussed in the text.

Okapi BM25

Okapi BM25 is the result of several experiments by Robertson et. al [Rob+92; Rob+93; Rob+94]. The idea is to transfer the strengths of the vector space model to the probabilistic model. These advantages are inverse document frequency, term frequency, and document length normalization. As foundation the classic probabilistic ranking equation (see Equation (2.20)) is used. The equation already covers the usage of the inverse document frequency, and is known as BM1.

To bring the term frequency into the probabilistic model the following equation was designed:

$$\mathcal{F}_{i,j} = S_1 \times \frac{f_{i,j}}{K_1 + f_{i,j}}, \tag{2.22}$$

where $f_{i,j}$ is the term frequency of term k_i that occurs in document d_j. Furthermore, K_1 is a constant that is determined based on the document collection. When K_1 is set to 0 the factor becomes 1, and the term frequency is not used for ranking. Finally, S_1 is used as a scaling factor.

The next step was to bring document length normalization into the probabilistic model. To that end, the above equation was adjusted:

$$\mathcal{F}'_{i,j} = S_1 \times \frac{f_{i,j}}{\frac{K_1 \times len(d_j)}{avg\_doclen} + f_{i,j}}, \tag{2.23}$$

where $len(d_j)$ is the number of terms in the document, and avg_doclen is the average document length over all documents in the collection. Furthermore, a correction factor for the document length, and the query length is introduced:

$$\mathcal{G}_{i,q} = K_2 \times len(q) \times \frac{avg\_doclen - len(d_j)}{avg\_doclen + len(d_j)}, \tag{2.24}$$

where $len(q)$ is the number of terms in the query, and K_2 is a constant. To introduce term frequency into queries, an additional equation was designed based on Equation (2.22):

$$\mathcal{F}_{i,q} = S_3 \times \frac{f_{i,q}}{K_3 + f_{i,q}}, \tag{2.25}$$

where $f_{i,q}$ is the term frequency of term k_i that occurs in query q. K_3 is an additional constant, and is determined based on queries. S_3 is a scaling factor for the fraction term.

With these 4 equations BM15, and BM11 can be defined:

$$sim_{\text{BM15}}(d_j, q) \sim \mathcal{G}_{i,q} + \sum_{k_i \in q \wedge k_i \in d_j} \mathcal{F}_{i,j} \times \mathcal{F}_{i,q} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right) \quad (2.26)$$

$$sim_{\text{BM11}}(d_j, q) \sim \mathcal{G}_{i,q} + \sum_{k_i \in q \wedge k_i \in d_j} \mathcal{F}'_{i,j} \times \mathcal{F}_{i,q} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right). \quad (2.27)$$

BM15 uses $\mathcal{F}_{i,j}$ without the extension of document length normalization, as introduced in BM11. Robertson and Walker [RW94] observed in their work that the best value for $K_2 = 0$. As a result, the correction factor $\mathcal{G}_{i,q}$ is eliminated. Furthermore, they suggest $S_1 = (K_1 + 1)$, and $S_3 = (K_3 + 1)$, where the evaluation results increased between 12 and 37 percent (depending on the query length) if K_3 was chosen large (e.g., 100 as in their experiments). Finally, they observed that the query term frequency factor $\mathcal{F}_{i,q}$ can be reduced to $f_{i,q}$, and for small queries set to be 1. Adding these findings to the BM15-, and BM11-equation they can be simplified:

$$sim_{\text{BM15}}(d_j, q) \sim \sum_{k_i \in q \wedge k_i \in d_j} \frac{(K_1 + 1) f_{i,j}}{(K_1 + f_{i,j})} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right) \quad (2.28)$$

$$sim_{\text{BM11}}(d_j, q) \sim \sum_{k_i \in q \wedge k_i \in d_j} \frac{(K_1 + 1) f_{i,j}}{\frac{K_1 \times len(d_j)}{avg\_doclen} + f_{i,j}} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right). \quad (2.29)$$

In these equations the correction factor, and the query term frequency factor are removed. Furthermore, S_1 is set to be $K_1 + 1$. As the document length normalization is only part of BM11 it outperforms the BM15.

BM25 was designed as a combination of BM15, and BM11. As a result, their term frequency factors (see Equation (2.22), and Equation (2.23)) was merged to generate the term frequency factor of BM25:

$$\mathcal{B}_{i,j} = \frac{(K_1 + 1) f_{i,j}}{K_1 \left[(1 - b) + b \frac{len(d_j)}{avg\_doclen}\right] + f_{i,j}}, \quad (2.30)$$

where b is a constant that takes values between 0.0 and 1.0. On the one hand, for $b = 0$ the equation is reduced to the BM15 term frequency factor.

On the other hand, for $b = 1$ the equation is reduced to the BM11 term frequency factor. When using $\mathcal{B}_{i,j}$ BM25 is defined as follows:

$$sim_{\text{BM25}}(d_j, q) \sim \sum_{k_i \in q \wedge k_i \in d_j} \mathcal{B}_{i,j} \times \log \left(\frac{N - n_i + 0.5}{n_i + 0.5} \right). \tag{2.31}$$

Robertson et. al [Rob+94] find in their work that $b = 0.75$, and $K_1 = 1$ is the best choice for most cases. In general, they proposed that b should be close to 1 to apply the document length normalization.

BM25 transfer the strengths of the vector space model to the probabilistic model. It requires additional tuning, but achieves better performance than the vector space model for general document collections. Therefore, it has become to a baseline to evaluate new ranking methods.

Divergence from Randomness

The Divergence from Randomness model was introduced by Amati and Rijsbergen [ARo2]. It is a probabilistic model that exhibits characteristics of a language model as well. In the model, the term weights are computed by evaluating the divergence between the actual term distribution and the term distribution generated by a random process.

The model is based on 2 assumptions:

1. First, terms of a document have different importance when describing the content of it. It assumes that less important words are distributed randomly over the document collection C. The probability distribution of a term k_i over the document collection C is given by $P(k_i|C)$. Furthermore, the amount of information of these terms over the whole document collection is defined as $-\log P(K_i|C)$.
2. Second, a complementary term distribution is received by considering only the subset of documents that contain term k_i. This subset is called elite set. The underlying probability distribution is defined as $P(k_i|d_j)$. The probability is high if the term k_i occurs often in document d_j.

Furthermore, if a term occurs rarely in a document, it is important to describe the content of the document. Therefore, the amount of information of these terms being in the elite set is given by $1 - P(k_i|d_j)$.

By combining these 2 assumptions the term weight is given by:

$$w_{i,j} = (-\log P(k_i|C)) \times (1 - P(k_i|d_j)), \tag{2.32}$$

where $w_{i,j}$ is the term weight of term k_i, which occurs in document d_j. Furthermore, the ranking function is defined as:

$$R(d_j, q) = \sum_{k_i \in q} f_{i,q} \times w_{i,j}, \tag{2.33}$$

where $f_{i,q}$ is the term frequency of term k_i in query q. To approximate the first term of the term weight the following definition are needed:

$$F_i = \sum_j f_{i,j} \tag{2.34}$$

$$\lambda_i = p \times F_i, \tag{2.35}$$

where F_i is the frequency of a term k_i over all documents in the collection. For Equation (2.35), p is defined to be $p = 1/N$.

The first part of the term weight (see Equation (2.32)) represents the amount of information of a term over the entire document collection. Amati and Rijsbergen [AR02] propose in their work 2 approximations:

$$\begin{aligned} -\log P(k_i|C) \approx & f_{i,j} \log\left(\frac{f_{i,j}}{\lambda_i}\right) + \left(\lambda_i + \frac{1}{12 f_{i,j} + 1} - f_{i,j}\right) \log e \\ & + \frac{1}{2} \log(2\pi f_{i,j}) \end{aligned} \tag{2.36}$$

$$-\log P(k_i|C) \approx -\log\left(\frac{1}{1+\lambda_i}\right) - f_{i,j} \times \log\left(\frac{\lambda_i}{1+\lambda_i}\right), \tag{2.37}$$

where $f_{i,j}$ is the frequency of term k_i in document d_j. For the second approximation $p = 1/(1 + \lambda_i)$.

The second part of the term weight (see Equation (2.32)) represents the amount of information of a term with respect to the elite set, which is calculated by:

$$1 - P(k_i|d_j) = \frac{1}{f_{i,j} + 1} \tag{2.38}$$

$$1 - P(k_i|d_j) = \frac{F_i + 1}{n_i \times (f_{i,j} + 1)}, \tag{2.39}$$

where n_i is the number of documents that contain term k_i.

The basic form of the Divergence from Randomness model does not take document length normalization into account. Therefore, there exist 2 extensions for the term frequency $f_{i,j}$. Each of them can be used to add document length normalization:

$$f'_{i,j} = f_{i,j} \times \frac{avg\_doclen}{len(d_j)} \tag{2.40}$$

$$f'_{i,j} = f_{i,j} \times \log\left(1 + \frac{avg\_doclen}{len(d_j)}\right), \tag{2.41}$$

where avg_doclen is the average document length over the entire collection, and $len(d_j)$ is the number of terms inside document d_j.

For a practical implementation of Divergence from Randomness the previously discussed alternatives of the individual parts of the equation (see Equation (2.36), Equation (2.37), Equation (2.38), Equation (2.39), Equation (2.40), and Equation (2.41)) can arbitrary be combined. Therefore, the best performing variant depends on the given document collection.

2.3 Structured Text Retrieval

Text documents always contain a certain structure. For example, a scientific paper can be split by its sections, subsections, and paragraphs. Another

example would be a book that is divided by pages and columns. Information retrieval models that leverages document structure, in addition to the content, are called structured text retrieval models. There are several stages in which the information retrieval systems can take the advantage of this structural information:

1. At index stage, where components of a document are detected, and indexed separately. The relationship between the components is preserved.
2. At retrieval stage, by allowing components to be retrieved in varying granularity.
3. As result of the presentation stage, where only relevant components are returned to the user and not the entire document.
4. At querying stage, where a query language is used that includes structural constrains.

Information retrieval models use structural information either in an explicit or implicit way. More frequently, an explicit structure is used where documents are structured in a defined scheme (e.g., XML scheme). The advantage of this structure is that relationships between the components of the document remain. For example, a nested component knows its parent, and the parent knows its children.

For implicit structure, it is not possible to distinguish between content and structural information. Documents are represented as a sequence of text tokens and mark-up tokens. At query time the sequence of tokens is searched for opening and closing mark-up tokens. Afterwards the structural component is generated with the content between these mark-up tokens. It exists only at query time and is discarded by the system afterwards.

In an information retrieval system it is possible to use multiple layers. For example, the first layer represents the logical structure of a scientific paper. Therefore, it contains the information about chapters, sections, subsections, and paragraphs. The second layer represents the layout structure. Hence, it contains information about columns and pages. Multiple layers are usually used for explicit structure, as maintaining them for implicit structure is too complex.

Inside one layer, content of a document is always assignable to one component (e.g., text in a subsection). If this is the case, components do not overlap. Across different layers it is possible that content is assigned to multiple components. Therefore, the components overlap. Alink et al. [Ali+06] tackle in their work the possibility to mix layers in one query. Therefore, they defined navigation steps to allow changing between layers.

One of the first models that use structured text retrieval was proposed by Burkowski [Bur92a; Bur92b]. It is based on non-overlapping lists. Therefore, a list for each type of document component is generated. For example, a first list contains all sections, a second list all subsections, and a third list all subsubsections. An inverted index is generated to search for a term. The inverted index stores a mapping from index terms to occurrences. Queries that contain terms and component types are used to search for content within the lists.

