CHALMERS

Calendar-Based Relevance Boosting

Master of Science Thesis

MARCUS CHRISTIANSSON
PETER WINTZELL

Department of Computer Science
Software Engineering and Technology
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2012

Calendar-Based Relevance Boosting

MARCUS CHRISTIANSSON
PETER WINTZELL

Examiner: ROBERT FELDT
Supervisor: SVEN-ARNE ANDREASSON

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden, 2012

Abstract

Information retrieval systems have become a more central component in modern Software Engineering. It is hence important to improve the models that determine relevance, both concerning what data is used and how to use the data, particularly to support more personalized systems. Among the multiplicity of models to determine relevance, context-based search aims to improve the relevance of retrieved information by considering in what context a search is being carried out. This study evaluates how improved search relevance can be achieved by considering the context of the calendar. Hence, a model concept was constructed, implemented in a typical enterprise search environment and furthermore evaluated using statistical testing. The results showed a statistically significant improvement on the search relevance using keyword matching. Nevertheless, temporal methods of determining context had no significant impact on the search relevance. Future works include evaluation of the concept in an authentic environment to determine concept applicability.

Keywords: information retrieval, context-based search, calendar context search, calendar-based relevance, information retrieval evaluation

Contents

Chapter 1

Introduction

"A squirrel dying in front of your house may be more relevant to your interests right now than people dying in Africa."

— Mark Zuckerberg, founder and CEO of Facebook

The information retrieval (IR) field has due to the growing amount of digital information become more central in Software Engineering and thus the subject of much research. A big issue is the concern of matching the information need of users with the retrieved results, also referred to as *relevance*. However, there is a consensus in the field that the concept of relevance is challenging to define and furthermore more challenging to model[1]. In spite of considerable advances in the field there is still indeed a demand for improving relevance in IR systems.

The ambiguity of users expecting different results when entering the same data is one of the biggest issues, which moreover has contributed to the need of personalized search. Today, Facebook filters the news feed based on users' relationships and previous interactions and even when a user is not logged in Google consider 57 different signals for a search[2]. The amount of information, as well as the applicability of the information, is two key elements in the success of any relevance model and furthermore key elements in a personalized IR system.

The main difficulties of personalized IR, where the information availability naturally is a prerequisite, concerns determining what user data is relevant to the current user information need and furthermore how that data can be used to influence the relevance in a search. Determining relevance of this data is from a user perspective seemingly intuitive but from a computational perspective complex. This complexity combined with the variety of search environments gives the area of personalized search countless implementations.

Commonly seen as a subsection of personalized search is *context-based search* where the IR system consider in what context the search is being carried out. This could refer to user physical location, user search phrase history or user calendar events. The latter will be explored in this thesis by the rationale that near events may be relevant to a user's information need.

1.1 Problem

Events near in time are likely to be relevant to a user's information need. When a user performs a search the hours before or after a calendar meeting it is possible that the carried out search is related to the near meeting. Hence this thesis assumes the following:

> **Assumption 1:** *When a user performs a search it possibly exists relevant events in the calendar that are likely to be near in time.*

This study aims to explore the possibility of using context-based search with calendar data as basis. To accomplish this, the following research questions will be answered:

> **RQ 1:** *What empirical studies have been conducted on context-based search?*
>
> **RQ 2:** *How can context be derived from calendar data and improve relevance in an information retrieval system?*
>
> **RQ 3:** *How can personalized information retrieval systems where context is based on calendar data be evaluated statistically?*

1.2 Method

This study was conducted in cooperation with Findwise AB, a consultancy company specialized in enterprise search. In order to answer the research questions the thesis was divided into three consecutive phases:

1. **Literature review** The goal of the literature review was to examine the current state of research and build a theoretical framework for the case study and its evaluation. The study reviewed information in the subject of information retrieval, personalized search and calendar data by searching prestigious digital libraries, books and web pages. Furthermore, to gain some industrial experience consultants from Findwise were interviewed.

2. **Case study** The goal of the case study was to create a model concept that uses calendar data to improve the relevance of search results. Additionally, the model was implemented in a typical Findwise technical environment to illustrate the solution applicability in industry as well as to be able to evaluate the solution quantitatively.

3. **Testing** In order to evaluate the model as well as answer the research questions, the model was tested according to well-recognized procedures at Findwise. The test results were at the end analyzed to determine whether the model significantly improved the search results.

Chapter 3 gives a more detailed description of how this thesis was conducted.

1.3 Delimitations

The study is restricted by the following delimitations:

- This thesis does not investigate assumption 1.
- This thesis does not discuss indexing data source.
- This thesis does only focus on meta data available in Microsoft Outlook and Apple iCal.
- All implementations are in line with, and furthermore delimited by, the software architecture of Findwise's solutions.

Chapter 2

Theory

The conducted literature study used *ACM Digital Library* as primary database. Where no relevant result was available *IEEE Xplore Digital Library* was used followed by *Google Scholar*. Following search terms were derived from chapter 1:

```
personalized search, personalizing search, personalizing web search,
personalized information retrieval, context-based search, context-based
rank, context-based information retrieval, calendar-based search,
calendar context search, calendar-based relevance, calendar search,
information retrieval, information retrieval evaluation
```

To expand the theoretical framework, further search terms were derived from the resulting papers:

```
vector space model, precision, recall, f-score, mean reciprocal rank,
statistical significance tests, wilcoxon signed rank text, sign test,
student's paired t-test
```

Furthermore, the case study required implementation specific literature, which was obtained from Findwise and by using the following search terms:

```
apache solr, date boosting
```

The following chapter presents the current state of research in the field and the main technologies used in this study. The results presented here serves as a theoretical framework for remaining chapters.

2.1 Context-based search

Search engines are rather commonly criticized for delivering the same results to all users for the same queries[3]. This is problematic since the need and goal for different users may be completely different for the same query[3, 4]. Numerous studies have been made of personalizing the search and thus differentiating this ambiguity, among others

[5, 6, 7, 8, 9, 10]. These studies evaluate different methods of considering the user context in searches.

Many attempts use automatic, although often with one or several manual tasks, categorization of either the user or the search data. In [5], where the latter is applied, publications are assigned into manually pre-determined contexts. Papers are then scored with respect to their context. When a search is performed a context is selected during the query, either by the user or automatically. The automatic context selection uses two approaches: a text-based similarity measure and context keyword matching. The search is after the context selection carried out in the selected context using the context score as an additional parameter in the relevance model.

Reichhold et al.[6] explored categorization of the user by using user roles. This study was conducted in an enterprise environment where they manually create user roles representing the user context by considering the user function, job description, location etc. The role definitions are handled by a role expert who also assigns roles to each employee. They present *Role Term Vectors* that are attached to the roles with weighted terms which are used to determine document relevance for a role. A merged score is furthermore computed based on the original document rank and the document relevance for the user role.

Several other studies propose an analysis of pre-query activity such as previous queries, search engine result page (SERP) clicks[9] and user browser behavior[11]. White et al.[9] introduces user interest models for current query, pre-query session activity and a combination of the two as *intent*. Matthijs and Radlinski[11] generates a user profile which is represented by a list of terms and weights, visited URLs and their number of visits and a list of previous queries and SERP-clicks.

Even context-based search for 3D models has been proposed where, besides from names of the models, object shape and spatial relationships are considered for the relevance model[8]. Similar to most attempts, this study included a manual step by allowing the user to select a region through a bounding box in the scene to determine context.

Personalized search has indeed numerous implementations. The information available as well as the structure of the information varies considerably with different domains, which often leads to the studies being somewhat domain-specific. Overall, most studies display good results. Dou et al.[3] evaluated five different click- and profile-based click strategies using click-logs data. On queries with large click entropy all strategies significantly improved the relevance. However, on queries with small click entropy the strategies sometimes had a negative impact. Similar results was found in [5] where the general results where improved, but some inaccuracy was found as well and all three proposed strategies in [11] displayed significantly better results than the original Google ranking.

2.2 Information retrieval

Information retrieval concerns the techniques of storing and retrieving data. Earlier more trivial models are boolean systems based on boolean logic[12]. The boolean model is easy to implement and intuitive but have several disadvantages; the difficulty to rank output is perhaps the most prominent one. Ranking of the result however is needed and expected of most users. Most IR systems rank the relevance of documents for a given query and assign a numeric score to each document. One of the most frequently used models in IR research is the *term vector model.*

2.2.1 Vector space model

Vector space model, sometimes referred to as term vector model, was first introduced in 1975[13]. Here documents and queries are represented as vectors of terms where terms typically are single keywords and longer phrases[12]. Each term corresponds to an independent dimension in a high dimensional space. When determining the relevance of a document for a query in the vector space model the similarity of the document vector and the query vector is measured. Each term in the document that matches a term in the query is assigned a non-zero value in the dimension corresponding to the term. The cosine of the angle between the resulting document vector and the query vector, referred to as cosine similarity, is calculated as

$$Sim(D_i,Q) = \frac{D_i \cdot Q}{\|D_i\|\|Q\|} = \frac{\sum_{j=1}^{n} D_{i,j} \times Q_j}{\sqrt{\sum_{j=1}^{n}(D_{i,j})^2} \times \sqrt{\sum_{j=1}^{n}(Q_j)^2}} \tag{2.1}$$

where D_i is the vector representing document i, Q the query vector, $D_{i,j}$ the jth term in the document D_i and Q_j the jth term in the query. The cosine similarity can furthermore consider terms that are more important. In the case of information retrieval, the terms are often weighted with respect to inter- and intra-document frequency.

2.2.2 TF-IDF

Term Frequency - Inverse Document Frequency, *tf-idf*, is a numerical statistic which is used to determine how important a certain word is for a document in a set of documents. It is often used by search engines to determine the relevance of each document with respect to the search query[12]. The mathematical formula below shows how it is calculated:

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \tag{2.2}$$

where $tf_{t,d}$ represents how many times the term t occurs in the document d. The second term (idf_t) represents in how many documents the term occurs. By this definition the following conclusions can be drawn:

- If a term T occurs frequently in a document D but in few other documents, D is likely relevant for T
- If a term T occurs frequently in a document D and in many other documents, D might not be relevant for T since T is a commonly used term

- If a term T occurs infrequently in document D and in few other documents, D could be relevant for T since T is an unusual term
- If a term T occurs infrequently in document D but in many other documents, D is likely not relevant for T

2.2.3 Boosting

Similar to how *tf-idf* regards the frequency of a term in a document and in a whole set, other aspects must be considered as well. E.g. Gelbukh[14] made the assumption that sentences near the beginning and end of a document is important and proposed a scoring model to consider this assumption. Another possibly more trivial way of achieving better relevance is to consider the fact that some documents are more important overall by increasing their score and thus boosting the relevance of these documents. One of the most common used boosting functions is *field boosting* with the intuitive, and also by Findwise empirically proven, rationale that the information in some fields are more important than other, e.g. the title is more important than a paragraph of a document. Boosting with respect to document age is presented in the following section.