Another model was introduced by Baeza-Yates and Navarro [NB95; NB97] that is based on proximal nodes. The model uses hierarchical index structures to store structural information. A more targeted search is possible as relationships between components are stored.

2.3.1 Ranking Strategies in XML Retrieval

When XML was introduced in 1998 it became to the standard in structured text retrieval. Nowadays XML Retrieval is almost a synonym for Structured Text Retrieval.

In the XML Retrieval documents are represented in a XML structure. Ranking Algorithms are used (cf. Section 2.2) to calculate ranks for document components. To that end, documents are represented as set of terms, and components are subsets of these terms. In addition, calculations require a ranking strategy, as ranking of structured text differs from the ranking of unstructured text. For example, a single component does not have to contain all query terms, as they can occur in different parts of the document.

The first ranking strategy is known as contextualization. For this strategy, the rank of the component is combined with the rank of its root element, which is the document. Arvola et al. [AJK05] suggest in their work to use the average of these 2 ranks. In addition, Mass and Mandelbrod [MM04] introduced in their work a scaling factor for each rank.

The second ranking strategy is named propagation. It requires that only leaf elements are indexed. The rank of the document is calculated from bottom-up. First, the rank of the leaf elements is calculated. Afterwards, the rank of the parent is calculated based on its children ranks. This is continued until the root element is reached. The most common propagation mechanisms are based on weighted sums. Geva [Gev06] introduced in his work a weighing based on the number of children:

$$score(e, q) = D(m) \times \sum_{e_c} score(e_c, q), \tag{2.42}$$

where e is the component, e_c are the child components of e, and m is the number of retrieved child components of e. Retrieved indicates that $score(e_c, q) > 0$ holds. Furthermore, $D(m) = 0.49$ if e has just one retrieved child ($m = 1$), otherwise $D(m) = 0.99$. Propagation mechanisms are more complex to implement than contextualization mechanisms, but provide good retrieval performance.

The third ranking strategy is called aggregation and is proposed by Chiaramella et al. [CMF96]. In their work they describe that the representation of a XML element can be interpreted as an aggregation of its own and its children content representations. Linear interpolation can be used as aggregation strategy. To calculate the probability for leaf elements the following definitions are needed:

$$P(q, M_e) = \prod_{k_i \in q} P(k_i | M_e, \lambda) \tag{2.43}$$

$$P(k_i | M_e, \lambda) = \lambda P(k_i | e) + (1 - \lambda) P(k_i | C), \tag{2.44}$$

where $P(k_i|e)$ is the probability of query term k_i in component e, and yields the element models term frequency. $P(k_i|C)$ is the probability of query term k_i in the collection C, and represents the element models inverse

document frequency (see Section 2.2.2 for different approaches to compute term frequency, and inverse document frequency). The constant λ is used as smoothing parameter. Equation (2.43) represents the probability of a query q being generated by a components language model M_e. Afterwards, the aggregation strategy based on linear interpolation is defined as:

$$P(k_i|M_e) = \sum_{e_j} w_j P(k_i|M_{e_j}), \tag{2.45}$$

where $\sum_{e_j} w_j = 1$. The rank of the leaf elements is generated by Equation (2.43). For example, a section s_1 contains 3 paragraphs p_1, p_2, and p_3. Furthermore, the probabilities when searching for *"apple"* are defined as $P("apple"|M_{p_1}) = 0.26$, $P("apple"|M_{p_2}) = 0.49$, and $P("apple"|M_{p_3}) = 0.15$. When *"apple"* is passed as query into the aggregation function then $P("apple"|M_{s_1}) = 1/3 \times (0.26 + 0.49 + 0.15) = 0.3$. Where $w_j = 1/3$ for all components as they contribute equally.

The fourth strategy is named merging. It requires a selective index strategy as described by Mass and Mandelbrod [MM04], where a separate index is created for each component type (similar to the model based on non-overlapping lists). Afterwards, the components are ranked for each type. As a result, ranked lists are created for each type. To merge the list normalization is required, as components of different types have different lengths (e.g., the length of a section, and the length of a paragraph). Hence, the normalization of components is defined as follows:

$$\max\left(\frac{score(e,q)}{score(q,q)}, 1\right), \tag{2.46}$$

where $score(q,q)$ is taken for normalization. There, any element that is identical to the query obtains the full score of 1. Afterwards, the ranked lists of components can be merged according their normalized score.

Manning et al. [MRS08] propose in their work a method to include zone scores. Their approach assigns a weight between 0.0 and 1.0 to a document d with respect to a query q. This is done by linear combinations of zone

scores, where each zone of the document contributes to a boolean value:

$$\sum_{i=1}^{l} g_i s_i, \tag{2.47}$$

where l is the number of zones, and g_i are the zone scores. Each zone score is between 0.0 and 1.0, and $\sum_{i=1}^{l} = 1$. The parameter s_i is the boolean score that represents a match between the query and the ith zone. The weighted zone scoring is also known as Ranked Boolean Retrieval.

There exist many more strategies to rank structured text. Some of them require special indexing strategies. The best ranking strategy always strongly depends on the used document collection.

2.4 IMRaD Structure

To increase the readability within articles it is important to have a common structure. Therefore, if the reader is familiar with the structure of the article it is easier to find the relevant pieces of information. Today the IMRaD (Introduction, Method, Results and Discussion) structure is a commonly agreed structure of scientific papers. Within the structure the chapters contain the following information:

- The *Introduction* section should answer the question why the study was done. Furthermore, it contains research questions, a hypothesis, and a research purpose.
- The *Method* section should answer the questions when, where, and how the study was done. Additionally, it contains information about the used materials.
- The *Results* section contains the answers to the research questions. In addition, the study results are listed and explained according to the hypothesis.
- The *Discussion* section should interpret the results. Furthermore, they are compared with results of other researchers and future work is discussed.

When having a look on the history of scientific articles their structure changed over time. Meadows [Mea85] describes in his work that the first articles appear in the the 17th century. At this time they are published in the form of descriptive letters and chronologically structured narratives. This format was used for more than 2 centuries.

Day [Day89] propose in his work that the first IMRaD-like writing structure was generated by *Louis Pasteur* with his book in *"Etudes sur la Biere"* originally published in 1876. In the book *Introduction*, *Method*, and *Discussion* can be found although the headings of the section were different. Sollaci and Pereira [SP04] describe in their work that the IMRaD structure began to be adopted in the 1940s. Furthermore, they propose that it became the standard format for scientific articles in the 1970s. Today, IMRaD is the standard for all major journals.

Although the format of IMRaD was introduced there are minor differences in the arrangement of chapters. Bertin et al. [Ber+13] analyzed articles of several journals according their IMRaD structure distribution. They used 7 peer-reviewed academic journals from the Public Library of Science (PLOS), compiled of scientific articles in the fields of biology and medicine. The largest journal is *PLOS ONE*, which contains 33,782 articles. On average an article of *PLOS ONE* consist of 4.47 sections, and 90.2% of the sections can

Name of Section	**Section** 1	**Section** 2	**Section** 3	**Section** 4
Introduction	**99.90**%	0.07%	0.00%	0.00%
Method	0.00%	47.82%	5.95%	**46.86**%
Results	0.00%	**51.48**%	**47.90**%	0.12%
Discussion	0.00%	0.08%	45.79%	46.78%
Other	0.09%	0.55%	0.36%	6.24%
Total	100.00%	100.00%	100.00%	100.00%

Table 2.5: **IMRaD chapter distribution.** For the IMRaD chapter distribution the *PLOS (Public Library of Science) ONE* journal which consists of 33,782 articles was analyzed. In this Table the relation between section and the IMRaD type is shown. For example, 99.90% of first sections are Introductions and 51.48% of second sections are Results.

be assigned to an IMRaD type.

In Table 2.5 the IMRaD chapter distribution for *PLOS ONE* is shown. The distribution represents the relation between section and the IMRaD type. For example, 99.90% of the articles has *Introduction* as first section. The other IMRaD types cannot be assigned so clearly to a section. *Method* is mostly assigned to the second or the forth section, *Results* to the second or third section, and *Discussion* to the third or fourth section.

The structure of articles is commonly used in information retrieval systems as it provides additional information. For example, the IMRaD structure can be leveraged by dividing articles according to their IMRaD sections. Therefore, section headings have to be mapped to IMRaD sections. This is done during the index process by classifying extracted section headings. For example, to use a query like "*network* IN `Methods`" the methods section has to be known for all articles in the document collection.

Ahmed and Afzal [AA20] describe in their work that dictionary terms, the template of a paper, and in-text citation are state-of-the-art technologies to map section headings onto logical sections. Furthermore, they proposed a novel approach that employs new features for these technologies.

In Summary, IMRaD is widespread as it has been used for centuries. Due to its clear structure and comprehensibility, it quickly became the standard for scientific articles. This is also the reason why it can be found in the publications of many scientific fields.

3 Materials and Method

3.1 Dataset

3.1.1 Generation

We created our dataset from approximately 3,000 scientific articles in PDF format. An important point was that these articles come from different scientific fields.

We used a text mining pre-processing technique as introduced by Vijayarani et al. [VIN15] to separate the text from the PDFs and add additional information. This technique consists of three key steps, which are called extraction, stopword removal, and stemming. First, we used a framework described in [Kla+14] to separated the article structure and the raw text from the PDF. Afterwards the stopwords are removed, and the remaining terms of the raw text are stemmed.

One additional information we had to add was IMRaD structure (see Section 2.4). The IMRaD structure maps *Related Work* to be part of the *Introduction*. Since most of our papers had an own section titled *Related Work* we introduce an additional type called *Background* for it. We were able to classify the IMRaD structure with simple keyword detection in the section titles. Table 3.1 shows the mapping between IMRaD-Types and these titles. Because it was not really possible to identify *Method* sections by using the section titles we used information about their position in the article. We manage that by using the *Background* section as upper bound, and the *Result*

section as lower bound. We classify all sections between these two bounds as a *Method* section. When the *Background* section was not available, we set the *Introduction* section as upper bound. Additionally, when the *Result* section was not available, we set the *Discussion* section as lower bound. If one of the two bounds could not be set, we discarded the scientific article. Note that, chapters can have several IMRaD-Types. For example, if a section was titled *"Results and Discussion"* it belongs to the types *Result* and *Discussion*.

Another additional detail we had to add were links between the scientific articles. For this we performed a semi-automated annotation. Thereby similarities about the references of an article and the titles of all other articles are compared, and if they exceed a given threshold a recommendation to create a link between the two was given.

We created each data record in such a way that it can be transferred directly to the database schema shown in Figure 3.1. To reduce noise during the

		per Section		Overall	
IMRaD Type	**Section Title**	**# Paper**	**Percent**	**# Paper**	**Percent**
Introduction	Introduction	822	100%	822	100%
Background	Related Work	465	56.57%	465	56.57%
Methods	Method	97	11.8%	312	37.96%
	Model	134	16.3%		
	Approach	81	9.85%		
Result	Experience	396	48.18%	687	83.58%
	Result	163	19.83%		
	Evaluation	128	15.57%		
Discussion	Conclusion	581	70.68%	773	94.04%
	Discussion	179	21.78%		
	Future Work	13	1.58%		

Table 3.1: **Mapping of the Section Titles to IMRaD-Types.** In this Table we show which section titles was used to generate the IMRaD structure information. Additionally, we show the relation between titles and how often they occurs in the used dataset.