Date boosting

The importance or relevance of news, business and local documents are often more related to date than regular web search. Newly edited or added documents are often of greater interest to the user than outdated news or documents. Hence, it is in these domains reasonable to consider document age in the relevance model. One of the most commonly used models to boost documents based on date is called “recency boost”[15] and is calculated as

$$boost = a/(m \times x + b) \tag{2.3}$$

where x is the time difference from events' dates in milliseconds and current date and a, b and m are configurable constants.

The constant m is defined to be 3.16×10^{-11} which is the inverse of the total number of milliseconds in a year. A result of this is that the term $m \times x$ will become greater than 1 if the event does not occur within a year and less than 1 if it occurs within a year.

One method of handling old documents is to only consider documents that have been edited in given time interval, e.g. the last year. This means that documents outside this time interval will not be considered at all, which might be inappropriate in some cases. To handle this issue, one could use an alternative method that bottoms out documents edited outside the interval by setting a minimum value for the date boosting.

2.3 Calendar

A typical calendar is populated by events, which also are referred to as appointments or meetings, however with some different fields depending on implementation. This thesis abstracts all three types to the common definition *events*. Furthermore, an event may in most implementations have a number of participants and may belong to one or several categories. A simplified calendar meta-model constructed with support for formats used by Microsoft Outlook and Apple iCal[16] can be seen in figure 2.1.

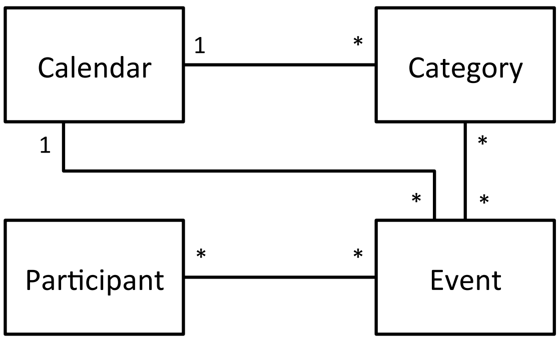

Figure 2.1: Calendar meta model.

As displayed by the meta-model a calendar can have multiple events and categories. An event must belong to a calendar, may belong to one or several categories and may furthermore contain several participants.

2.3.1 Calendar data fields

A basic calendar event can be represented with the following fields:

- Title of the event
- Description text of the event
- Start and end date of the event (including time if necessary)
- Name of the location where the event takes place
- A list of names of the participants

In the simplest case an event is only represented by title or description and the event dates.

2.4 Software

In a typical software solution for a customer at Findwise there are four components; an index service, a search engine (e.g. Apache Solr, Atonomy IDOL and Microsoft FAST), a transparent service-layer (Jellyfish) and a graphical user interface (GUI). In almost all cases the GUI is a web application developed in Java. Among the available search engines, the most common one for Findwise's customers is Apache Solr.

2.4.1 Apache Solr

Apache Solr is a stand-alone open-source enterprise search platform written in Java. It includes features such as powerful full-text and faceted search and supports several data sources such as databases, XML-files and rich documents (e.g. Word-documents). Documents are stored in an index that follows a schema which describes the documents' meta-data. The platform is highly scalable which simplifies implementation of custom functionality[17].

Query structure

A query in Apache Solr is represented by a string containing words separated with an operator (AND or OR). Using an AND-operator between two words in a query results in that only documents containing both words will match the query. In contrast, only one of the two words needs to exist in the document when the OR-operator is used[17, 18].

```
query = "dog OR cat"
```

The example above demonstrates a query where only documents containing the words "dog" or "cat" will match.

Query boosting

To enrich a query further, each word in the query can be boosted with a value (default value is 1)[19]. The bigger the value, the more important the word is for that query. If boost is not given for any of the words all words in the query are valued equally. To demonstrate the opposite, the example below illustrates when words are not valued equally.

```
query = "dog^5 OR cat^1"
```

In the example above the word "dog" is worth five times more than the word "cat". Hence documents containing the word "dog" are more likely to receive a higher score than documents containing the word "cat".

Document structure

All indexed documents follow a specific schema that represents the structure of the documents. The schema describes the meta-data with fields of various types (string, integers, booleans etc)[17]. To demonstrate this, a simplified schema is shown in table 2.1. Having the data separated in different fields allows the possibility of searching for documents that contain a specific word in a specific field, e.g. all documents that contain the word "dog" in the field *title*.

Field	Type	Boosting value
id	integer	
title	string	5
text	string	1

Table 2.1: Schema explaining the document structure.

Field boosting

It is also possible to boost fields by the same principle as for the query boosting[17, 19]. Findwise normally boosts the *title* field five times more than the *text* field (see table 2.1). However the values are relative which implies that there are no standard reference values (e.g. maximum value)[19].

Document score

Each document in the index will receive a score based on the query. The score gives an indication on how relevant the document is to the query and is normally used to sort the documents in the search result list. However there might be occasions where sorting on other fields is more convenient (e.g. date sorting for news).

There are no limits or default values for the score which results in some considerable consequences. Firstly, it is impossible to calculate the maximum score and therefore not possible to transform the score to a percentage[20]. Another consequence is that documents' relevance can not be determined individually by looking at the score but needs to be put in relation to other documents.

Document scores are based on a number of different factors. The main factor is the value of the *tf-idf* (see section 2.2.2). Although, the field and query boosting will affect the total score as well[17, 21].

2.4.2 Jellyfish

Jellyfish is a transparent service-layer that operates between the search engine (e.g. Apache Solr) and the GUI (e.g. a web application) (see figure 2.2). This means that the GUI does not communicate directly with the search engine which in turn makes it possible to exchange the search engine without affecting the GUI components [22]. Jellyfish is developed by Findwise and is commonly used in customer projects.

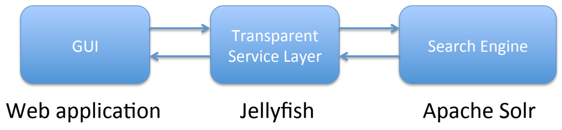

Figure 2.2: Overview of Jellyfish.

Pipeline step

As figure 2.3 shows, Jellyfish basically consists of two pipelines [22]. The first pipeline serves a purpose to process the query while the second pipeline processes the result instead. The pipelines are composed of pipeline steps of either type `QueryModifier` or `ResultModifier` [22]. For example, to remove all special characters in a query, one can add a pipeline step of type `QueryModifier`. The number of steps in a pipeline is unlimited. Both modifiers mentioned above have their own interface. To extend the functionality in a pipeline one has to create a Java class that implements the corresponding interface and configure Jellyfish to include the new pipeline step.

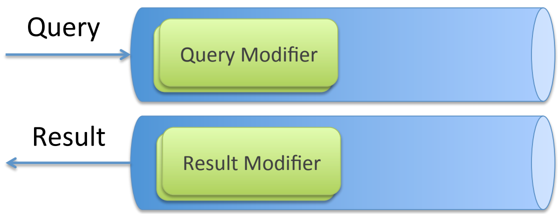

Figure 2.3: Pipelines for query and result processing.

2.5 Information retrieval evaluation

Many measuring techniques of evaluating the performance of a retrieval system require that every document in the test set is either relevant or non-relevant to the query. Among the multiplicity of techniques, three standard fundamental indicators are often discussed: precision, recall and F-score[23]. Precision is defined as

$$precision = \frac{TP}{TP + FP} \tag{2.4}$$

where TP stands for true positive, which in this case also can refer to the set of relevant retrieved documents, and FP stands for false positive which similarly can refer to the set of non-relevant retrieved documents. The sum $TP + FP$ can furthermore refer to the set of retrieved documents. Recall is defined as

$$recall = \frac{TP}{TP + FN} \tag{2.5}$$

where FN stands for false negative, which here refer to the set of relevant non-retrieved documents. Thus, precision is the fraction of the result that is relevant to the search and recall is the fraction of all relevant documents that are retrieved in the results. The two measures alone are not sufficient to measure the performance. However, there is a well-known fact that there is a trade-off between the two measures[24]; hence a combination of the two, F-score or F-measure, can be used

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{\beta^2 \times precision + recall} \tag{2.6}$$

where $\beta = 1$ implies a balanced F-score (also referred to as F_1 measure). Other commonly used F measures are F_2 that weights recall twice as much as precision and $F_{0.5}$ that weights precision twice as much as recall.

This measure is only a single-value metric that is not sufficient when retrieving a ranked list. An elaborated measurement, average precision (AP)[24], can instead be used

$$AP = \sum_{i=1}^{k} p(i) \times \Delta r(i) \tag{2.7}$$

where i is the rank in sequence of the result, $p(i)$ is the precision when using i as cut-off (the maximum number of results to consider) and $\Delta r(i)$ is the difference in recall from $i-1$ to i. Mean Average Precision (MAP) could also be applied by computing average position over a set of queries.

2.5.1 Mean reciprocal rank

The amount of relevant documents in a search is sometimes fairly small[25]. Known-item search refers to a search where only one document is relevant and is a common use case in most search systems. This type of search is better tested by measuring techniques that only regard the relevant document position in the result. A commonly used measure here is the Mean Reciprocal Rank (MRR)[25, 26, 27]

$$MRR@k = \frac{1}{|Q|} \sum_{q \in Q} \begin{cases} \frac{1}{1+r_q}, & r_q < k \\ 0, & r_q \geq k \end{cases} \tag{2.8}$$

where q is a query in the set of queries Q, r_q is the rank at which the relevant document is found and k is the cut-off. The measure evaluates the quality of a result by the time it takes to find the document. Assuming that the relevant document can be found at position n, all positions up to n has to be looked at, leading to an efficiency of $\frac{1}{n}$. Hence, the measure values changes in ranking at a higher ranks more than changes in lower rank positions. This can be illustrated by considering the comparison of two algorithms in table 2.2.

Query	**Alg 1** r_q	**Alg 2** r_q	**Alg 1** RR	**Alg 2** RR
cat	1	2	1	$\frac{1}{2}$
dog	7	2	$\frac{1}{7}$	$\frac{1}{2}$
horse	5	4	$\frac{1}{5}$	$\frac{1}{4}$

Table 2.2: Rank and reciprocal rank for two algorithms.

The above data gives $MRR(Alg1) = \frac{47}{105}$ ($\approx$ 0,45) and $MRR(Alg2) = \frac{5}{12}$ ($\approx$ 0,42) using cut-off $k \geq 8$. Even though algorithm 2 performed better considering rank alone (Alg 1 $\sum_{q \in Q} r_q = 13$ and Alg 2 $\sum_{q \in Q} r_q = 8$) the loss of a top rank impacts the MRR to the extent that algorithm 1 is considered to perform better.

2.5.2 Student's paired t-test

In the information retrieval community three test of statistical significance are commonly used: the Wilcoxon signed rank test, the sign test and the Student's paired t-test. However, according to Smucker et al.[28] both Wilcoxon and sign test display a poor ability to detect significance. Hence, the use of the Student's paired t-test is recommended amongst the three.

Student's paired t-test assumes that the observed data are drawn from a population with normal distribution. This assumption is according to [29] reasonable given that the p-value produced by the t-test is close to the p-value produced by the randomization test, which it is in practice has been proven to be[28]. The test is well suited for comparing a group of objects that has been tested twice, e.g. before and after a treatment. The null hypothesis is typically that the mean difference of the two samples, before and after in our example, is equal. The t statistic to test the hypothesis can be calculated as follows

$$t = \frac{\overline{X}_D - \mu_0}{s_D/\sqrt{n}} \tag{2.9}$$

where $\overline{X}_D$ is the mean of the differences, μ_0 the specific value the mean is equal to under the null hypothesis, s_D the standard deviation of the differences and n the sample size. Once the t statistic and the degrees of freedom, $df = n - 1$, is calculated, the p-value can be found using a t-table.

Chapter 3

Methodology

The project was to a large extent using an iterative process. Apart from the initial literature review each iteration ended up with a meeting with a Findwise supervisor, where deliverables was presented and evaluated. The feedback from these meetings served as valuable input in the decision-making process. Additionally, the next iteration goals were determined during the final part of this meeting.

The iterations concerned the following topics:

1. Environment setup and data source indexing
2. Calendar data structure analysis, parsing and indexing
3. Relevance model concept development
4. Relevance model refinement
5. Relevance model evaluation

This chapter describes how this thesis was carried out in three phases: an initial literature review to be followed by a case study and a final testing phase. The sections are presented in chronological order.

3.1 Literature review

The literature review can be divided into five different areas, jointly covering all the given research questions. In each area research papers were reviewed to get a theoretical understanding. In addition, practical examples were studied to achieve a better understanding which aided the model implementation in the case study.

3.1.1 Context-based search

As a first and possibly most important part of the literature review, other attempts on using user context as an additional source of information for improving the search relevance were studied. Studies that included a practical implementation were in addition studied from an architectural perspective.

3.1.2 Information retrieval

This area focused on fundamental concepts of information retrieval and how relevance in terms of text analysis is determined. Models and principles that are used both in enterprise search as well as in other search areas were covered here.

3.1.3 Calendar data

Some focus in the literature review was on calendar meta-data and calendar meta-model. As mentioned in section 1.3, only data available in Microsoft Outlook and Apple iCal was considered.

3.1.4 Software architecture

Since this thesis included a model implementation, some research concerned the software components the that implementation was based on. The limitations of the software components were essential information in the construction of the model.

3.1.5 Information retrieval evaluation

This part of the literature review focused on finding an appropriate test method in order to evaluate the model. Methods of measuring relevance as well as methods of determining statistical significance of test results were studied.

3.2 Case study

The main goal of this thesis was to construct a model that, by using the context of the user calendar, can achieve more relevant search results. The development of this model consisted of three iterations where the model concept firstly was created, secondly improved and thirdly implemented.

3.2.1 Technical environment setup

Before any model was constructed the technical environment was set up. The selection of software components was made in cooperation with Findwise in order to support future implementations in industry.

3.2.2 Model concept

Based on the knowledge from the literature review, an initial concept of the relevance model was created. The concept was theoretically constructed, however the limitations of the software components was considered throughout the whole process.

3.2.3 Implementation

The model concept was implemented by extending the functionality of the existing software components. The implementation part also included indexing a data source in order to evaluate the model in the next phase.

3.3 Testing

The model was tested in order to evaluate the model impact on the search relevance. The outcome of this last phase determined whether the model fulfilled its purpose stated in section 1.1.

3.3.1 Use cases

The testing was based on 50 use cases that each contained three elements; a query, a calendar event related to the query and a document related to the query. The idea of the use cases was to simulate how the model would perform in real environments, e.g. when a user with a relevant calendar searches for information.

3.3.2 Statistical testing

All use cases were tested with a framework for relevance testing that Findwise normally uses. The outcome of the testing is the positions of the related documents in the search result lists. The positions were evaluated using mean reciprocal rank and furthermore tested for statistical significance using Student's paired t-test. Moreover, in order to optimize the result, the use cases were tested in iterations with different model configurations.

Chapter 4

Design

This chapter presents the model (referred to as relevance model) that was constructed after the literature review. The first three sections describe the model through a theoretical perspective and the last section describes the software architecture of the model.

4.1 Model concept

The overall idea of the relevance model is to enrich the search query by including more words to it. Based on the interviews, the more information a query has the more likely it is for the search engine to return relevant results for the user. The added words are extracted from the events in the calendar where only events that seem relevant with respect to the original query are considered. As an example, when a user searches for "amazon" and has an event in the calendar including the term "Amazon.com", the search query will add "Amazon.com" to the query. This will probably boost documents related to the online book store and punish documents related to the amazon rainforest.

When a search is performed it is processed in five consecutive phases shown in figure 4.1. The second phase (Query processing) and the fourth phase (Result processing) conforms to the two types of pipeline steps discussed in section 2.4.2. The idea of the relevance model is to modify the query in the Query processing phase before it is sent to the search engine.

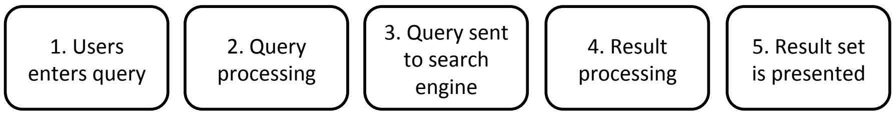

Figure 4.1: Elements of query and result processing [30].

4.1.1 Structure

The relevance model consists of three different data sets; the original query, the calendar query and the final query. All three data sets consist of words together with their corresponding boosting value. Hence the final query, which is a combination of the original query and the calendar query, will be in the same format as the original query and therefore the structure of the query remains unchanged.

If a user enters a query S, the query can be defined as

$$Q_{original} = \{(w,b_0) : w \in S\} \tag{4.1}$$

where w is a word in the query and b_0 is the word's boosting value. Since S is entered by the user it will be a plain string and therefore all words will have a default boosting value of 1. The format of the query corresponds to the query format in Apache Solr (see section 2.4.1).

Similarly, the calendar query is defined with the exception that the words are taken from a data set C rather than from the user query S. This data set consists of words from calendar events that seem relevant with respect to S. Section 4.2 describes how C in more detail is defined.

$$Q_{calendar} = \{(w,b) : w \in C\} \tag{4.2}$$

The final query (the query that is sent to the search engine) is a merged query of both the original query and the calendar query. A detailed description of this can be found in section 4.3.

$$Q_{final} = Q_{original} \cup Q_{calendar} \tag{4.3}$$

4.1.2 Calendar data schema

The calendar data is indexed in the search engine and follows a given schema. The fields in the schema is shown in table 4.1 together with their boosting values. These values are standard values that Findwise normally uses for similar schemas. Note that two of the

Field	**Type**	**Boosting value**
Title	string	10
Text	string	2
Participant	string	1
Location	string	1
Start date	date	
End date	date	

Table 4.1: Calendar data schema.

fields (*start date* and *end date*) do not have any boosting value. This can be explained by the fact that these fields are not used for keyword extraction, but to determine how relevant a certain event is (see section 4.2.5).

4.1.3 Relevance parameters

There are three parameters that are specific to the relevance model and affects the relevance of the search results. Although, there are more parameters that affects the search relevance such as field boosting and query boosting. These are however not specific to the relevance model and therefore not considered to be part of the relevance model.

- **Number of words** The number of words from the calendar query that will be added to the final query. If this parameter is set to 0 the final query will be the same as the original query which results in no impact on the relevance. The mathematical definition of this parameter is $numWords$ and section 4.2.4 describes how it is used.
- **Query boost** The boosting value for the words in the original query. This boosting value will replace the default boosting value b_0 for all words and is mathematically defined as *queryBoost*. Section 4.3 describes how the boosting values are exchanged.
- **Date boost** The boosting value for how much the events' dates should affect the relevance when finding the most relevant events. This parameter should be considered with respect to the field boosting values for the data fields. The mathematical definition is *dateBoost* and section 4.2.5 describes in detail how this parameter is used.

To optimize the results the parameters above needs to be configured. There is no optimal configuration since the result is highly dependent on the data source the search engine is working with. According to the consultants at Findwise, one has to look from case to case to set appropriate values for the parameters.

4.2 Extracting calendar data

One of the most important parts of the model is the part that extracts words from the calendar. Only words relevant to the original query is added to the calendar query. Two of the parameters mentioned in section 4.1.3, $numWords$ and *dateBoost*, are involved in this.

4.2.1 Words processing

Given the calendar meta-model in section 2.3.1 the events from the calendar can be described as shown in the following formula:

$$E = \{(e_i, s_i) : e_i \in I, i < 0 < n\} \tag{4.4}$$

where E is a set containing n events that are found in the calendar index I. Each event e_i also has a score s_i that is determined by the search engine [19]. Formula 4.5 shows that each event e_i has a set W_i that contains all words related to that event.

$$\forall e_i : W_i = \{(w_{i,j},b_{i,j}) : w_{i,j} \in e_i, 0 < j < m_i\} \tag{4.5}$$

where m_i is the number of words for the event e_i and $w_{i,j}$ is a word with a corresponding boosting value $b_{i,j}$ found in that particular event. The calculation of this boosting value is explained in section 4.2.3. Furthermore, all n sets of words can be defined in one singular set showed in the following formula

$$W = [W_1 \cup W_2 \cup ... \cup W_n] \tag{4.6}$$

where W is a set containing all words from all n events. Since W is unsorted, W_{sorted} is defined to be the sorted set where the sorting is based on the boosting values $b_{i,j}$.

$$W_{sorted} = \{(w_k,b_k) : 0 < k < r\} \tag{4.7}$$

where r is the total number of words from all n events. Notice that a word w_k is no longer related to the event it originally belonged to. As a last step, W_{sorted} is limited to only contain the first $numWords$ words. Section 4.2.4 further explains this limitation.

4.2.2 Finding relevant events

The calendar index I contains all events. However, only events that are considered relevant to the original query should be in the set E (see section 4.2.1). To find these events two aspects are taken into count:

1. How well the words in the event match the original query.
2. How near in time the events occur.

A combination of these two aspects will determine which events are most relevant. The query used to retrieve events from the calendar index is basically $Q_{original}$ but with a minor modification. If the relevance only would consider the first aspect (keyword matching) there would be a risk that no documents are found. To prevent this, there is an additional term added to the final query: "*". The term "*" matches all words which means that all documents in the index will match that term as well[18], hereafter referred to as *star-search*. Furthermore, the relevance needs to be based on something other than keyword matching, which leads us to the second aspect which is described in more detail in section 4.2.5.

4.2.3 Query boosting

As already mention in section 4.2.1, each word in the calendar query has a boosting value. This value represents how important the word is in relation to the other words in the query. The following formula 4.8 describes how the boosting value is defined:

$$\forall w_{i,j} : \exists b_{i,j} = s_i \times t_{i,j} \tag{4.8}$$

where s_i is the score of the event the word is extracted from and $t_{i,j}$ is the *tf-idf* value of the word. Hence, a word is worth more if it is extracted from an event with a high score or if the word's *tf-idf* value is high (or both). In other words, if the event seems relevant or if the word is unusual.