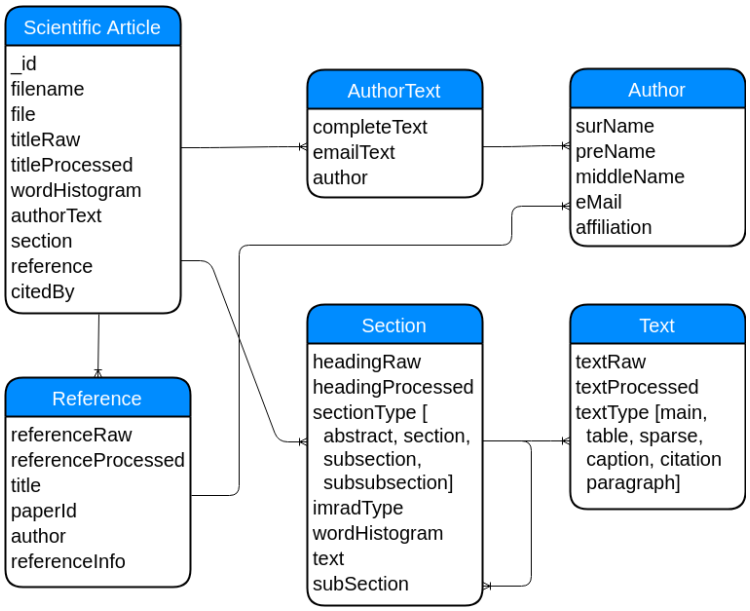


Figure 3.1: Database Schema for the used Dataset

evaluation we removed all articles without any connection to other articles.

Finally, we have 821 scientific articles in our dataset. This are only 27 percent of the initial set. This small number is due to the environment and the used scientific articles. One major problem was that the framework used to separate the structure had issues with documents that were not created with latex.

3.1.2 Structure of a Scientific Article

We designed a database schema, which corresponds to the structure of a scientific article. The table ”scientific article” in the schema (see Figure 3.1) can be seen as the root node for each database entry.

The ”author text” attribute refers to the table ”AuthorText”. It contains the names and email addresses of all associated authors. These values are

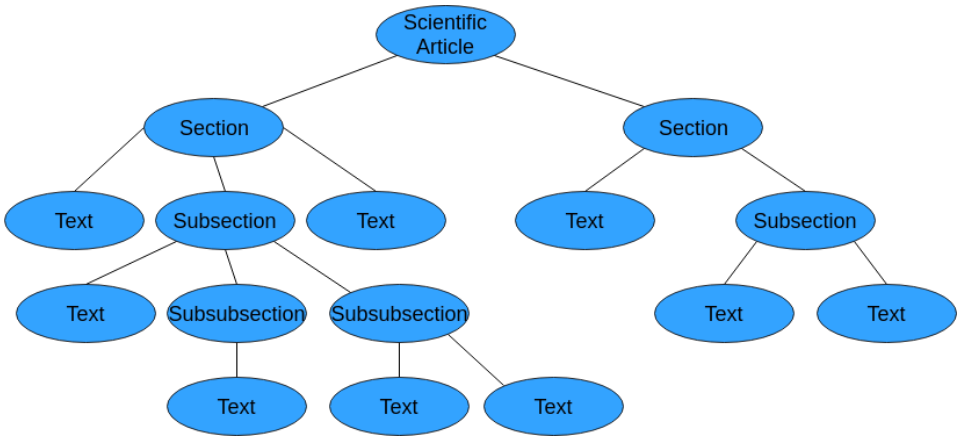


Figure 3.2: **Example of a Scientific Article Tree.** In this figure we highlight the hierarchical structure of an typical scientific article. For example it can be composed into multiple sections.

generated from the author area, which is normally on the first page of each paper. In the database schema this attribute consists of three values. First, the complete text, which contains the whole author area. Second, the email text, which are all email addresses separated from the complete text. Finally, a list with all authors. The author area is the only area that is not prepossessed.

One of the most important characteristics are sections, and the underlying structure. Figure 3.2 shows an example of an article, and the tree-like structure that comes with it. Chapters are non-leaf nodes, and text areas are leaf nodes. This is represented in database schema as two lists, one for subsections and one for text areas. In addition to the lists, each section itself has a section type. This attribute describes whether the section is a section, subsection, or subsubsection. The IMRaD structure information is stored as the IMRaD-Type. As described in Section 3.1.1, each section may have several IMRaD-Types. Each type of section holds its own list of these types. Hence, subsections and subsubsections keep the same IMRaD-Types as their parent section.

We store word histograms for articles as well as the sections, so we do not have to scan the entire text for each search request. These histograms

contain the term frequencies of the corresponding area. Therefore, subsubsections contain the frequencies of their text areas, subsections contain the frequencies of their text areas, and their subsubsections text areas, and so on. Finally, an article holds the term frequencies of the whole document.

The last two attributes of an article are the reference-, and the cited-by-lists. The two lists are used to generate connections between articles. A reference holds the identifier of a referred paper. In turn, the referred paper has an entry with the paper id in the cited-by-list. Additionally, we store the text of whole reference, the authors, and other available information such as publisher, pages, or the volume.

One characteristic over all tables of the schema is, that all extracted text values are available as raw-, and processed text values. Raw represents the text as it was separated from the used framework. Processed is the raw text after the stopword removal and the stemming.

3.1.3 Citation Network

A Citation network represents the relationship between scientific articles. In general citation means that one article mentions the work of another article. Therefore a reference with the title, the authors and the publication journal is added. Figure 3.3 shows the structure of such a network. Scientific articles are nodes, and citations are directed edges between these nodes. The timeline indicates that new articles are citing already existing articles, and thus there can not be cyclic dependencies.

M. Kas [Kas11] defined the basic properties of citation networks in their work. In our case the most important ones are:

- Directed.
- Acyclic.
- All edges point backwards in time
- Edges are permanent

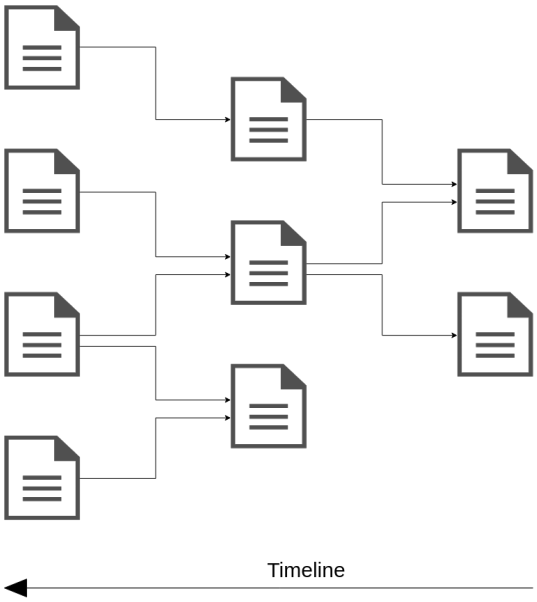


Figure 3.3: **General Structure of a Citation Network.** The timeline indicates that new articles citing existing articles, and thus there can not be cyclic dependencies between them.

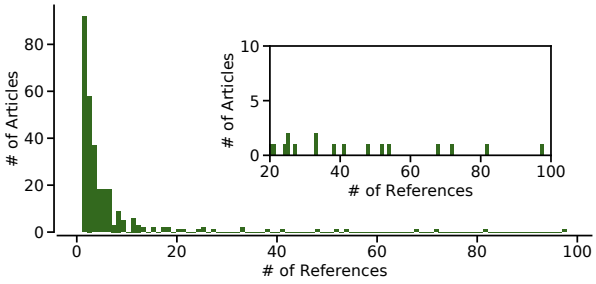
- The existing part is mostly constant. Only the leading edges changes

The network is directed and acyclic because each article has a publication date and can only cite previously published articles. Due to these properties edges can only point back in time. The edges of the network have to be permanent because the references of the existing articles never change. When a new node is added it generates edges to existing articles. This means that all other nodes and edges stays constant.

The main properties of our citation network are shown in Table 3.2. Scientific articles are the nodes, and citations are the edges between these nodes. This means that our network consists of 821 articles, and 1,716 citations between these articles. During the generation of our dataset we found cycles due to preprints. Preprints are versions of scientific articles which are not peer reviewed, and published in a scientific journal. In our case we removed all preprints from the dataset. The longest path length indicates that the longest citation chain of our network consists of 12 articles. The root nodes

Max References	98
Mean References	5.8767
Median References	2

(a) Properties



(b) In-Degree Distribution

Figure 3.4: **In-Degree Distribution.** The in-degree distribution describes how often articles get referred by other articles. The long tail of the distribution indicates that there are a lot of articles which are cited only a few times, and a few articles which are cited more often. There are some outliers with a higher degree than 20. This can also be seen by the difference between the mean and the median. The maximum number of references in connection with the zoomed view represents that one single article was cited by 98 other articles.

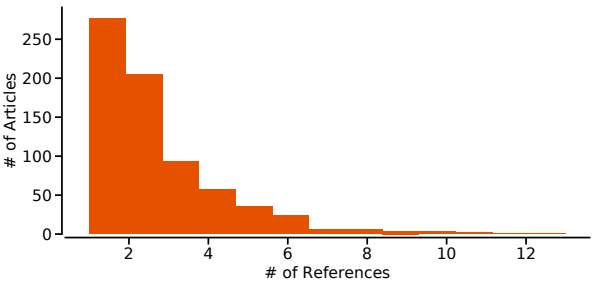
are nodes without any outgoing edges. In our network are 107 root articles which cite no other article. This happens because none of their referred articles are part of our dataset.

Figure 3.4 describes the in-degree distribution of our citation network. In general, the in-degree of a node is the number of ingoing edges. The in-

Number of Nodes	821
Number of Edges	1,716
Longest Path Length	12
Number of Root Nodes	107

Table 3.2: **General Properties about the citation network.** The citation network represents the relationship between our used scientific articles. The number of nodes indicates the number of articles, and the number of edges citations between those articles. There are no cycles inside the graph, and the longest citation chain consists of 12 articles. There are 107 articles which has no outgoing edges. That means that none of their referred articles are part of our dataset.

Max References	13
Mean References	2.4034
Median References	2



(a) Properties (b) Out-Degree Distribution

Figure 3.5: **Out-Degree Distribution.** The out-degree distribution describes how often articles refer other articles. The long tail of the distribution indicates that there are a lot of articles which refer less than 7 other articles, and only few with refer to more articles. Mean and median of the outgoing edges are low, because not every refereed article is part of our dataset. By the small difference between the mean and the median can be seen that there are less outliers. The maximum number of references represent that the highest number of a single article refers other articles is 13.

degree distribution represents the probability distribution of these nodes over the whole network. Regarding a citation network the in-degree of a node is the number of articles which referred to this article. The long tail of the in-degree distribution indicates that there are a lot of articles which are referred only a few times, and a few articles which are referred more often. There are only some articles with an in-degree higher than 20. The maximum number of references in connection with the zoomed view represents that one single article was referred by 98 other articles.

The out-degree distribution and their properties are displayed in Figure 3.5. In contrast to the in-degree distribution, the out-degree distribution describes the number of outgoing edges. Regarding to the citation network, the out-degree of a node can be described as the number of articles which gets referred by this article. The long tail of the out-degree distribution indicates that there are a lot of articles which refer less than 7 other articles, and only few with refer 7 or more articles. Mean and median of the outgoing edges are low, because not every refereed article is part of our dataset. By the small difference between mean and median we can also see that there

are less outliers. The maximum number of references represent the highest number of a single article refers to other articles, which is in our case 13.

3.2 Model

An information retrieval model is defined by the quadruple $[\mathbf{D}, \mathbf{Q}, \mathcal{F}, \mathcal{R}(q_i, d_j)]$ that contains the design of documents in the document collection $\mathbf{D}$, queries $\mathbf{Q}$, the framework $\mathcal{F}$, and the ranking function $\mathcal{R}(q_i, d_j)$ (cf. Section 2.1). We design our system with the goal to compare various common ranking algorithms. Therefore, we generate a model that allows that ranking algorithms can easily be exchanged by a configuration parameter. Additionally, our model is designed to work with unstructured as well as structured data. This is reflected by the query language.

For structured text retrieval, we structure the documents according to their IMRaD sections (see Section 2.4). In our dataset the background chapter is available in addition to the IMRaD chapters. Therefore, it was introduced as an additional IMRaD type.

We define documents to provide data unstructured and structured. Hence, each document in the dataset contains a bag of words where all index terms of its text content (DBW_{All}) are stored. In a bag of words each term is represented as a pair of term and term frequency. Additionally, the documents contain 5 bag of words for each IMRaD-type ($DBW_{Introduction}$, $DBW_{Background}$, $DBW_{Methods}$, $DBW_{Results}$, $DBW_{Discussion}$), where:

$$\begin{aligned} DBW_{IMRaD} = & DBW_{Introduction} \cup DBW_{Background} \cup DBW_{Methods} \\ & \cup DBW_{Results} \cup DBW_{Discussion}. \end{aligned} \tag{3.1}$$

DBW_{IMRaD} differs from DBW_{All}, as there exist areas in the document that cannot be assigned to an IMRaD-type (e.g., Abstract).

We remove stopwords and stem words during the indexing process of a document to generate index terms. Afterwards, we assign the index terms

to bag of words of a document. For example, *"I love deadlines."* is in the *Introduction* and *"I like the whooshing sound they make as they fly by."* is in the *Methods* of a document A. The underlying bags of words in the document are given by:

$$\begin{aligned} DBW_{All} &= \{"love" : 1, "deadlin" : 1, "whoosh" : 1, "sound" : 1, "fly" : 1\} \\ DBW_{Introduction} &= \{"love" : 1, "deadlin" : 1\} \\ DBW_{Methods} &= \{"whoosh" : 1, "sound" : 1, "fly" : 1\}. \end{aligned} \tag{3.2}$$

We represent IMRaD sections that do not occur in the document by an empty bag of words. In this example, the union of $DBW_{Introduction}$ and $DBW_{Methods}$ is identical to DBW_{All}. When we define A to additionally contain *"I love music."* in the *Abstract* then DBW_{All} changes, but $DBW_{Introduction}$ and $DBW_{Methods}$ stay the same.

We define the query language in a way that the user can choose if unstructured or structured text retrieval is used. Therefore, we design an IN-statement that connects query terms and IMRaD-Types. For example, with the query *"local*, *network* IN `Methods`" a user can specify that the terms *local* and *network* should occur in the `Methods` section of a document.

The left part of the IN-statement is converted into a set of query terms. Hence, multiple occurrences of a single term in the query does not affect the outcome. The right part of the IN-statement captures in which set the query term is stored. Therefore, given the 5 IMRaD-Types we defined 6 query sets (QS_{All}, $QS_{Introduction}$, $QS_{Background}$, $QS_{Methods}$, $QS_{Results}$, $QS_{Discussion}$). The query sets are connected directly to the documents bags of words in the framework. For example, the terms in $QS_{Introduction}$ are used to search in $DBW_{Introduction}$.

When a query contains terms without any IN-statements the terms are added to QS_{All}. Hence, query terms are stored in an IMRaD query set if an IN-statement is present, or in QS_{All} otherwise. As a result, query terms are searched in DBW_{All} or DBW_{IMRaD}, but never in both of them. This query structure avoids the usage of unstructured and structured text retrieval with a single request.

We introduce an AND-statement to combine multiple IN-statements. For example, "*area* IN `Background`" AND "*local*, *network* IN `Methods`" expresses that *area* should occur in `Background` section and the terms *local* and *network* should occur in the `Methods` section of a document. Therefore, "*area*" is added to $QS_{Background}$ and "*local*", "*network*" is added to $QS_{Methods}$. The definition of the AND-statement allows to search multiple bags of words with a single request. We combine the query sets on the left part of the IN-statement at query time when an IMRaD type appears multiple times. The same holds when unstructured text retrieval is used (e.g., "*area*" AND "*local*, *network*").

Only one query term has to occur in the specified bag of words to mark a document as relevant. This increases the number of relevant documents in the resulting ranked list. For example, a query is given by "*plane* IN `Introduction`" AND "*fly* IN `Methods`". When document A in the previous example (see Equation (3.2)) is searched with respect to the defined query it will be marked as relevant. This happens as *fly* appears in $DBW_{Methods}$. We design our system that the maximum number of relevant documents in a generated ranked list can be set via a configuration parameter. When this setting is disabled all documents that are marked as relevant are returned to the user.

In our system, it is possible to use entire documents (in PDF format) to search for other documents. In this case, the user has to specify if unstructured or structured text retrieval should be used. The specification influences the generation of the query sets QS. The system processes query sets generated by a document the same way as they are generated by a query. To transform a document into query sets we perform the following steps:

1. First, we extract the article structure and the text content from the PDF. To that end, we use a framework described in [Kla+14].
2. Second, we map section headings to IMRaD sections by keyword detection. We apply a similar approach to map the *Methods* section as we did for the generation of the dataset (see Section 3.1.1). The only difference is that a document will not be rejected if the upper or lower bound cannot be detected. In that case, the *Methods* section is not mapped. This step is only performed for structured text retrieval, as

IMRaD-Type detection is not necessary for unstructured text retrieval.
3. Third, we remove stopwords and stem terms for the whole content of the document.
4. The fourth step differs depending on the usage of unstructured or structured text retrieval:
   - For unstructured text retrieval, we added all terms to QS_{All}.
   - For structured text retrieval, we added the terms to a query set with respect to the IMRaD-Type of the section they occur.

Finally, our system searches for other documents with the usage of the generated query sets. Therefore, we calculate a rank for each document that contains query words in the specified section. In Section 2.2 we discuss several ranking algorithms. All of them calculate rankings based on a query q, and a document d_j. For unstructured text retrieval, the ranking algorithms are used without any transformation as QS_{All} represents the query and DBW_{All} the document.

We change the calculation of the rank for structured text retrieval to take the IMRaD structure into account:

$$sim(d_j, q) = \frac{1}{|\text{IMRaD-TYPES}|} \times \sum_{k \in \text{IMRaD-TYPES}} sim(d_{j,k}, q_k), \qquad (3.3)$$

where IMRaD-TYPES is an array containing all IMRaD-Types. The document d_j is a dictionary that contains all IMRaD bags of words, and $d_{j,k}$ is the bag of words of IMRaD-type k. Furthermore, q is a dictionary containing all query sets, and q_k is the query set of IMRaD-type k. However, for structured text retrieval the mean rank over all IMRaD-types is assigned to be the documents rank.

In our system, *Term Frequency (TF)*, *Term Frequency - Inverse Document Frequency (TF-IDF)*, *BM*25, *Divergence from Randomness (DfR)*, and *Ranked Boolean Retrieval (RBR)* are available as ranking algorithms. A configuration parameter defines which ranking algorithm is used. Additional parameters of the ranking algorithm (e.g., b and K_1 for BM25) are also added to the configuration.

The simplest ranking algorithm is *TF*. It sums the frequencies given in a bag of words, for all terms in a query set. For example, document B is given by:

$$\begin{aligned} DBW_{All} &= \{"love" : 5, "deadlin" : 1, "whoosh" : 2\} \\ DBW_{Introduction} &= \{"love" : 3, "deadlin" : 1\} \\ DBW_{Methods} &= \{"love" : 2, "whoosh" : 2\}. \end{aligned} \tag{3.4}$$

Furthermore, we configure *TF* as ranking algorithm and a query is given by *"love, whoosh"*. Hence, $QS_{All} = \{"love", "whoosh"\}$ and the rank of B is 7 as the counts of DBW_{All} are taken.

Another example would be to use the same document B and *TF* as ranking algorithm, but the query is given by *"love* IN `Introduction`" AND *"whoosh* IN `Methods`". Therefore, the underlying query sets are $QS_{Introduction} = \{"love"\}$ and $QS_{Methods} = \{"whoosh"\}$. We calculate the ranking based on Equation (3.3) since IMRaD structure features are used. The rank of B is 1 as counts are taken from $DBW_{Introduction}$ and $DBW_{Methods}$, and are divided by the number of IMRaD-types. There exist different variants to calculate the term frequency (see Section 2.2.2).

RBR is also a simple ranking algorithm, as it combines zone scores with boolean expressions. The score is applied to the ranking if terms occur in the zone (cf. Section 2.3.1). *TF* and *RBR* do not have any external data sources that have to be provided by the framework. Therefore, the entire calculation of the rank can be done with the document and the query that are passed into the ranking function.

To calculate a rank *TF-IDF* and *BM*25 require the number of documents, in which a term occurs in the document collection. This term occurrence is precalculated for all terms of the *Alphabet* to increase the performance. We store multiple term occurrences for each term as we assume that the distribution of words changes over different IMRaD sections. Hence, we generate them as bags of words with respect to entire documents, and additionally for each IMRaD-type (TO_{All}, $TO_{Introduction}$, $TO_{Background}$, $TO_{Methods}$, $TO_{Results}$, $TO_{Discussion}$). On one hand, TO_{All} expresses the term occurrence for unstructured text retrieval. On the other hand, the bag of words ac-

cording to the IMRaD-type signals the term occurrence for structured text retrieval.

For *TF-IDF* and *BM*25 the term occurrence is related to the calculation of the inverted document frequency. The inverted document frequency represents the importance of a term regarding the whole document collection. There exist different variants to calculate the inverted document frequency (see Section 2.2.2).

Term frequencies over the entire document collection are required to calculate rankings with *DfR*. Therefore, we precalculate them the same way as we do for the term occurrence of *TF-IDF* and *BM*25.

Our proposed ranking algorithms require additional variables such as average document length, and number of documents in the collection. We do not discuss their calculation in detail as they can be computed with negligible effort. All precalculated values are bound by the document collection. Therefore, we recalculate them when the document collection changes.

4 Results and Discussion

In this chapter we discuss the results of our information retrieval model. For our evaluation, we generated a dataset with 821 scientific articles. Each document in the dataset was stored with its logical structure (title, headings, chapters, sections, subsections, subsubsections). Additionally, sections contain information about their IMRaD-Types. Usually a section has only one IMRaD-Type, but some of them have more (e.g., Results and Discussion are assigned the IMRaD-Types Result and Discussion). Furthermore, the articles are linked according their citations. For more information about the creation of our dataset see Section 3.1.