In contrast, a word is worth less if it is extracted from an event with a low score or if the word's *tf-idf* value is low (or both). This seems reasonable since words that are very common and extracted from irrelevant events should not be considered important.

4.2.4 Query limitations

Since the calendar can contain an infinite number of events, the calendar query could contain an infinite number of words. A large query sent to the search engine will affect the performance. A study from Google showed that their traffic decreased with 20 % when their search became a half second slower (even if the number of results per page increased from 10 to 30) [31].

As an example, consider a calendar with 200 events where each event contains 50 words. In total, there would be 10 000 words added to the query since all events are considered relevant with different impact. Therefore, only $numWords$ words are added to the calendar query (see formula 4.9).

$$C = \{(w_k,b_k) : w_k \in W_{sorted}, 0 < k < numWords\} \tag{4.9}$$

where C is the final set of words w_k (with corresponding boosting value b_k) that are added to the calendar query (see section 4.1.1).

4.2.5 Date boosting

There might be cases when the calendar does not contain any relevant events with respect to the original query. In other words, there are no words among the events that matches the original query. However, as mentioned in the introduction of this paper, an event in the near future (or an event that just occured) could still be highly relevant even of there is no keyword matching. Therefore, there must be something that complements the keyword matching.

To solve this issue, the dates of the events are also considered to influence the relevance. This implies that all events in the calendar will be considered with different degree of relevance. For example, events that recently occurred will be boosted and therefore more relevant than events that occurred a long time ago. This new dimension of the relevance is referred to as "date boosting" and uses the same principle mentioned in section 2.2.3. Although, there are few minor adjustments to avoid that old documents are punished. The boosting value is calculated as

$$boost = dateBoost \times \frac{a}{m \times min(|ms|,limit) + b} \tag{4.10}$$

where ms is the number of milliseconds between current date and the event's start date, *dateBoost* the relevance parameter that determines the impact of the date boosting and *limit* the maximum value in milliseconds between current date and the event's start date. The constants a, m and b remains unchanged.

The first difference compared to formula 2.3 is that the absolute value of ms is used in order to handle events that already occurred. The second difference is that the minimum of $|ms|$ and *limit* is used instead of only $|ms|$. The rationale for using *limit* is to avoid that old documents will receive an insignificant boosting value. In this case, *limit* is set to 2.59×10^9 which corresponds to the number of milliseconds in a month.

4.3 Combining two queries

The final query (Q_{final}), which replaces the query that is sent to the search engine, is based on the following:

- Original query ($Q_{original}$)
- Data from the calendar ($Q_{calendar}$)

These two queries have to be weighted with respect to each other. One can never know if an event retrieved from the calendar is relevant by looking at the score[20]. In this case, where the words' boosting values in the calendar query are affected by the scores returned from the search engine, the boosting values for the words in the original query can not be constant.

Consequently the weighting has to be done by tuning the configurable parameters in order to achieve the best possible result. In the relevance model all words in the original query are given a boosting value, *queryBoost*:

$$\forall(w,b_0) : w \in Q_{original}, b_0 = queryBoost \tag{4.11}$$

4.3.1 Calendar data quality

The use cases displayed in table 4.2 and table 4.3 illustrates the problem where it is impossible to determine how relevant data from the calendar is. In case A the word with highest boosting value is "polo" (0.8) and in case B the word with highest boosting value is "human" (0.01). Thus, "polo" is more relevant for the original query than "human" (80 times more important). Furthermore if one decides that the words in the original

Case	Original query	Calendar query
A	horse	polo^0.8 ...
B	horse	human^0.01 ...

Table 4.2: An example demonstrating relevance differences among queries.

query is always twice as much worth as the maximum boosting value from the calendar query, it would give the values showed in table 4.3. It may seem reasonable if one looks at the cases individually. However, if one compares case C with case D one can see that "human" is considered to be worth as much as "”polo" in relation to "horse".

The problem here is, as already stated in section 4.3, that there are no limits to refer to [19]. The values in the tables 4.2 and 4.3, could as well be 80 (for case A), 1 (for case B), 160 (for case C), 2 (for case D) and still be as much worth relative to each other. Therefore, with this approach, it is impossible to determine the boosting values for the original query without considering in which context it is going to be used.

Case	Original query	Calendar query
C	horse^1.6	polo^0.8 ...
D	horse^0.02	human^0.01 ...

Table 4.3: An example demonstrating query relationships.

4.4 Software architecture

Due to the delimitations of this thesis the software solution follows architectural guidelines provided by Findwise. The software components of the solution are commonly used in Findwise's customer solutions, which allows potential integrations in future. Figure 4.2 shows how the two main components, which are the transparent service-layer Jellyfish and the search engine Apache Solr, relate to each other.

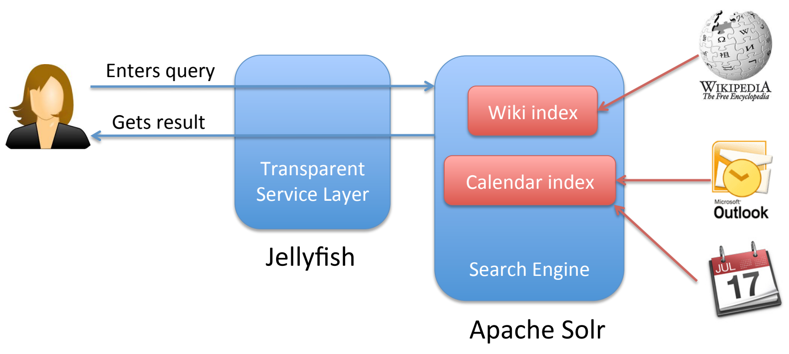

Figure 4.2: Main components of the software solution.

4.4.1 Separate indexes

A difference compared with a typical Findwise solution is that there are two indexes in this solution. That is due to the fact that there are not only one but two data sources that need to be indexed:

- English articles from Wikipedia.

- Events from the calendar.

The english articles from Wikipedia are indexed in order to illustrate a data source, which are needed for the testing phase. This index, named *wikipedia index*, could in real customer cases be e.g. documents from an intranet such as PDF-files, word-documents or contact information for the employees. Furthermore, events from the calendar are placed in another index in order to separate the two data sources. This second index, named *calendar index*, follows the schema described in section 4.1.2. The indexes are illustrated in figure 4.2 by red boxes.

The fact that the indexes are separated is a consequence of how relevance is determined in Apache Solr. Since the document scores are based on *tf-idf* among other factors[21], the data sources need to be indexed separately in order to not affect the calculated *idf* values. Otherwise event scores will be based on both data from the events and the articles. Hence, in order to make the data sources independent they need to be separated. Furthermore, separated indexes also adds to the scalability of the solution. In order to integrate this solution in an existing one, an additional index needs to be added while the original index can remain the same.

4.4.2 Two instances of Jellyfish

Due to the fact that there are two indexes in Apache Solr there are also two instances of the service-layer Jellyfish (one connected to *wikipedia index* and one to *calendar index*). In figure 4.3 these instances, both containing their own pipelines, are illustrated with blue color.

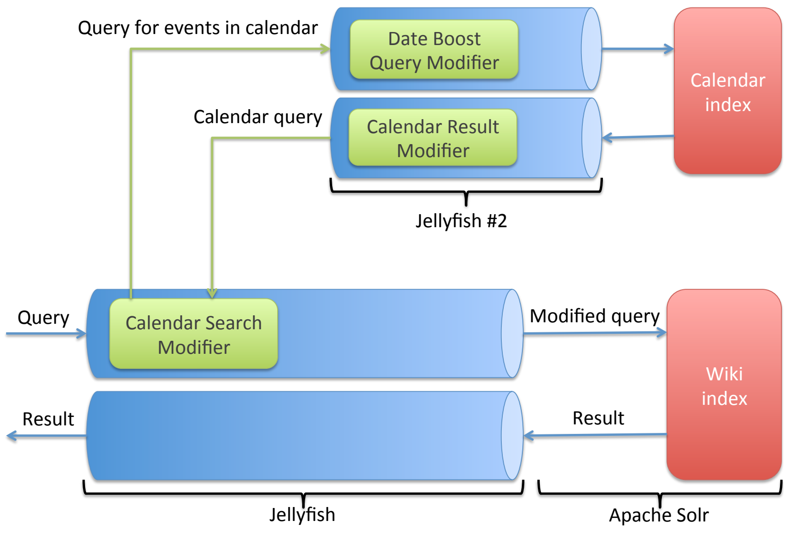

Figure 4.3: Two Jellyfish instances, one for each index.

There are three different pipeline steps that represent the main functionality of the solution. These three steps are named `CalendarSearchModifier`, `DateBoostQueryModifier` and `CalendarResultModifier` and are described in more detail in section 4.4.3. The green boxes in figure 4.3 show where these pipeline steps are placed in an architectural perspective.

4.4.3 Pipeline steps

Two of the pipeline steps, `CalendarSearchModifier` and `DateBoostQueryModifier`, have functionality that to some extent modifies the query and are therefore of type `QueryModifier`. However the third step, `CalendarResultModifier`, is instead of type `ResultModifier` though its functionality modifies the result. Furthermore, each pipeline step represents a specific part of the relevance model, which is described below. To increase the understanding on details, pseudo code for all three steps is found in appendix C.

Calendar Search Modifier

This pipeline step can, to a great extent, be mapped with the concept of replacing the original query explained in section 4.1.1. Thus, the step makes the basis of the relevance model and in order to disable the model this step could be removed and hence there would be no impact on the search relevance.

The replaced query, Q_{final}, is a combination of the original query ($Q_{original}$) and the calendar query ($Q_{calendar}$). Furthermore, the data for the original query is available in the pipeline step while querying the *calendar index* for events retrieves the data for the calendar query.

Date Boost Query Modifier

The purpose of this pipeline step is to boost events that occur in the near future (or events that recently occured). The boosting is performed by one of the standard features in Apache Solr, which supports boosting documents based on their data[15, 17], and is based on the same formula as described in section 4.2.5. All the mathematical operators required by the boosting formula are supported by Apache Solr[19].

In practice, the boosting is not performed in this pipeline step but in Apache Solr. Hence, this pipeline step only creates the formula, which in turn is sent to the search engine as an additional parameter to the query.

Calendar Result Modifier

The default structure of results returned from Apache Solr[17] does not correspond to the query format described in section 2.4.1. Thus, this pipeline step modifies the result by putting all words from all documents, with their corresponding boosting value, in a list and furthermore returns it instead. A word's boosting value is calculated as described in section 4.2.3, i.e. the word's *tf-idf* value multiplied with the score of the event the word belongs to.

4.4.4 Calendar data indexer

The relevance model requires that words from the calendar events can be extracted and in order to do this, the events must be indexed. The functionality that indexes the calendar events into Apache Solr is independent from the rest of the solution and is shown in figure 4.4.

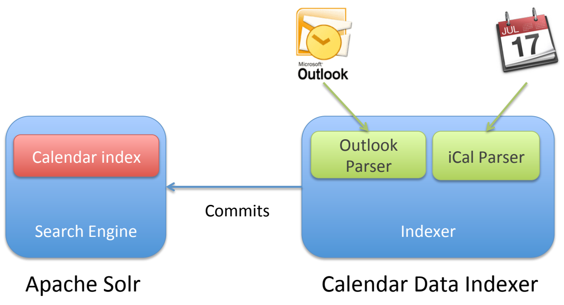

Figure 4.4: Components for indexing calendar data.