In our experiments we compare 5 common ranking algorithms:

1. First, *Term Frequency* is the simplest approach to rank generated lists. We used *Raw Frequency* of the term frequency variants. There the occurrences of each query term are counted within the document. To read more about different variants of term frequency and inverse document frequency see Section 2.2.2.
2. Second, *TF-IDF* to take inverse document frequency into account. The inverse document frequency represents the importance of a term with respect to the entire document collection. Therefore, *Inverse Frequency* is applied in combination with *Raw Frequency*.
3. Third, *Okapi BM*25 is a baseline ranking algorithm. We use the params as suggested by Robertson et. al [Rob+94]. Therefore, we set $b = 0.75$ and $K_1 = 1$ (cf. Section 2.2.3).
4. Fourth, *Divergence from Randomness* has 2 assumptions that are formalized with equations. In our experiments we used Equation (2.36) and Equation (2.38) for our practical implementation of the assumptions. Furthermore, Equation (2.40) was applied for document length

normalization (cf. Section 2.2.3).

5. Fifth, *Ranked Boolean Retrieval* is a ranking method that combines zone scores with boolean expressions. The score is applied to the ranking if terms occur in the zone. The configuration of the algorithm depends on the experiment (cf. Section 2.3.1).

4.1 Evaluation of Ranking Algorithms

The effectivity of an information retrieval system (IRS) is lean on the underlying ranking function. Therefore, it is important to measure the performance of these ranking functions to make them comparable.

Evaluation of a IRS is based on the set of relevant documents provided by the system. To evaluate a generated set 2 information retrieval basic measures known as *Precision* and *Recall* are used. Both measures do not take any ranking of the relevant documents into account. *Precision* is a measurement of retrieved documents (see Figure 4.1 for sets in the document collection split according their relevance) that are relevant for the user:

$$\begin{aligned}\text{Precision} &= \frac{\#\text{ relevant items retrieved}}{\#\text{ retrieved items}} \\ &= P(\text{relevant}|\text{retrieved}) \\ &= \frac{\text{true positives}}{\text{true positives} + \text{false positives}}.\end{aligned} \tag{4.1}$$

Recall represents the fraction of relevant documents that are received:

$$\begin{aligned}\text{Recall} &= \frac{\#\text{ relevant items retrieved}}{\#\text{ relevant items}} \\ &= P(\text{retrieved}|\text{relevant}) \\ &= \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}.\end{aligned} \tag{4.2}$$

There is a trade-off between the two measures. Therefore, when having a *Recall* of 1 it is possible to have a low *Precision*. This happens as *Recall* always

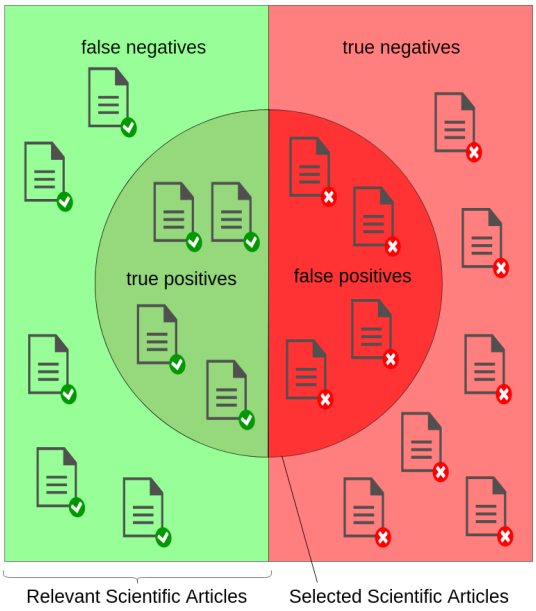


Figure 4.1: **Document collection split according their relevance.** During the evaluation of a ranking algorithm the document collection is split into 4 subsets. First, the *true positive* documents retrieved as relevant for the user, which are actually relevant. Second, the *false positive* documents retrieved as relevant, but are not relevant for the user. Third, the *true negatives* documents were correctly received as not relevant. Forth, the *false negatives* documents were incorrectly received as not relevant. The *true positive* set, and *true negative* set should be as large as possible as it is directly related to the effectivity of the ranking algorithm.

increasing until all relevant documents are retrieved, but new received documents can be *false positives*. Hence, the *Precision* decreases, and *Recall* stays the same.

There exist many different metrics to evaluate generated sets with ranking (ordered). Average Precision is one of the most commonly used evaluation techniques. It is defined as the average of all precision values after a new relevant document is observed:

$$AP_i = \frac{1}{|R_i|} \sum_{k=1}^{|R_i|} P(R_i[k]), \tag{4.3}$$

where R_i is the set of relevant documents with respect to query q_i. $R_i[k]$

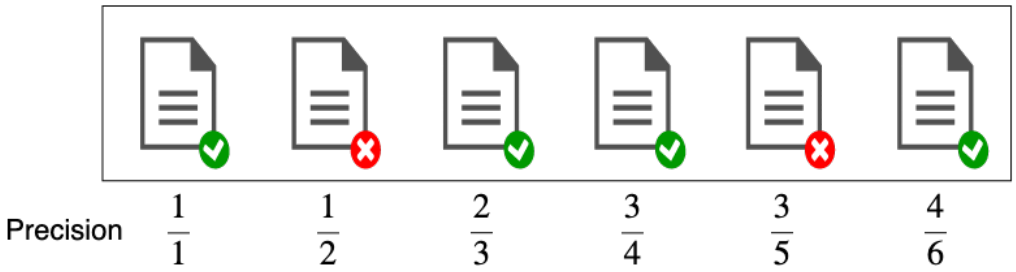


Figure 4.2: **Example for Mean Average Precision of a single query.** The Precision values are calculated according the retrieved set of relevant documents. Documents with a green hook are *true positives* (TP), and documents with a red cross are *false positives* (FP). The precision values of FP documents are ignored for these documents, as they are not counted. In the example $R_1 = \{\text{TP, FP, TP, TP, FP, TP}\}$, and therefore $|R_1| = 6$. Inserting these values into the Mean Average Precision formula of a single query (see Equation (4.3)) results in $MAP_1 = 1/6 \times (1 + (2/3) + (3/4) + (4/6)) = 0.513\overline{8}$.

represents the reference of the kth document in R_i, and $P(R_i[k])$ is the *Precision* of the document (see Equation (4.1)). If $R_i[k]$ belongs to a *false positive* document $P(R_i[k]) = 0$. Furthermore, the Mean Average Precision over a set of queries is defined as:

$$MAP = \frac{1}{N_q} \sum_{i=1}^{|N_q|} AP_i, \tag{4.4}$$

where N_q is the total number of queries.

Mean Average Precision is widely used as it is simple, easy to implement, versatile, and stable. Therefore, we apply it to compare the generated ranked lists of our proposed ranking algorithms (for an example see Figure 4.2).

4.2 Leveraging IMRaD Structure Features

In our first experiments we compare unstructured text retrieval with structured text retrieval. Therefore, each document in the document collection

contains multiple bag of words. In a bag of words, each index term is represented as a pair of term and term frequency (see Section 3.2 for the definition of the document). For structured text retrieval, we focus on the underlying IMRaD structure.

First, we evaluate explicit search for which query terms have to be formulated explicitly. For example, given a query q_1 ="*network* IN `Methods`" the system leverages different bag of words to search for the term. For structured text retrieval the Methods-bag of words is used as it was specified in the query. In our system IN-statements of queries are ignored for unstructured text retrieval, as all terms are searched in the bag of words containing all index terms of a document.

Second, we discuss implicit search were we use entire scientific articles to search for other articles. For structured text retrieval, we split the input paper according to its IMRaD-structure. We generate a query that is similar to the query of the explicit variant with the extracted terms. For unstructured text retrieval the same process is used to create the query, however IN-statements are ignored here as well.

4.2.1 Explicit Search using Word N-Grams

Searching with information provided by the user is denoted as explicit search. For our information retrieval model this is done by formalizing user queries. These queries can be seen as a sequence of keywords.

Our generated dataset consists of scientific articles and links between them. We extracted citations in order to get queries that are used for testing. We base our model on the assumption that citations describe the content of referenced articles. Therefore, a referenced article is represented by the citation.

We store each citation as a set of terms. An important addition is that the order of terms within the set is not lost. Additionally, we assume that

a typical user searches by combining keywords. Therefore, we removed stopwords from the set, but terms were not stemmed. For each citation set we store information about the cited article and the IMRaD-Types of the section. For example, *"The authors of [1] present a comprehensive system for the structure extraction of PDF books, which is used within a commercial e-book software."* is in the *Introduction* of scientific article A. Furthermore, *"[1]"* is the reference to the cited scientific article B. The resulting citation set looks as follows: $cs_{A1} = \{$*"authors"*, *"present"*, *"comprehensive"*, *"system"*, *"structure"*, *"extraction"*, *"PDF"*, *"books"*, *"commercial"*, *"e-book"*, *"software"*$\}$, and additionally the reference to article B, and the IMRaD-Type *Introduction* is stored for cs_{A1}.

Queries are produced as N-Grams using the citation sets. This means the citation sets are split into subsets of size N. Furthermore, the order of the terms is also important for these subsets. For example, cs_{A1} is split into 5-Grams. The 7 resulting subsets are:

$$NG_{cs_{A1},1} = \{\text{"authors"}, \text{"present"}, \text{"comprehensive"}, \text{"system"}, \text{"structure"}\}$$
$$NG_{cs_{A1},2} = \{\text{"present"}, \text{"comprehensive"}, \text{"system"}, \text{"structure"}, \text{"extraction"}\}$$
$$NG_{cs_{A1},3} = \{\text{"comprehensive"}, \text{"system"}, \text{"structure"}, \text{"extraction"}, \text{"PDF"}\}$$
$$NG_{cs_{A1},4} = \{\text{"system"}, \text{"structure"}, \text{"extraction"}, \text{"PDF"}, \text{"books"}\}$$
$$NG_{cs_{A1},5} = \{\text{"structure"}, \text{"extraction"}, \text{"PDF"}, \text{"books"}, \text{"commercial"}, \}$$
$$NG_{cs_{A1},6} = \{\text{"extraction"}, \text{"PDF"}, \text{"books"}, \text{"commercial"}, \text{"e-book"}\}$$
$$NG_{cs_{A1},7} = \{\text{"PDF"}, \text{"books"}, \text{"commercial"}, \text{"e-book"}, \text{"software"}\}$$

In general, the number of subsets that can be generated from a citation set cs is defined as:

$$l = len(cs) - N + 1, \tag{4.5}$$

where $len(cs)$ is the number of terms in cs.

The last step of our query generation process is to transform query sets and IMRaD-type into our used query structure. For example, the resulting query compiled by $NG_{cs_{A1},1}$ looks as follows: $q_1 =$*"authors*, *present*, *comprehensive*,

system, *structure* IN `Introduction`". In our system IN-statements of queries are ignored if IMRaD chapter features are disabled.

Mean Average Precision (see Section 4.1) is used to evaluate our proposed ranking algorithms. Therefore, we pass the query into our information retrieval system. The Average Precision depends on the position of the cited article in the generated ranked list.

Table 4.1 highlights the performance of our proposed ranking algorithms. We evaluate different query lengths (the number of terms in the query), where query lengths vary in the range between 2 to 14. Furthermore, we compare the algorithms with respect to the usage of IMRaD chapter features, where disabled features denote unstructured text retrieval and enabled features represent structured text retrieval. We highlight the best results with enabled features in violet and with disabled features in blue for every algorithm.

All 5 algorithms achieve better performance results without IMRaD chapter features. Therefore, the ranking is generated with respect to their achieved accuracy when IMRaD chapter features are disabled. The best performing algorithm is *TF-IDF*. It achieves a MAP of 0.2199 with disabled features, where the query consists of 11 terms. In contrast, the same algorithm achieves a MAP of 0.1642 with enabled features and a query length of 12. Therefore, unstructured text retrieval is 5.57% better than structured text retrieval for this configuration.

The second best algorithm is *TF*. It achieves an accuracy of 0.1966 with disabled features and a query length of 11. In contrast, the same algorithm achieves a MAP of 0.1293 with enabled features and a query length of 12. Therefore, unstructured text retrieval is 6.73% better than structured text retrieval for this configuration. In comparison to *TF-IDF* the achieved accuracy is 2.33% lower with disabled features, and 3.49% lower with enabled features. The best performing query lengths are the same as for *TF-IDF*.

The third best algorithm is *Ranked Boolean Retrieval*. Zone scores *zs* have to be defined to configure the algorithm (cf. Section 2.3.1). For the experiments only zones that contain text areas are set to a value greater zero. This is done as all citations were extracted from text areas.

# Terms in Query	# Queries	Using IMRaD Chapter Features	Term Frequency	TF-IDF	Ranked Boolean Retrieval	Okapi BM25	Divergence from Randomness
2	8770	No	0.0882	0.1128	0.1035	0.0442	**0.0498**
		Yes	0.0696	0.0897	0.0638	0.045	**0.0379**
3	7070	No	0.1038	0.1382	0.1198	0.064	0.046
		Yes	0.0785	0.1042	0.0713	0.0628	0.0323
4	5589	No	0.1197	0.1547	0.1336	0.0739	0.0448
		Yes	0.0894	0.1167	0.0787	0.0734	0.031
5	4347	No	0.1317	0.1689	0.1479	0.0794	0.0416
		Yes	0.0993	0.1262	0.0844	0.0791	0.0317
6	3336	No	0.1469	0.1766	0.1603	0.0804	0.0396
		Yes	0.1042	0.1319	0.0918	0.0819	0.0311
7	2550	No	0.1588	0.1857	0.167	0.0817	0.0404
		Yes	0.1085	0.1375	0.0961	0.0842	0.0312
8	1911	No	0.1712	0.1957	0.1718	0.0818	0.0449
		Yes	0.1158	0.1441	0.0988	0.0916	0.0303
9	1402	No	0.1804	0.2074	0.1757	0.0879	0.0456
		Yes	0.1213	0.1496	**0.1015**	0.0969	0.0301
10	1051	No	0.1847	0.2153	0.1851	0.0952	0.0466
		Yes	0.1235	0.1555	0.1005	0.099	0.0304
11	787	No	**0.1966**	**0.2199**	**0.1921**	0.1075	0.0439
		Yes	0.1217	0.1589	0.0978	0.1034	0.0296
12	606	No	0.1923	0.2159	0.1851	0.1102	0.0397
		Yes	**0.1293**	**0.1642**	0.0974	0.1043	0.0296
13	470	No	0.1846	0.205	0.1712	0.1192	0.0319
		Yes	0.1247	0.1637	0.0958	**0.1058**	0.0295
14	362	No	0.1634	0.1919	0.1558	**0.1207**	0.0221
		Yes	0.1183	0.1589	0.0964	0.1008	0.0311

Table 4.1: **Ranking results of our proposed ranking algorithms using explicit search.** Mean Average Precision is used to evaluate our proposed ranking algorithms. Furthermore, we compare them with and without IMRaD chapter features. Without IMRaD chapter features the entire document is used to search for query terms (unstructured). When IMRaD chapter features are enabled query terms are searched only in specified sections. We evaluate query lengths (the number of terms in the query) in the range between 2 to 14.

In our model, these are text areas of sections, subsections, and subsubsections. Therefore, the constant zone scores are $zs_{\text{Section}} = 0.34$, $zs_{\text{Subsection}} = 0.33$, and $zs_{\text{Subsubsection}} = 0.33$.

Ranked Boolean Retrieval achieves an accuracy of 0.1921 with disabled features and a query length of 11. Hence, in the context of unstructured text retrieval the best performing query length is the same as for *TF-IDF* and *TF*. In contrast, the same algorithm achieves a MAP of 0.1015 with enabled features and a query length of 9. Therefore, unstructured text retrieval is 9.06% better than structured text retrieval for this configuration. In comparison to *TF-IDF* the achieved accuracy is 2.78% lower with disabled features and 6.27% lower with enabled features. Furthermore, in comparison to *TF* the accuracy is 0.45% lower with disabled features, and 2.78% lower with enabled features.

The forth best algorithm is *Okapi BM*25. It achieves an accuracy of 0.1207 with disabled features. The associated best performing query length is 14, which was the upper bound of the query lengths for our experiments. The upper bound comes from the number of queries that can be generated from our dataset. In contrast, the same algorithm achieves a MAP of 0.1058 with enabled features and a query length of 13. Therefore, unstructured text retrieval is 1.49% better than structured text retrieval for this configuration. In comparison to *TF-IDF* the achieved accuracy is 9.92% lower with disabled features and 5.84% lower with enabled features. Furthermore, in comparison to *Ranked Boolean Retrieval* the accuracy is 7.14% lower with disabled features, and 0.43% better with enabled features.

The fifth best algorithm is *Divergence from Randomness*. It achieves an accuracy of 0.0498 with disabled features, and an accuracy of 0.0379 with enabled features. Therefore, unstructured text retrieval is 1.19% better than structured text retrieval for this configuration. The best performing query length is 2 with disabled and enabled features. In comparison to *TF-IDF* the achieved accuracy is 17.01% lower with disabled features, and 12.63% lower with enabled features. Furthermore, in comparison to *Okapi BM*25 the accuracy is 7.09% lower with disabled features, and 6.79% lower with enabled features.

An interpretation of the results captures that it is not necessary to have the overhead of IMRaD chapter features when only a few keywords are used to search for scientific articles. This happens as the keywords define content that should occur anywhere in the articles. Additional constraints that restrict where terms can appear are rather obstructive as they tend to prevent the retrieval of relevant articles.

Furthermore, we interpret the results of each ranking algorithm. *TF-IDF* outperforms the other algorithms, which was unsurprisingly as good results are often observed for this algorithm. During our research we found *Okapi BM*25 with similar results. However, to achieve a good performance an extensive parameter search is required. We use *Okapi BM*25 with recommended parameters, which results in worse accuracy. In comparison to the other ranking algorithms *Divergence from Randomness* is always outperformed by all other algorithms. This bad performance is probably related to our dataset. The good performance of *Ranked Boolean Retrieval* was surprising. We can probably attribute this to the zone scores, as they are hiding irrelevant areas, and mark important article areas.

4.2.2 Implicit Search using Scientific Articles

In traditional information retrieval models information is available that was not explicitly provided by the user. Leveraging this type of information in addition is denoted implicit search. In our model, we use explicit and implicit information of scientific articles to search for other articles. This approach is referred to as *more like this*.

For our experiment we transformed scientific articles into user queries. Therefore, queries contain information about index terms and the IMRaD sections they occur. For example, q_1 =”*structure*, *present* IN `Introduction` AND *system*, *structure* IN `Methods`” is a query that can be executed by our system. IN-statements are used to define where the terms occur, and AND-statements are used to combine subqueries. In our system IN-statements of queries are ignored if IMRaD chapter features are disabled.

We use a generated dataset with 821 scientific articles to evaluate our proposed ranking algorithms. Therefore, we assume that the content of a scientific article has to be similar to its cited articles. As a result, the Mean Average Precision (see Section 4.1) is calculated with respect to the position of the cited articles in the generated ranked list.

Table 4.2 highlights the performance of our proposed ranking algorithms. *TF*, *TF-IDF*, *Ranked Boolean Retrieval*, and *Okapi BM*25 obtain higher results with the usage of IMRaD chapter features. We find no difference in performance of *Divergence from Randomness* when leveraging IMRaD chapter features.

The ranking of the proposed ranking algorithms is generated with respect to their achieved accuracy when IMRaD chapter features are enabled. The best performing algorithm is *TF-IDF*. It achieves an accuracy of 0.1613 with enabled features, and 0.1163 with disabled features. Therefore, structured text retrieval is 4.5% better than unstructured text retrieval for this configuration.

The second best performing algorithm is *TF*. It achieves an accuracy of 0.1463 with enabled features, and 0.1186 with disabled features. Therefore, structured text retrieval is 2.77% better than unstructured text retrieval for this configuration. In comparison to *TF-IDF* the achieved accuracy is 1.5% lower for enabled features, and 0.2% better for disabled features.

Using IMRaD Chapter Features	**Term Frequency**	**TF-IDF**	**Ranked Boolean Retrieval**	**Okapi BM25**	**Divergence from Randomness**
No	0.1186	0.1163	0.0466	0.0554	0.0137
Yes	0.1463	0.1613	0.0506	0.0882	0.0137

Table 4.2: **Ranking results of the used weighting schemes using scientific articles.** We compare our proposed ranking algorithms with respect to the underlying structure. Therefore, we use scientific articles unstructured and structured to search for other scientific articles. For structured articles, we focus on the underlying IMRaD structure. Mean average precision was used to evaluate the results.

The third best performing algorithm is *Okapi BM*25. It achieves an accuracy of 0.0882 with enabled features, and 0.0554 with disabled features. Therefore, structured text retrieval is 3.28% better than unstructured text retrieval for this configuration. In comparison to *TF-IDF* the achieved accuracy is 7.31% lower for enabled features, and 6.09% lower for disabled features. Furthermore, in comparison to *TF* the accuracy is 5.81% lower for enabled features, and 6.32% lower for disabled features.

The forth best performing algorithm is *Ranked Boolean Retrieval*. Zone scores *zs* have to be defined to configure the algorithm. For our experiments we determine the best configuration using parameter search. The best performing configuration is as follows:

$$\begin{aligned} zs_{\text{Title}} &= 0.2 \\ zs_{\text{Section Title}} &= 0.3 \\ zs_{\text{Section Text}} &= 0.2 \\ zs_{\text{Subsection Title}} &= 0.18 \\ zs_{\text{Subsection Text}} &= 0.05 \\ zs_{\text{Subsubsection Title}} &= 0.05 \\ zs_{\text{Subsubsection Text}} &= 0.02, \end{aligned}$$

where "Title" defines the header of the document component, and "Text" the text area of the of the document component. Fur further information about structured text retrieval, document components, and *Ranked Boolean Retrieval* see Section 2.3.

Ranked Boolean Retrieval achieves an accuracy of 0.0506 with enabled features, and 0.0466 with disabled features. Therefore, structured text retrieval is 0.4% better than unstructured text retrieval for this configuration. In comparison to *TF-IDF* the achieved accuracy is 11.07% lower for enabled features, and 6.97% lower for disabled features. Furthermore, in comparison to *Okapi BM*25 the accuracy is 3.76% lower for enabled features, and 0.88% lower for disabled features.

The fifth best performing algorithm is *Divergence from Randomness*. It achieves an accuracy of 0.0137 with enabled and disabled features. In comparison

to *TF-IDF* the achieved accuracy is 14.76% lower for enabled features, and 10.26% lower for disabled features. Furthermore, in comparison to *Ranked Boolean Retrieval* the accuracy is 3.69% lower for enabled features, and 3.29% lower for disabled features.