The *calendar index* contains events from both Microsoft Outlook and Apple iCal. Thus, both of the calendars have their own parser which is illustrated by green boxes in figure 4.4. In addition to the fact that separated parsers adds to the scalability of the solution, it also allows indexing the calendars independently.

The architecture is designed to support multiple parsers (not only the parsers mentioned previously). In order to index events from a calendar, a parser that implements a specific interface must be created. The interface guarantees that the format of the data, which the parsers commit, corresponds to the calendar schema (see section 2.3.1).

Chapter 5

Test results

The following chapter presents the use case structure and its rationale, the test results from three test iterations and a comparison of the test results with and without the relevance model.

5.1 Use Cases

Use cases in this study consist of a query and a single relevant document, also referred to as known-item search (see section 2.5.1). Table 5.1 presents a known-item use case example which represents a search for the site "http://chalmers.se" using a query consisting of the terms "university" and "gothenburg".

Id	**Query**	**URL**
Test1	university gothenburg	http://chalmers.se

Table 5.1: A known-item search use case.

The fundamental assumption 1 in this thesis states that when a user performs a search there may be an upcoming or recent event which is related to the search. As mentioned in section 3.3.1, each use case thus has a corresponding calendar event. The use case mentioned above is extended in table 5.2 with a corresponding event presented in table 5.3.

Id	**Query**	**URL**	**Event id**	**Test type**
Test1	university gothenburg	http://chalmers.se	Event1	Positive-match

Table 5.2: A known-item search use case with calendar event.

Worth noting is that the event displayed in table 5.3 is simplified by removing location and participant data. All use cases and events used in this study can be found in appendix A and appendix B respectively.

Event id	Title	Dates	Text
Event1	Chalmers	2012-01-01 08:00-09:00	Visit Chalmers University.

Table 5.3: A simplified calendar event.

In order to get objective test data there are both tests that should have a positive and a negative influence on the search. A positive use case has a corresponding event in the calendar which is relevant to the expected document. In contrast, a negative use case denotes that the corresponding event in the calendar is not relevant to the expected result. Moreover another aspect of the tests is considered where the positive and negative use cases are divided in matching and non-matching categories:

- **Positive-match** A term in the query exists in a relevant event.
- **Positive-non-match** A relevant event exist but matches no term in the query.
- **Negative-match** A term in the query exist in non-relevant event.
- **Negative-non-match** No relevant event exist and no events matches any term in the query.

There are a total of 50 use cases with the following distribution: 20 positive-match, 10 positive-non-match, 10 negative-match and 10 negative-non-match. The rationale for having 20 positive-match versus 10 negative-match is to further advance assumption 1 by assuming that events that have a matching keyword are more likely to be relevant to the search. Thus, the assumption is quantified by having the double amount of positive-match use cases.

In addition, each use case is tested with different time intervals between the search date and the start date of the relevant calendar event. The intervals used are one hour, one day, one week and lastly one month.

5.2 Iteration results

As introduced in section 3.3.2, the testing is carried out in iterations to optimize the parameters presented in section 4.1.3. Parameters included in the optimization process are *queryBoost* and *dateBoost*. The parameter $numWords$ is constant at this phase in the testing. As initial maximum and minimum values for the parameters, $queryBoost_{min}$ and $dateBoost_{min}$ are set to 1 and $queryBoost_{max}$ and $dateBoost_{max}$ are set to 20. The values are selected with respect to the minimum and maximum boosting of fields in the calendar. The minimum values of the parameters are equal to the minimum field boosting value and the parameters' maximum values are double the maximum field boosting value in the calendar schema. The maximum limit value here is selected by the rationale that it is of interest to see the behaviour of boosting values over the maximum calendar field boosting value.

Using the limits specified above, and three additional values between the limits for each parameter, results in twenty-five different configuration combinations. Figure 5.1 and 5.2 presents the results of these configurations by plotting the average mean reciprocal rank of the $dateWeight$ and $queryBoost$ respectively.

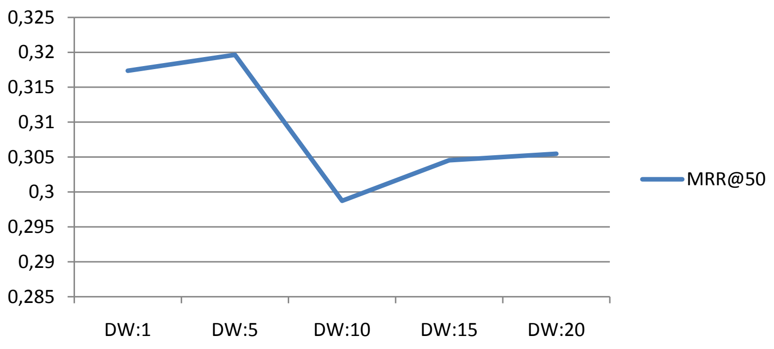

Figure 5.1: Average MRR@50 for $dateWeight$

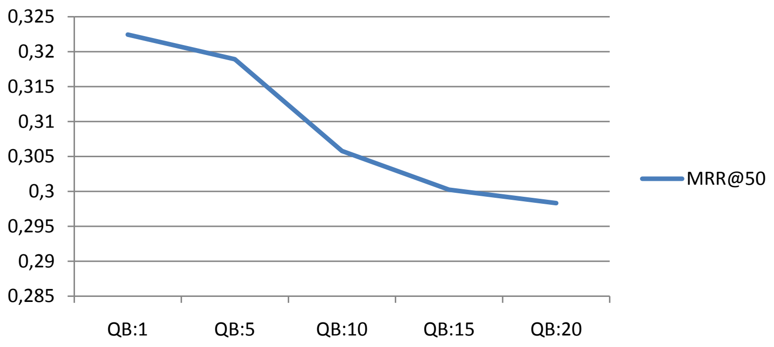

Figure 5.2: Average MRR@50 for $queryBoost$

A simple analysis of the two graphs would lead to the conclusion that the best result in this test set is achieved by using $queryBoost = 1$ and $dateWeight = 5$. Given this information, further testing should be focused on testing in the area of these two values. However, the best rank in the first iteration is found at $queryBoost = 1$ and $dateWeight = 1$. Hence the optimization of the two parameters can not be performed individually since they to a large extent are co-dependent.

The graph 5.3 presents the mean reciprocal rank of the first iteration results in a three-dimensional graph. By looking at the graph a trend of better results can be seen where $queryBoost$ and $dateWeight$ both approach 1. Given this trend, one could optimize the parameters further by testing around the current maximum at $queryBoost = 1$ and $dateWeight = 1$. Using parameter values $queryBoost = \{0.5,0.75,1,1.25,1.5\}$ and $dateWeight = \{0.5,0.75,1,1.25,1.5\}$ in iteration two similarly gives a trend of better results where $queryBoost$ and $dateWeight$ both approach 1.5. Therefore the third iter-

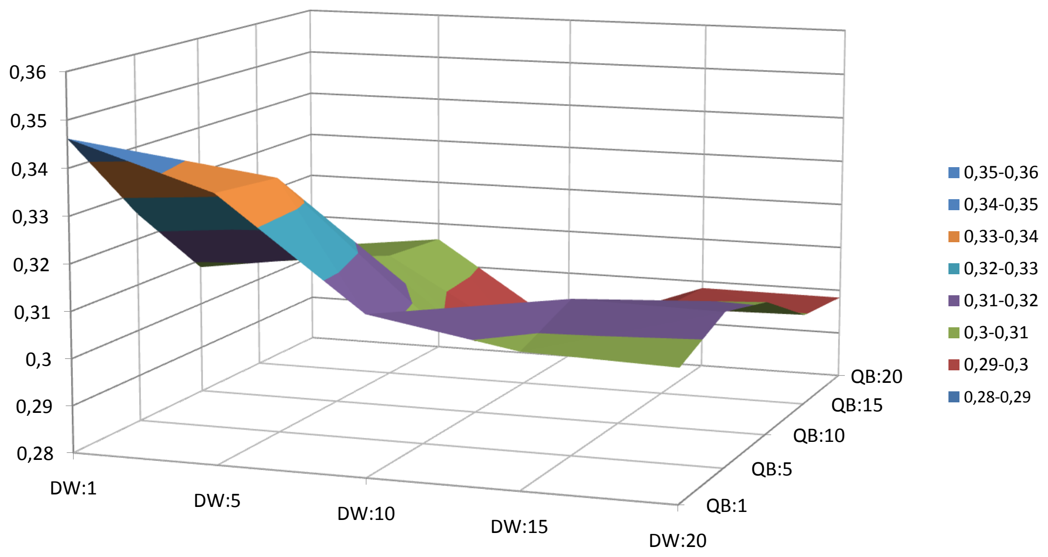

Figure 5.3: $MRR@50$ using parameter values $queryBoost = \{1,5,10,15,20\}$ and $dateWeight = \{1,5,10,15,20\}$.

ation tests the values above the values in the second iteration with the same intervals. Figure 5.4 displays the results of iteration three.

The configuration with the highest mean reciprocal rank ($MRR@50 = 0{,}358$) in all tests presented in this thesis can be found at $queryBoost = 2$ and $dateWeight = 2$. Due to the co-dependency of the parameters a more extensive testing process with significantly smaller intervals may lead to better results using another configuration but this will not be further evaluated in this thesis.

5.3 Search with vs. without model

Testing the use cases presented above without the model results in a $MRR@50 = 0{,}230$. A comparison of the results without and with the optimized model is presented in table 5.4. In addition to the total results the table presents a comparison of the MRR of the different test types individually. A Student's paired t-test shows that the improvement on the test with all test types is statistically significant (p-value $< 0{,}001$). Furthermore, the increase in positive-match and negative-non-match as well as the decrease in negative-match and positive-non-match are all statistically significant (p-values $< 0{,}005$).

Moreover, table 5.5 presents a comparison of the different time intervals. The biggest difference can be found by comparing the interval of one day with the interval of one week ($MRR@50 = 0.368$ and $MRR@50 = 0.349$). This difference is not, according to the Student's t-test, statistically significant; thus the date boosting impact is not significant. All statistical significance test calculations can be seen in appendix E.