When articles are used to search for other articles they can be seen as large and precise queries. IMRaD chapter features are leveraged to define these queries. These are helpful as they introduce constraints that describe the expected content of articles.

As expected, we find that TF-IDF achieves the best accuracy. We used *Okapi BM25* only with recommended parameters. An exhaustive parameter search would be necessary to fit our generated dataset. In comparison to the other ranking algorithms *Divergence from Randomness* was always outperformed by all other algorithms. This bad performance is probably related to our dataset. In this experiments it can be seen that the zone scores of *Ranked Boolean Retrieval* are not enough to fit the complexity of large queries.

4.3 Chapter Based Search

When searching with entire scientific articles (see Section 4.2.2) we obtain better results with the usage of IMRaD chapter features compared to the

Section	Introduction	Background	Methods	Results	Discussion
Introduction	0.1242	0.1226	0.1095	0.1092	0.1049
Background	0.1454	0.1249	0.1331	0.1255	0.1106
Methods	0.0947	0.0857	0.1017	0.0897	0.0668
Results	0.0877	0.0783	0.0815	0.0783	0.0631
Discussion	0.1188	0.1078	0.0957	0.0914	0.084

Table 4.3: **Chapter based Search using TF-IDF.** Keywords of a single chapter are used to search in individual chapters of other articles. These input chapters are represented as rows, and the search chapters are represented as columns. Mean average precision was used to evaluate the results of the TF-IDF ranking algorithm.

usage without IMRaD chapter features. Therefore, we focus on structured text retrieval and the influence of single chapters on the search result. The idea is that keywords of a chapter can be used to search in other chapters. For example, the Methods section of one paper can be referenced in the Related Work of another paper.

We use the same dataset and evaluation process as for our implicit search evaluation (see Section 4.2.2). In our dataset the background chapter is available in addition to the IMRaD chapters. The evaluation was done for *TF-IDF* and *Okapi BM*25 as they are state-of-the-art ranking algorithms.

The chapter of the article that is used to search for other articles is defined as input chapter. Furthermore, the chapter that is searched in the articles of the collection is defined as search chapter. For example, the introduction of an article is used to search in the discussion of articles in the collection. In this example the introduction is the input chapter and the discussion is the search chapter.

Table 4.3 highlights the performance results of *TF-IDF*. When *Introduction*, *Background*, *Results*, or *Discussion* is the input chapter the best results are obtained when *Introduction* is the search chapter. Furthermore, when *Methods* is the input chapter then the best performance is given if *Methods* is also the search chapter. The highest accuracy is 0.1454, where *Background* is the input chapter, and *Introduction* the search chapter. When summing up accuracies

Section	Introduction	Background	Methods	Results	Discussion
Introduction	0.0884	0.0686	0.0631	0.061	0.0708
Background	0.0909	0.0715	0.076	0.0618	0.0751
Methods	0.0565	0.0417	0.0593	0.0379	0.0403
Results	0.0438	0.0426	0.0433	0.0461	0.0443
Discussion	0.0799	0.0682	0.0587	0.0595	0.0616

Table 4.4: **Chapter based Search using Okapi BM**25**.** Keywords of a single chapter are used to search in individual chapters of other articles. These input chapters are represented as rows, and the search chapters are represented as columns. Mean average precision was used to evaluate the results of the Okapi BM25 ranking algorithm.

Background is the best performing input chapter, and *Introduction* the best performing search chapter.

Table 4.4 highlights the performance results of *Okapi BM*25. When *Introduction*, *Background*, or *Discussion* is the input chapter the best results are obtained when *Introduction* is the search chapter. For *Methods* and *Results* the best results are obtained when they are used as input chapter as well as search chapter. The highest accuracy is 0.0909, where *Background* is the input chapter, and *Introduction* the search chapter. When summing up accuracies *Background* is the best performing input chapter, and *Introduction* the best performing search chapter.

TF-IDF achieves better results than *Okapi BM*25 when comparing the performance results. Both ranking algorithms have *Background* as their best performing input chapter, and *Introduction* as their best performing search chapter. When comparing their highest accuracy *TF-IDF* is 5.45% better than *Okapi BM*25. This reflects also approximately the performance of the other input chapters and search chapters.

The accuracy of implicit search with enabled IMRaD chapter Features is 1.59% better than the highest accuracy of search with single chapters. One interesting point is that queries for single chapters are one fifth of queries of implicit search, but have almost the same performance. We used *Okapi BM*25 only with recommended parameters. A more precised parameter search would be necessary to suite our generated dataset.

5 Conclusion

5.1 Summary

Search engines help to reduce the time required to find a piece of information, and minimize the number of information sources that need to be searched. We focus on scientific literature search where search engines help to find scientific articles.

An advantage of scientific articles is that they share a common structure to increase the readability. This structure is known is IMRaD (Introduction, Method, Results and Discussion). In our work we tackle the problem whether it is possible to improve the search result quality when searching for scientific works by leveraging IMRaD structure information. Specifically,

(a) Does the search result improve for explicit search using queries?
(b) Does the search result improve for implicit search using complete scientific papers?
(c) Does the search result improve if only a single chapter of the scientific paper is used for searching?

In the related work section we describe the definition of an information retrieval model. Afterwards, we discuss the 3 classical models for unstructured text retrieval. First, the boolean model where documents and queries are represented as sets. Terms are combined with boolean operators to formulate queries. Second, the vector model where documents and queries are represented as a vector in a t-dimensional space. Third, the probabilistic model where documents and queries are represented based on probability

theory. Specifically, by estimating the probability of a term appearing in a relevant document.

Additionally, we describe extensions of the vector-, and the probabilistic model. First, the TF-IDF model which is based on the vector space model, and is one of the most popular weighting schemes in information retrieval. Second, the BM25 model which is based on the probabilistic model. It is the result of several experiments by Robertson et. al [Rob+92; Rob+93; Rob+94]. Third, the Divergence from Randomness model was introduced by Amati and Rijsbergen [AR02] and is a probabilistic model that exhibits characteristics of a language model as well.

Next, we discuss techniques of structured text retrieval. We focus specifically on 5 ranking strategies known as contextualization, propagation, aggregation, merging, and zone scores. The model based on zone scores is proposed by Manning et al. [MRS08], and is also known as Ranked Boolean Retrieval.

In the last part of the related work we focus on the IMRaD structure in scientific articles. Sollaci and Pereira [SP04] describe in their work that the IMRaD structure began to be adopted in the 1940s, and became the standard format for scientific articles in the 1970s. Furthermore, we discuss IMRaD structure distributions as proposed by Bertin et al. [Ber+13], and how IMRaD structure can be leveraged in information retrieval systems.

In the methods section we started with the description of our dataset. Our dataset is composed of 821 scientific articles. We added additional information such as IMRaD mappings, and links between the articles based on citations. Furthermore, we defined our database schema with respect to the article structure and analyze the citation network.

Afterwards, we describe our introduced system and the underlying model. We design our system to compare various common ranking algorithms. Hence, the ranking algorithms require to be easily interchangeable. In addition, our model is designed to work with unstructured as well as structured data. This is reflected by the query language.

In the results and discussion section we describe a measure for the performance of ranking algorithms, which is required to compare our proposed ranking algorithms.

Finally, we list our study results, and interpret them. We divide our interpretation into 3 parts, which is based on our research questions:

(a) First, we compare unstructured text retrieval with structured text retrieval where we simulate explicitly formulated queries with the usage of a generated test set. We find that it is not necessary to have the overhead of IMRaD chapter features when only a few keywords are used to search for scientific articles. This happens as the keywords define the content that should occur anywhere in the articles. Additional constraints that restrict where terms can appear are rather obstructive, as they tend to prevent the retrieval of relevant articles.
(b) Second, we discuss implicit search were we use entire scientific articles to search for other articles. When articles are used to search they can be seen as large and precise queries. IMRaD chapter features are leveraged to define these queries. We find that these are helpful as they introduce constraints that describe the expected content of articles.
(c) Third, we focus on structured text retrieval and the influence of single chapters on the search result. Searching with all IMRaD chapter of a document performs marginally better than searching with single IMRaD chapters. We obtain the best accuracy when we use the Background section to search in Introduction sections. One interesting point we found out is that queries for single chapters are one fifth of queries of implicit search, but have almost the same performance.

5.2 Overarching results

In this chapter we briefly summarize the results we obtained in Chapter 4. Afterward, we discuss these results against each other to present them in a larger context.

For our first experiments, we compare unstructured text retrieval with structured text retrieval where we simulate explicitly formulated queries with the usage of a generated test set. The accuracies of the 3 best performing algorithms are between 0.1921 and 0.2199 for unstructured text retrieval, and between 0.1015 and 0.1642 for IMRaD structured text retrieval. We only take the best 3 results as others have less significance for the best performing results. Based on the results we find that unstructured text retrieval outperforms IMRaD structured text retrieval. This happens as terms in a short query define content that should occur anywhere in the articles. Additional constraints that restrict where terms can appear are rather obstructive as they tend to prevent the retrieval of relevant articles.

For our second experiments, we compare unstructured text retrieval with structured text retrieval according entire documents. The accuracies of the 3 best performing algorithms are between 0.0554 and 0.1186 for unstructured text retrieval, and between 0.0882 and 0.1613 for IMRaD structured text retrieval. Based on the results we find that IMRaD structured text retrieval outperforms unstructured text retrieval. This happens as documents can be seen as large and precise queries when searching for other documents. IMRaD chapter features are leveraged to define these queries. These are helpful as they introduce constraints that describe the expected content of articles.

The comparison of the accuracies in the 2 experiments may lead to the assumption that searches with queries result with more relevant papers than searches with documents. This conclusion is misleading as the 2 experiments cover different requirements of a user. On one hand, when a short query is used to search for other articles a user expects a lot of documents due to unknown expectations. On the other hand, when a user searches with the usage of a document then other documents that are related to it are expected.

The accuracies of *Term Frequency*, *TF-IDF*, and *BM*25 are similarly high for the first and the second experiment when using IMRaD structured text retrieval. Only *Ranked Boolean Retrieval* lost a lot of its accuracy. We assume that this is due to the fact that the *Ranked Boolean Retrieval* already provides a structure for the ranking, and the 2 structures influence each

other negatively. All 4 algorithms show a similar decrease in accuracy for unstructured text retrieval. *Divergence from Randomness* always performs bad, and therefore the results are hard to interpret.

For our third experiments we focus on structured text retrieval and the influence of single chapters on the search result. We used *TF-IDF* and *BM*25 as ranking algorithms for the experiment. The highest accuracy was obtained when the Background section is used to search in Introduction sections, where *TF-IDF* obtains an accuracy of 0.1454 and *BM*25 obtains an accuracy of 0.0909.

In comparison with our first experiment the accuracies are lower, but also they reflect different requirements. Furthermore, in comparison to our second experiment the performance is 1.59% worse. Therefore, searching with all IMRaD chapter of a document performs marginally better than searching with single IMRaD chapters.

During the development of our system, we focus on 2 applications for information retrieval system in the field of scientific literature research. We find that the usage of IMRaD structure features depends on the use-case. The first application is based on breadth-first search and covers the initial search process. During this initial search a user wants to gets a first overview of a topic, and obtain many articles based on a set of given keywords. Therefore, the system has to provide these articles in response to simple structured queries. We discussed for our first experiments that additional constraints that restrict where terms can appear are rather obstructive as they tend to prevent the retrieval of relevant articles. IMRaD structure features are not necessary when a system is used for the initial search.