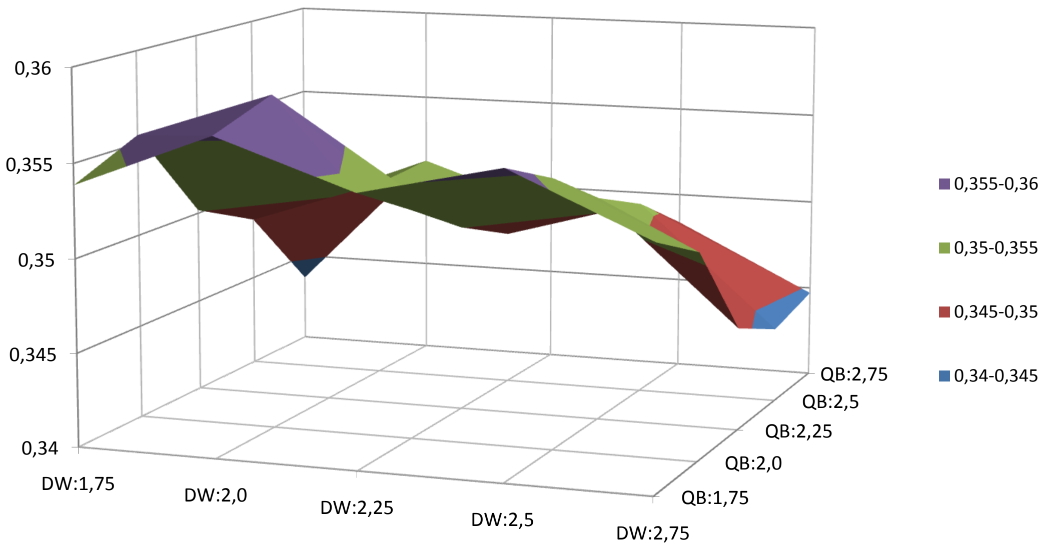

Figure 5.4: $MRR@50$ using parameter values $queryBoost = \{1.75,2,2.25,2.5,2.75\}$ and $dateWeight = \{1.75,2,2.25,2.5,2.75\}$.

Test type	Without model	Optimized model	Difference
All tests	0.230	0.358	0.128
Positive-match	0.079	0.422	0.343
Negative-match	0.495	0.290	-0,205
Positive-non-match	0.086	0.011	-0,075
Negative-non-match	0.409	0.646	0,237

Table 5.4: $MRR@50$ for different test types without vs. with relevance model.

Date interval	Optimized model
1 hour	0.352
1 day	0.368
1 week	0.349
1 month	0.363

Table 5.5: $MRR@50$ for different time intervals with relevance model.

Chapter 6

Discussion

This chapter discusses the experiences acquired from the testing phase, critical parts of the relevance model and finally alternative approaches that could be applied to the model, as well as their exclusion motivation.

6.1 Test results reflections

Testing relevance for a context-aware information retrieval system is a complex task. As mentioned in chapter 1, determining relevance computationally is problematical and is further complicated by considering the seemingly vague concept of context. To model realistic use cases in a realistic context, which will be further elaborated in section 6.2, is crucial for obtaining reliable test results.

Testing computationally implies some quantification of the result, which sometimes makes it tempting to draw overhasty conclusions based on the result. As an illustration one could consider the discovery described in section 5.2; when the relevance parameters are individually examined, the highest score is received by using values 1 and 5. However, when they are examined together the best result is received when both values are set to 2. Hence there is no clear relation between the relevance parameters, i.e. too many hidden factors are involved. In order to explain why these values give the best result one needs to do extensive analyses of all parameters and hidden factors.

As could be expected, the test results demonstrate that there is an improvement for the positive-match use cases and a negative effect for the negative-match tests. This indicates that it is possible to improve relevance by using data from a calendar, given that assumption 1 is accurate.

Opposed to the matching tests, the non-matching use cases surprisingly demonstrate inverted results. There is a decrease for the positive-non-match tests while there is an increase for the negative-non-match tests. This can be explained by the fact that the events are randomly selected (due to the star-search) and hence the test result is too dependent on the events in the calendar. Furthermore, it is suggested to only consider events that are relevant using keyword matching or by using a more advanced relevance matching approach, e.g. the one proposed by Ratprasartporn et al. [5].

Based on assumption 1 the highest score for the positive use cases should be received when the events occur in one hour and the lowest score should be received when the events occur in one month. This was however not demonstrated by the test results where the highest score was received in the following order: day, month, hour and week. Hence, the date boosting did not behave as intended and the order previously mentioned could be a result of the random selection. No difference between the date intervals were statistically significant, which implies that the date boosting de facto did not have any impact on the results at all.

6.2 Use cases and data source impact

The test results are highly dependent on the test data, i.e. the use cases, the calendar events and the data source (Wikipedia). Other use cases and a different data source would most likely result in other numbers than the ones presented in this thesis. This does not however imply that the tests display incorrect results but simply that it is related to the quality of the test data.

A weakness with the use cases is that each use case only considers one single event. This event is in the positive use cases relevant and not relevant in the negative use cases. Using the keyword matching alone, only relevant and non-relevant events with matching keyword are considered. The influence of all events, in terms of user overall context, is not considered which furthermore implied no impact on the non-matching use cases. Hence, the star-search was introduced which in contrast results in all use cases matching all events. In table 6.1 one can see this by taking the sum of *TP* and *FP* for each test type, which equals the total number of events. A consequence of the star-search is that there is significantly more *FP* than *TP*, which in turn results in a low precision. In contrast, the precision would be high if the star-search had not been introduced but that would also result in the positive-non-match use cases being uninfluenced (*TP* would become *TN*).

Test type	**TP**	**TN**	**FP**	**FN**
Positive-match	1	0	19	0
Positive-non-match	1	0	19	0
Negative-match	0	0	20	0
Negative-non-match	0	0	20	0

Table 6.1: Number of events received for respective test type.

Most use cases in the test set represent different non-relating domains which consequently implies that most calendar events are not related to other events. This represents a user context poorly since a user's calendar events to a larger extent more likely are related. Implementing the solution in an authentic environment in industry, with authentic calendars, would probably give a more specific context and thus more relevant events for each use case. Non-matching use cases would in this environment possibly display better results since more events in the result would be categorized as TP. Furthermore, testing the relevance model in an authentic environment, with an improved date boosting discussed in section 6.3, could also evaluate the date boosting's applicability.

6.3 Boosting errors in the model

One of the first problems discovered in the analysis of the test results were that the relevance model uses scores from the search engine in an incorrect way. All scores are given as relative values without any boundaries (e.g. no maximum value). As a consequence, a score for a given document can only be related to other scores for documents that was received under the same conditions, i.e. using the same query. For example, three documents (A, B and C) are received for a specific query with the scores: 1, 2 and 20 respectively. Based on these scores document A is considered to be the least relevant document and its score is furthermore 5 % of the maximum result. However, by removing document C only two documents (A and B) will be retrieved for the same query. Given unaffected scores, by ignoring factors such as *tf-idf*, document A score is now 50 % of the maximum result for the same query. In addition, one could further consider the change in *tf-idf* that may occur by removing document C, which would give a change in the score values as well.

Further complications with using the score can be seen by considering the date boosting. The highly tunable parameters in the date boosting function presented in section 2.2.3 could easily be changed to represent specific requirements. One could as an illustration determine that an event currently occurring is twice as relevant as an event occurring in eight hours and hence tune the function to match this requirement. The scores returned by this function would however not match the requirement and furthermore be different for different queries.

To base boosting values for words in a query on scores, though how intuitively compelling it may seem, can thus be inaccurate. However, this is implemented in the relevance model by multiplying a word's *tf-idf* value with the score of the event it is extracted from. To solve this issue, the number of relevance parameters in the model is minimized and the values of the remaining parameters rely on comprehensive testing. Utilizing this approach is supported by the common practices at Findwise were each customer solution is configured individually to optimize the relevance.

The *tf-idf* value is well-recognized and used among several search engines to determine relevance. One of the concepts of *tf-idf* is to punish, in all documents, frequently occurring words to leave emphasis on less common words. However, as with the score, the *tf-idf* value is not used in a correct way in the relevance model. For example, if a user's calendar frequently contains the word "music" it is likely that the user is interested in music which furthermore would be a word describing the context well. Conversely, the word "music" will have a low influence on the search since the *tf-idf* value is low.

The last problem that was discovered during the testing is that the model does not take into account in which fields the extracted words from the calendar come from. To some extent, the search engine will do that when the calendar index is queried by using the field boosting. However, the score of a document reflects the contents of the whole document and not the specific fields. If a document receives a high score because it contains one of the keywords from the query in the title field, all the words from the document's other fields (e.g. text field) will benefit from this.

6.4 Alternative approaches

During the design phase of the case study several different approaches were considered. The alternative solutions solved issues previously discussed but also introduced other disadvantages.

Most discussions concerned how to combine the original query with the calendar query. One of the first suggestions, when only the keyword matching cases were considered, was to only use the latter query. Only events that contained the words in the original query were to be retrieved which makes this suggestion feasible. The benefit of this approach is that there will be a relationship between the words in the original query and the words retrieved from the calendar. This relationship could be used to specify the importance of the original query versus the words added from the calendar.

However, this solution was not applicable when considering non-matching events since words from the original query would not exist in the final query. Intuitively, removing words from the original query should be avoided to ensure that the system returns results that reflect the supplied query logically. To illustrate this it would be inaccurate if a user enters the search string “dog food” and only gets results based on the word “food” since no calendar events contained the word “dog”. All words in the original query therefore need to exist in the final query as well, which introduces further difficulties when determining boosting values.

Another topic that was discussed extensively concerned the date boosting. As the relevance model is constructed in this thesis, events will be boosted based on their date. This is to satisfy assumption 1 that states that relevant events are more likely to be near in time. Besides from determining relevance between matching events, the date boosting is the only factor that impacts the relevance for the non-matching use cases. However, the test results show that the relevance is not affected at all by the date boosting for the non-matching use cases, which thus would suggest a model were no non-matching events should be matched.

An alternative interpretation of the insignificantly small variations of the date boosting could be that the date boosting function should be more aggressive. The parameter $dateWeight$ were suppose to represent this in the optimization process but this was not reflected since the increase of the parameter caused many non-relevant events to be more significantly used in the query. Another approach could be to use a more aggressive date boosting function or preferably implement the date relevance directly in the model and thus not use the search engine score.

The testing furthermore tested the same matching use case by different time intervals which may caused the optimized model to decrease the significance of date since the optimization only pursued best total results. Hence the use cases in the optimization could instead consider assumption 1 to a larger extent by valuing use cases, with relevant event near in time, more.

6.5 Context based on calendar data

In summary, on basis of the results presented in this thesis, it is suggested to use a model that only uses events matched by keywords or a more advanced model for keyword matching. The model should use a more aggressive date boosting function or preferably implement the date boosting directly in the model and thus not use the search engine score.

As proven by the literature study, a collection of keywords that are relevant to a context can be used to improve relevance in a search. Given that upcoming or recent events are relevant to the context, the calendar can successfully be used to determine context and furthermore improve the performance of the information retrieval system. The big issue for future works does not regard specific implementation, optimization or testing but an analysis of the fundamental assumption. This analysis should be performed by testing the above-suggested implementation in an authentic environment. If shown by this analysis that usage patterns of the calendar conforms to this assumption and that the events furthermore contains keywords relevant to the context, the implementation in this thesis could be used to improve the performance of an information retrieval system.

6.6 Implications for Software Engineering in general

Seen context-based search from a wider perspective one could state that any information, given that it is has a textual representation, could be used to influence a search. If the information is considered to be relevant to the information need of the user it can in any information retrieval system in some manner be added to enrich the query, or to filter the results. The difficulty lies within determining the relevance of the information for each given search and user. The implementations of determining relevant information are countless, especially considering domain-specific solutions. However, the rather simple method of matching a keyword can be applied on many cases which moreover could imply that any set of documents containing user relevant data can be used to improve the relevance of a search. This can, as explored in this thesis, then furthermore be improved by considering other relevance methods such as date boosting for calendar events or possibly boosting documents which is currently being edited by the user.

Most information representing a user context can be evaluated using the same statistical evaluation method suggested in this thesis. It does however require that the software architecture of the information retrieval system allows add-on functionality. In this thesis, the created model consists of different independent features that are integrated in a software without introducing any new limitations. Hence it allows software to be extended to a context-aware system using one or several methods of modelling the context without limiting or making any significant changes to the software.

Chapter 7

Conclusion

Previous research concerning personalizing search have been conducted where context is determined by categorizing users manually and automatically, analyzing click-logs data or analyzing pre-query activity such as previous queries. No research however was found were context is based on information from the calendar. Hence a method of deriving context from the calendar and using the context in a search was developed. The model proposed in this study found relevant calendar events by matching keywords in the search query with keywords in the calendar events. An additional model that boosted events near in time was used to further determine relevance, but did however in the tests display poor results. The query was enriched with words extracted from the relevant events. The relative significance of the words was determined by the relevance of events and words.

This study evaluated the information retrieval system statistically by setting up use cases consisting of a query and a single relevant document. The position of each document was calculated to quantify the results. Evaluating a context-aware system however requires a computational representation of the context where all context data is added to each use case which was done by having one single related event for each use case.

The model used in this thesis did not introduce any new limitations on the software which thus implies that most information retrieval systems can be extended to be aware of the context of the calendar and furthermore aware of the context of the user in general. The big issue, which is not explored in this thesis, concerns if a typical user calendar is used in a way that is computationally usable and relevant to the user information need, which should be the subject of future research.

Bibliography

[1] B. Hjørland, The foundation of the concept of relevance, J. Am. Soc. Inf. Sci. Technol. 61 (2) (2010) 217–237.
URL http://onlinelibrary.wiley.com/doi/10.1002/asi.21261/full

[2] E. Pariser, Beware online "filter bubbles" (May 2011).
URL http://www.ted.com/talks/lang/en/eli_pariser_beware_online_filter_bubbles.html

[3] Z. Dou, R. Song, J.-R. Wen, A large-scale evaluation and analysis of personalized search strategies, in: Proceedings of the 16th international conference on World Wide Web, WWW '07, ACM, New York, NY, USA, 2007, pp. 581–590.
URL http://doi.acm.org/10.1145/1242572.1242651

[4] R. Krovetz, W. B. Croft, Lexical ambiguity and information retrieval, ACM Trans. Inf. Syst. 10 (2) (1992) 115–141.
URL http://doi.acm.org/10.1145/146802.146810

[5] N. Ratprasartporn, J. Po, A. Cakmak, S. Bani-Ahmad, G. Ozsoyoglu, Context-based literature digital collection search, The VLDB Journal 18 (1) (2008) 277–301.
URL http://www.springerlink.com/index/10.1007/s00778-008-0099-9

[6] M. Reichhold, J. Kerschbaumer, G. Fliedl, Optimizing enterprise search by automatically relating user context to textual document content, in: Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies, i-KNOW '11, ACM, New York, NY, USA, 2011, pp. 22:1–22:6.
URL http://doi.acm.org/10.1145/2024288.2024316

[7] C.-T. Li, M.-K. Shan, S.-D. Lin, Context-based people search in labeled social networks, Proceedings of the 20th ACM international conference on Information and knowledge management - CIKM '11 (2011) 1607.
URL http://dl.acm.org/citation.cfm?doid=2063576.2063809

[8] M. Fisher, P. Hanrahan, Context-based search for 3D models, ACM SIGGRAPH Asia 2010 papers on - SIGGRAPH ASIA '10 29 (6) (2010) 1.
URL http://portal.acm.org/citation.cfm?doid=1866158.1866204

[9] R. W. White, P. N. Bennett, S. T. Dumais, Predicting Short-Term Interests Using Activity-Based Search Context (iii) 1009–1018.

[10] Z. Ma, G. Pant, O. R. L. Sheng, Interest-based personalized search, ACM Transactions on Information Systems 25 (1) (2007) 5–es.
URL http://portal.acm.org/citation.cfm?doid=1198296.1198301

[11] N. Matthijs, F. Radlinski, Personalizing web search using long term browsing history, in: Proceedings of the fourth ACM international conference on Web search and data mining, WSDM '11, ACM, New York, NY, USA, 2011, pp. 25–34.
URL http://doi.acm.org/10.1145/1935826.1935840

[12] A. Singhal, Modern information retrieval: A brief overview, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4) (2001) 35–42.
URL http://singhal.info/ieee2001.pdf

[13] G. Salton, A. Wong, C. S. Yang, AVector Space Model for Automatic Indexing 18 (11).

[14] A. Gelbukh, Computational Linguistics and Intelligent Text Processing: 13th International Conference, Cicling 2012, New Delhi, India, March 11-17, 2012, Proceedings, no. del 2 in Lecture Notes in Computer Science / Theoretical Computer Science and General Issues, Springer, 2012.
URL http://books.google.se/books?id=xxBlBAfgc4kC

[15] T. Potter, Boosting Documents in Solr by Recency, Popularity, and User Preferences.
URL http://www.lucidimagination.com/sites/default/files/Potter%20Timothy%20-%20Boosting%20Documents%20in%20Solr.pdf

[16] F. Dawson, D. Stenerson, Internet calendaring and scheduling core object specification (icalendar), Internet RFC 2445 (Nov. 1998).

[17] D. Smiley, E. Pugh, Apache Solr 3 Enterprise Search Server, Packt Publishing, 2011.

[18] R. Kuc, Apache solr 3.1 cookbook, Packt Publishing, 2011, pp. 121–143.

[19] A. S. Foundation, Solr relevancy faq (Apr. 2012).
URL http://wiki.apache.org/solr/SolrRelevancyFAQ

[20] A. S. Foundation, Scores as percentages (Sep. 2009).
URL http://wiki.apache.org/lucene-java/ScoresAsPercentages

[21] A. S. Foundation, Similiraty lucene api (2012).
URL `http://lucene.apache.org/core/old_versioned_docs/versions/2_9_1/api/core/org/apache/lucene/search/Similarity.html`

[22] Documentation of Jellyfish provided by Findwise AB (classified).

[23] C. Goutte, E. Gaussier, A Probabilistic Interpretation of Precision , Recall and F-Score , with Implication for Evaluation (2005) 345–359.

[24] M. Zhu, Recall, Precision and Average Precision (2004).
URL `http://sas.uwaterloo.ca/stats_navigation/techreports/04WorkingPapers/2004-09.pdf`

[25] I. Soboroff, On evaluating web search with very few relevant documents, in: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '04, ACM, New York, NY, USA, 2004, pp. 530–531.
URL `http://doi.acm.org/10.1145/1008992.1009105`

[26] M. A. Najork, Comparing the effectiveness of hits and salsa, in: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, CIKM '07, ACM, New York, NY, USA, 2007, pp. 157–164.
URL `http://doi.acm.org/10.1145/1321440.1321465`

[27] N. Craswell, D. Hawking, S. Robertson, Effective site finding using link anchor information, in: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '01, ACM, New York, NY, USA, 2001, pp. 250–257.
URL `http://doi.acm.org/10.1145/383952.383999`

[28] M. D. Smucker, J. Allan, B. Carterette, A comparison of statistical significance tests for information retrieval evaluation, in: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, CIKM '07, ACM, New York, NY, USA, 2007, pp. 623–632.
URL `http://doi.acm.org/10.1145/1321440.1321528`

[29] R. Fisher, The design of experiments, Vol. first edition, Oliver and Boyd, 1935.

[30] Search query processing, FAST Search Best Practices.
URL `http://download.microsoft.com/download/0/7/3/073431A7-3B32-436A-8DBF-DF5DD2FF0EB6/Search_Query_Processing.pdf`

[31] G. Linden, Marissa mayer at web 2.0 (May 2008).
URL `http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html`

Appendix A

Use cases

Test id	Query String	Site id	Site title	Event id	Test type
Rankingtest1	amazon	id90451	Amazon.com	Event1	Positive-match
Rankingtest2	SEB	id206351	Skandinaviska Enskilda Banken	Event2	Positive-match
Rankingtest3	jaguar	id147676	Jaguar Cars	Event3	Positive-match
Rankingtest4	global warming	id2260887	Politics of global warming	Event4	Positive-match
Rankingtest5	triumph	id929324	Triumph Records	Event5	Positive-match
Rankingtest6	the river	id429230	The River (Bruce Springsteen album)	Event6	Positive-match
Rankingtest7	relevance	id442684	Relevance (information retrieval)	Event7	Positive-match
Rankingtest8	java	id15881	Java (programming language)	Event8	Positive-match
Rankingtest9	princess victoria	id24427	Victoria, Crown Princess of Sweden	Event9	Positive-match
Rankingtest10	SEO	id187946	Search engine optimization	Event10	Positive-match
Rankingtest11	titanic	id19285924	RMS Titanic	Event11	Positive-match
Rankingtest12	link building	id187946	Search engine optimization	Event12	Positive-match
Rankingtest13	CSS	id23290197	Cascading Style Sheets	Event13	Positive-match
Rankingtest14	Anders Karlsson	id9698368	Anders Karlsson (politician)	Event14	Positive-match
Rankingtest15	cayenne	id6528823	Cayenne (programming language)	Event15	Positive-match
Rankingtest16	major	id10690	Men's major golf championships	Event16	Positive-match
Rankingtest17	ping	id6519883	Ping (golf)	Event17	Positive-match
Rankingtest18	Telia	id30004	TeliaSonera	Event18	Positive-match
Rankingtest19	real	id26413	Real Madrid C.F.	Event19	Positive-match