The second application is based on depth-first search and covers the specific search of literature. More precise queries are necessary when a user is finished with the initial search, and specifically wants to find additional literature based on the articles obtained in the first step. Therefore, the system has to leverage additional functionality to handle more complex queries. In our second experiment we observe that IMRaD structure features are helpful as they introduce constraints that describe the expected content of articles. As a result, IMRaD structure features improve the results for the

specific literature search.

5.3 Future Work

The implemented information retrieval system, and the generated dataset provide various opportunities for possible extensions and experiments. The following examples should give ideas of what are possible paths for future research.

For IMRaD structured text retrieval we calculate the rank based on the mean over all IMRaD chapters. As discussed in Section 2.3.1 there exist other ranking strategy as well. They can be added as extensions to our system. Afterwards, different combinations of ranking algorithms and ranking strategies can be evaluated.

Another possible improvement would be a parameter search for BM25. Robertson et. al [Rob+94] propose in their work a general configuration that suites many cases. We find that BM25 always performs mediocre with this configuration. With an extensive parameter search the performance may increase, and BM25 could catch up with the best performing algorithms.

We use a dataset that consists of 821 scientific articles. One improvement would be to add additional articles. However, we assume for our query evaluation that citations describe the content of referenced articles. Afterward, queries were auto-generated based on this assumption. These auto-generated queries require to be verified as the assumption may not hold for all of them. All queries where the assumption does not hold needs to be removed from the test set. Another possibility is to create document collections from journals of the Public Library of Science (PLOS). These journals consist of thousands of articles, which increases the expressiveness of experiments.

An additional extension for our system would be to calculate similarities based on article clusters. The idea is that the user provides a set of articles,

which are clustered with the articles in the document collection. These clusters are generated based on different properties (e.g., based on the Introduction of the articles). Afterwards a ranked list is generated based on the distances in the cluster.

Bibliography

[AA20] Ibrar Ahmed and Muhammad Tanvir Afzal. "A Systematic Approach to Map the Research Articles' Sections to IMRAD." In: *IEEE Access* 8 (2020), pp. 129359–129371. URL: http://dblp.uni-trier.de/db/journals/access/access8.html#AhmedA20a (cit. on p. 36).

[ABD06] Eugene Agichtein, Eric Brill, and Susan Dumais. "Improving web search ranking by incorporating user behavior information." In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '06. Seattle, Washington, USA: ACM, 2006, pp. 19–26. ISBN: 1-59593-369-7 (cit. on p. 2).

[AJK05] Paavo Arvola, Marko Junkkari, and Jaana Kekäläinen. "Generalized contextualization method for XML information retrieval." In: *CIKM*. Ed. by Otthein Herzog et al. ACM, 2005, pp. 20–27. ISBN: 1-59593-140-6 (cit. on p. 32).

[Ali+06] Wouter Alink et al. "XIRAF - XML-based indexing and querying for digital forensics." In: *Digital Investigation* 3.Supplement-1 (2006), pp. 50–58 (cit. on p. 31).

[AR02] Gianni Amati and C. J. van Rijsbergen. "Probabilistic models of information retrieval based on measuring the divergence from randomness." In: *ACM Trans. Inf. Syst.* 20.4 (2002), pp. 357–389 (cit. on pp. 27, 28, 67).

[Ber+13] M. Bertin et al. "The Distribution of References in Scientific Papers: an Analysis of the IMRaD Structure." In: *In proceeding of: 14th International Society of Scientometrics and Informetrics Conference* 1 (2013), pp. 591–603 (cit. on pp. 35, 67).

[BP98] Sergey Brin and Lawrence Page. "The Anatomy of a Large-scale Hypertextual Web Search Engine." In: *Comput. Netw. ISDN Syst.* 30.1-7 (1998), pp. 107–117. ISSN: 0169-7552 (cit. on p. 9).

[Bur92a] Forbes J. Burkowski. "An algebra for hierarchically organized text-dominated databases." In: *Information Processing and Management* 28.3 (1992), pp. 333–348. ISSN: 0306-4573. DOI: https://doi.org/10.1016/0306-4573(92)90079-F (cit. on p. 31).

[Bur92b] Forbes J. Burkowski. "Retrieval Activities in a Database Consisting of Heterogeneous Collections of Structured Text." In: *SIGIR*. Ed. by Nicholas J. Belkin, Peter Ingwersen, and Annelise Mark Pejtersen. ACM, 1992, pp. 112–125. ISBN: 0-89791-523-2 (cit. on p. 31).

[CMF96] Yves Chiaramella, P. Mulhelm, and Franck Fourel. "A Model for Multimedia Information Retrieval." In: (Nov. 1996) (cit. on p. 32).

[Day89] Robert A. Day. "The Origins of the Scientific Paper: The IMRAD Format." In: *American Medical Writers Association Journal* 4.2 (1989). Archived from the original on September 27, 2011. Retrieved 2011-06-17., pp. 16–18 (cit. on p. 35).

[Gev06] Shlomo Geva. "GPX - Gardens Point XML IR at INEX 2005." In: *Advances in XML Information Retrieval and Evaluation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 240–253 (cit. on p. 32).

[Gud+97] V.N. Gudivada et al. "Information retrieval on the World Wide Web." In: *Internet Computing, IEEE* 1.5 (1997), pp. 58–68. ISSN: 1089-7801 (cit. on p. 10).

[Jon72] Karen Spärck Jones. "A statistical interpretation of term specificity and its application in retrieval." In: *Journal of Documentation* 28.1 (1972) (cit. on p. 16).

[Kas11] M. Kas. *Structures and Statistics of Citation Networks*. Master's thesis, Carnegie Mellon University Pittsburgh, PA. 2011 (cit. on p. 41).

[Kla+14] Stefan Klampfl et al. "Unsupervised document structure analysis of digital scientific articles." In: *Int. J. on Digital Libraries* 14.3-4 (2014), pp. 83–99 (cit. on pp. 37, 47).

[Kle99] Jon M. Kleinberg. "Authoritative Sources in a Hyperlinked Environment." In: *J. ACM* 46.5 (1999), pp. 604–632. ISSN: 0004-5411 (cit. on p. 9).

[Luh57] H. P. Luhn. "A statistical approach to mechanized encoding and searching of literary information." In: *IBM Journal of Research and Development* 1 (1957), pp. 309–317 (cit. on p. 15).

[Mea85] Arthur J. Meadows. "The scientific paper as an archaeological artefact." In: *Journal of Information Science* 11.1 (1985), pp. 27–30 (cit. on p. 35).

[Mel09] Massimo Melucci. "Boolean Model." In: *Encyclopedia of Database Systems*. Boston, MA: Springer US, 2009, pp. 260–260. ISBN: 978-0-387-39940-9 (cit. on p. 10).

[MM04] Yosi Mass and Matan Mandelbrod. "Component Ranking and Automatic Query Refinement for XML Retrieval." In: *INEX*. Ed. by Norbert Fuhr et al. Vol. 3493. Lecture Notes in Computer Science. Springer, 2004, pp. 73–84. ISBN: 3-540-26166-4 (cit. on pp. 32, 33).

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008 (cit. on pp. 20, 33, 67).

[NB95] Gonzalo Navarro and Ricardo Baeza-Yates. "A Language for Queries on Structure and Contents of Textual Databases." In: *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '95. Seattle, Washington, USA: Association for Computing Machinery, 1995, pp. 93–101. ISBN: 0897917146. DOI: 10.1145/215206.215336 (cit. on p. 31).

[NB97] Gonzalo Navarro and Ricardo Baeza-Yates. "Proximal Nodes: A Model to Query Document Databases by Content and Structure." In: *ACM Trans. Inf. Syst.* 15 (Oct. 1997), pp. 400–435. DOI: 10.1145/263479.263482 (cit. on p. 31).

[RB99] Berthier Ribeiro-Neto and Ricardo Baeza-Yates. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999 (cit. on pp. 7, 11, 13, 14, 17, 20, 22).

[RJ76] S. E. Robertson and Sparck K. Jones. "Relevance weighting of search terms." In: *Journal of the American Society for Information Science* 27.3 (1976), pp. 129–146. DOI: 10.1002/asi.4630270302 (cit. on pp. 21, 22).

[Rob+92] Stephen E. Robertson et al. "Okapi at TREC." In: *TREC*. Ed. by Donna K. Harman. Vol. 500-207. NIST Special Publication. National Institute of Standards and Technology (NIST), 1992, pp. 21–30 (cit. on pp. 24, 67).

[Rob+93] Stephen E. Robertson et al. "Okapi at TREC-2." In: *TREC*. Ed. by Donna K. Harman. Vol. 500-215. NIST Special Publication. National Institute of Standards and Technology (NIST), 1993, pp. 21–34 (cit. on pp. 24, 67).

[Rob+94] Stephen E. Robertson et al. "Okapi at TREC-3." In: *TREC*. Ed. by Donna K. Harman. Vol. 500-225. NIST Special Publication. National Institute of Standards and Technology (NIST), 1994, pp. 109–126 (cit. on pp. 24, 27, 51, 67, 71).

[Rob04] S. Robertson. "Understanding Inverse Document Frequency: on Theoretical Arguments for IDF." In: *Journal of Documentation* 60 (2004), pp. 503–520 (cit. on p. 24).

[Rob97] Stephen E. Robertson. "The probability ranking principle in IR." In: *Journal of Documentation 33* (1997), pp. 294–304 (cit. on p. 21).

[RW94] Stephen E. Robertson and Steve Walker. "Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval." In: *SIGIR*. Ed. by W. Bruce Croft and C. J. van Rijsbergen. ACM/Springer, 1994, pp. 232–241 (cit. on p. 26).

[Sal71] G. Salton, ed. *The SMART Retrieval System Experiments in Automatic Document Processing.* Englewood Cliffs: Prentice-Hall, 1971 (cit. on p. 19).

[SBM96] Amit Singhal, Chris Buckley, and Mandar Mitra. "Pivoted Document Length Normalization." In: *SIGIR*. ACM, 1996, pp. 21–29. ISBN: 0-89791-792-8 (cit. on p. 15).

[SFW83] Gerard Salton, Edward A. Fox, and Harry Wu. "Extended Boolean Information Retrieval." In: *Communications of the ACM* 26.11 (1983), pp. 1022–1036 (cit. on p. 10).

[SP04] L.B. Sollaci and M.G. Pereira. "The introduction, methods, results, and discussion (IMRAD) structure: A fifty-year survey." In: *Journal of the Medical Library Association* 92.3 (2004), pp. 364–367 (cit. on pp. 3, 35, 67).

[SWY75] G. Salton, A. Wong, and C. S. Yang. "A Vector Space Model for Automatic Indexing." In: *Communications of the ACM* 18.11 (1975), pp. 613–620. ISSN: 0001-0782 (cit. on p. 12).

[SY73] G. Salton and C. S. Yang. "On the specification of term values in automatic indexing." In: *Journal of Documentation.* 29.4 (1973), pp. 351–372 (cit. on pp. 16, 17, 20).

[VIN15] S. Vijayarani, M. J. Ilamathi, and M. Nithya. "Preprocessing Techniques for Text Mining - An Overview." In: *International Journal Computer Science and Communication Network* 5 (2015), pp. 7–16 (cit. on p. 37).

[Zip32] G.K. Zipf. *Selected Studies of the Principle of Relative Frequency in Language*. Harvard University Press, 1932 (cit. on p. 16).