Continued on next page

Test id	Query String	Site id	Site title	Event id	Test type
Rankingtest20	architecture patterns	id48364	Software architecture	Event20	Positive-match
Rankingtest21	amazon	id48139	Amazon rainforest	Event1	Negative-match
Rankingtest22	jaguar	id16217	Jaguar	Event3	Negative-match
Rankingtest23	triumph	id1605133	Triumph International	Event5	Negative-match
Rankingtest24	river	id18842395	River	Event6	Negative-match
Rankingtest25	java	id16527	Java coffee	Event8	Negative-match
Rankingtest26	lake victoria	id9585762	Lake Victoria (Victoria)	Event9	Negative-match
Rankingtest27	CSS	id4359151	Canadian Space Society	Event13	Negative-match
Rankingtest28	cayenne	id90859	Cayenne pepper	Event15	Negative-match
Rankingtest29	Major League	id38776	Major Leauge Baseball	Event16	Negative-match
Rankingtest30	ping	id24265	Ping	Event17	Negative-match
Rankingtest31	online bookstore cd dvd	id90451	Amazon.com	Event1	Pos. non-match
Rankingtest32	Scandinavian Private Bank	id206351	Skandinaviska Enskilda Banken	Event2	Pos. non-match
Rankingtest33	Swallow Sidecar Company	id147676	Jaguar Cars	Event3	Pos. non-match
Rankingtest34	rising temperature earth	id042951	Global warming	Event4	Pos. non-match
Rankingtest35	label meek barrington	id8098051	Triumph Records (United Kingdom)	Event5	Pos. non-match
Rankingtest36	iron man	id1299122	Iron Man (song)	Event6	Pos. non-match
Rankingtest37	relevancy	id442684	Relevance (information retrieval)	Event7	Pos. non-match
Rankingtest38	palace	id322922	Stockholm Palace	Event9	Pos. non-match
Rankingtest39	improving visibility online	id187946	Search engine optimization	Event10	Pos. non-match
Rankingtest40	first division soccer spain	id37981	La Liga	Event20	Pos. non-match
Rankingtest41	nora wall	id24124047	Nora Wall	Event1	Neg. non-match
Rankingtest42	Never Ending Dylan	id7118144	Never Ending Tour	Event2	Neg. non-match
Rankingtest43	frisbee ultimate	id31775	Ultimate (sport)	Event3	Neg. non-match
Rankingtest44	bangladesh	id3454	Bangladesh	Event4	Neg. non-match
Rankingtest45	jesus christ superstar	id7651220	Jesus Christ Superstar (film)	Event5	Neg. non-match
Rankingtest46	wild frontier tour	id27776800	Live in Stockholm: Wild Frontier Tour	Event6	Neg. non-match
Rankingtest47	brolin	id1610526	Tomas Brolin	Event7	Neg. non-match
Rankingtest48	sgt pepper	id6540	Sgt. Pepper's Lonely Hearts Club Band	Event8	Neg. non-match
Rankingtest49	golden ratio	id12386	Golden ratio	Event9	Neg. non-match
Rankingtest50	hussein	id29490	Saddam Hussein	Event10	Neg. non-match

Appendix B

Events

Event id	Title	Dates	Location	Participants	Text
Event1	Amazon book club	2012-01-01 08:00-09:00	Home		A meeting with my book club where we will discuss the book ivanhoe which can be found on amazon.com.
Event2	SEB people search discussion	2012-01-02 08:00-09:00	SEB Stockholm		”Göran wants to discuss the new people search. His mail: ”Hi, I'm not satisfied with the new people search. Lets discuss how we can improve it. Best Regards Göran Johansson +467XXXXXXXX SEB (Skandinaviska Enskilda Banken)”
Event3	Car buyer visit	2012-01-03 08:00-09:00	Home		A guy is interested in buying my Jaguar E-type.
Event4	Environment discussions	2012-01-04 08:00-09:00	World Trade Center, Gothenburg		Some politics will be discussed regarding global warming in general and what we as politicians can do to put more pressure on the car industry.
Event5	Record meeting	2012-01-05 08:00-09:00			Meeting with triumph records.
Event6	Rehearsal	2012-01-06 08:00-09:00	Söder	All band members	”Band rehearsal. Practice the following: Bruce Springsteen - The river Black Sabbath - Paranoid Fred åkerström - Jag ger dig min morgon”
Event7	Calendar relevance	2012-01-07 08:00-09:00	Findwise office, Stockholm	Peter Wintzell, Marcus Christiansson, Simon Stenström	Discussions concerning information science in general, evaluation of a relevance model and the cluster hypothesis.

Continued on next page

Event id	**Title**	**Dates**	**Location**	**Participants**	**Text**
Event8	Java course	2012-01-08 08:00-09:00	Seattle		Java programming course.
Event9	Royal birth	2012-01-09 08:00-09:00	Stockholm		Crown princess Victoria planned birth date.
Event10	SEO	2012-01-10 08:00-09:00	Anywhere		Introduction to search engine optimization.
Event11	Titanic 3D	2012-01-11 08:00-09:00	SF, Stockholm		
Event12	Link Building seminar	2012-01-12 08:00-09:00	KTH, Stockholm		SEO seminar regarding link building without using black hat such as link spamming and link farms.
Event13	CSS seminar	2012-01-13 08:00-09:00	KTH, Stockholm		Continuing going through CSS in the web development course.
Event14	FRA discussions	2012-01-14 08:00-09:00		Anders Karlsson	Make the politician disapprove the FRA-law
Event15	Customer meeting	2012-01-15 08:00-09:00			Forex considering using functional programming for some transaction solutions and cayenne in particular.
Event16	Watch golf	2012-01-16 08:00-09:00	Augusta National Golf Club		Watch the first major.
Event17	Test clubs	2012-01-17 08:00-09:00	Haninge Golf Club		Test clubs with Ping and Taylor Made representatives.
Event18	Contact support	2012-01-18 08:00-09:00			Referring to the following mail: ”Hi! You’ve received a mail from Telia custoner support and you can read by clicking the following link: http://webbguide.telia.se/i.jsp?id=d40087f07c-25e399 Best regards Teliasonera AB”
Event19	O’learys	2012-01-19 08:00-09:00	O’learys, Sveavägen 143		Real Madrid - FC Barcelona
Event20	ADL Course deadline	2012-01-20 08:00-09:00			Finish the software architecture course.

Appendix C

Code

C.1 Calendar Search Modifier

```
public void CalendarSearchModifier(args) {
   String originalQuery = args.getQuery();

   Search calendarSearch = new Search();
   calendarSearch.setQuery(originalQuery);
   Map<String,double> words = calendarSearch.getResults();

   while(i++ < numWords) {
      calendarQuery += words.getKey() + "^" + words.getValue();
   }

   String finalQuery =  originalQuery + calendarQuery;
   args.setQuery(finalQuery);
}
```

C.2 Date Boost Query Modifier

```
public void DateBoostQueryModifier(args) {
   float a = 0.08, b = 0.05, m = 1.64E-9, limit = 6.22E-10;

   int dateBoost = 10; // Relevance parameter

   // Creates formula (in three steps for visually purpose)
   String formula1 = "min(abs(ms(START_DATE, NOW)), msLimit)";
   String formula2 = div(a, sum(b, product(" + formula1 + ", m)));
   String formula3 = formula2 + "^" + dateBoost;
   args.setBoost(formula3);
}
```

C.3 Calendar Result Modifier

```
public void CalendarResultModifier(args) {
   Map<String,double) words = new Map();

   foreach(Document d : args.getDocuments()) {
      foreach(Field f : d.fields()) {
         foreach(Word w : f.words()) {
            double boost = d.getScore() * w.getTDIDF();
            words.put(w, boost);
            }
        }
    }

    args.setResult(words);
}
```

Appendix D

Test results

The table below presents rank, r_q, using the following configuration: $numWords = 15$, $queryBoost = 2$ and $dateWeight = 2$.

Test id	**Without model**	**1 hour**	**1 day**	**1 week**	**1 month**
Rankingtest1	4	1	1	1	1
Rankingtest2	31	17	17	17	17
Rankingtest3	5	6	6	5	5
Rankingtest4	6	7	7	7	7
Rankingtest5	14	15	11	10	13
Rankingtest6	>50	1	1	42	2
Rankingtest7	13	2	2	2	2
Rankingtest8	>50	>50	>50	>50	>50
Rankingtest9	6	1	1	1	1
Rankingtest10	>50	1	1	1	1
Rankingtest11	17	1	1	4	1
Rankingtest12	>50	>50	>50	>50	>50
Rankingtest13	>50	>50	>50	>50	>50
Rankingtest14	3	3	3	3	3
Rankingtest15	17	6	6	6	6
Rankingtest16	>50	>50	>50	>50	>50
Rankingtest17	31	1	1	1	1
Rankingtest18	25	10	>50	>50	>50
Rankingtest19	>50	1	1	1	1
Rankingtest20	12	2	2	2	2
Rankingtest21	3	8	7	8	8
Rankingtest22	1	1	1	1	1
Rankingtest23	28	>50	>50	>50	>50
Rankingtest24	1	46	1	2	5
Rankingtest25	4	>50	>50	>50	>50
Rankingtest26	2	>50	>50	>50	>50
Rankingtest27	>50	>50	>50	>50	>50

Continued on next page

Test id	**Without model**	**1 hour**	**1 day**	**1 week**	**1 month**
Rankingtest28	2	5	5	5	5
Rankingtest29	3	1	1	1	1
Rankingtest30	1	8	12	6	6
Rankingtest31	9	>50	>50	>50	>50
Rankingtest32	6	>50	>50	>50	>50
Rankingtest33	18	>50	>50	>50	>50
Rankingtest34	>50	>50	>50	>50	>50
Rankingtest35	4	>50	>50	>50	>50
Rankingtest36	5	5	8	14	24
Rankingtest37	>50	>50	>50	>50	>50
Rankingtest38	19	>50	>50	>50	>50
Rankingtest39	37	>50	>50	>50	>50
Rankingtest40	>50	>50	>50	>50	>50
Rankingtest41	1	1	1	1	1
Rankingtest42	17	1	1	1	1
Rankingtest43	4	2	2	1	1
Rankingtest44	1	1	1	1	1
Rankingtest45	4	2	2	1	1
Rankingtest46	2	15	14	11	10
Rankingtest47	3	2	2	2	2
Rankingtest48	>50	>50	>50	>50	>50
Rankingtest49	2	3	3	2	3
Rankingtest50	5	1	1	1	1
MRR@50	0,230	0,352	0,368	0,349	0,363

Appendix E

Calculations

E.1 Statistical significance tests

$$t = \frac{\overline{X}_D - \mu_0}{s_D/\sqrt{n}}$$

Withouth model vs. all test types ($queryBoost = 2$, $dateWeight = 2$)

$$\begin{aligned} \mu_0 &= 0, \\ \overline{X}_D &= 0.128, \\ s_D &= 0.439, \\ n &= 200, \\ t &= 4.14 \end{aligned}$$

Without model vs. positive-match ($queryBoost = 2$ **and** $dateWeight = 2$)

$$\begin{aligned} \mu_0 &= 0, \\ \overline{X}_D &= 0.343, \\ s_D &= 0.417, \\ n &= 80, \\ t &= 7.36 \end{aligned}$$

Without model vs. negative-match ($queryBoost = 2$ **and** $dateWeight = 2$)

$$\begin{aligned} \mu_0 &= 0, \\ \overline{X}_D &= 0.206, \\ s_D &= 0.415, \\ n &= 40, \\ t &= 3.14 \end{aligned}$$

Without model vs. positive-non-match ($queryBoost = 2$ **and** $dateWeight = 2$)

$$\begin{aligned}\mu_0 &= 0,\\ \overline{X}_D &= 0.075,\\ s_D &= 0.081,\\ n &= 40,\\ t &= 5.87\end{aligned}$$

1 day vs. 1 week ($queryBoost = 2$ **and** $dateWeight = 2$)

$$\begin{aligned}\mu_0 &= 0,\\ \overline{X}_D &= 0.020,\\ s_D &= 0{,}216,\\ n &= 50,\\ t &= 0{,}65\end{aligned}$$