Framework and API for Assessing Quality of Documents and Their Sources

Radim Svoboda, 185050

April 7, 2013

University of Eastern Finland
Joensuu Campus
Department of Computer Science
Master's Thesis

Abstract

Enormous amounts of both relevant and irrelevant information is available on-line. Because of the fierce competition, business leaders need to access relevant information in time in order to gain appropriate business intelligence before rivals do. This research is a part of an effort to build ***D****ata* ***A****nalysis and* ***V****isualization a****I****d for* ***D****ecision-making* (DAVID) system for finding, extracting, and analyzing business-relevant information from large amounts of automatically collected documents from off-line and on-line sources. Textual information available on the Internet is of varying quality. Hence, a system such as DAVID has to filter out low quality documents which are potentially useless. In order to improve the filtering of relevant information in DAVID, there needs to be a new filtering component which is applied on every new collected document. This thesis describes: (1) Analysis of quality dimensions that can be assessed from the documents collected by DAVID. (2) Comparison of existing information quality frameworks. (3) A new information quality assessment framework and system called ***F****ramework and* ***API*** *for* ***Q****uality* ***A****ssessment of* ***D****ocuments* (FAQAD) (4) Experiments with the new quality framework. Our experimental results show that FAQAD was able to classify as relevant 99.88% of the relevant business articles in our data set, and on the other hand, was able to filter out 85.59% of the e-mail spam in our test data.

ACM Computing Classification System (CCS):

H.3.1 [Information storage and retrieval]: Content Analysis and Indexing – Linguistic processing;
I.2.7 [Natural Language Processing] – Text analysis;
I.7.m [Document and Text Processing] – Miscellaneous;
J.1 [Administrative data processing] – Business;

Keywords: text quality assessment, evaluation, information quality, text mining, natural language processing, business intelligence

Acknowledgements

This work was supported by the "Towards e-leadership: higher profitability through innovative management and leadership systems" project (2009 – 2012), which was funded by the European Regional Development Fund and Tekes - the Finnish Funding Agency for Technology and Innovation.

I would like to thank the University of Eastern Finland for allowing me to be a part of International Master's Degree Programme in Information Technology (IMPIT) and providing me a lot of opportunities to learn, and to improve my skills and knowledge.

I would like to thank my classmate and fellow member of the project, Tabish Fayyaz Mufti, who was my first link to the e-leadership project, and who introduced me to other members and leaders of the project.

My greatest appreciation is for my supervisor, Dr. Tuomo Kakkonen, that accepted me as member of the e-leadership project. I am grateful for his support, motivation, positive attitude and the ability to point me into the right direction when I needed.

At last but not least, I want express my gratitude towards my family, friends, acquaintances and even strangers from the social environment that did not help me directly with the thesis itself, but provided me essential moral support.

Contents

Chapter 1

Introduction

Having and using right information in the right time helps to avoid making inappropriate and irresponsible decisions, and hence, is the key to success in everyday life, and in particular in business. In former times, moving the information on a piece of paper from a place A to a place B by human or by an animal was relatively slow, and in many cases the speed was not sufficient at all. The time of information distribution decreased and the speed of information flow did significantly improve with the discovery of electricity and electromagnetic wave in 19th century [55]. Another rapid and significant change came at the turn 20th and 21st century [21]. There was a boom in the number of the users of the internet, because most of the average households could afford the internet and use it. Today, almost everybody in Western world uses a PC or a mobile device with an internet access to communicate with others or to reach a desired information potentially quickly and easily. On the contrary, reaching the right information, nowadays, might take a while because of the enormous amount of information, both relevant and irrelevant, that is available.

In addition to finding information, anybody with an internet access can contribute and publish information and their own ideas, opinions and experiences. In contrast to the past, the distribution of information is very fast indeed. However, it is not always easy to determine whether the information that is found on the internet is reliable. While in the past, published written texts were in most cases written and edited by professionals, Internet and social media allows for practically anyone to publish his or her opinions at a low cost. Obviously, the *information quality* (IQ) of electronic publications lacking professional supervision vary. The IQ assessment of documents is the key to categorize and filter the documents according to their quality.

IQ has more than one definition, e.g *"The fitness for use of the information provided"* or *"The assurance that the information meets the needs of the consuming business processes"*[51]. In conclusion, the information of low quality is useless. Fortunately, computers with their processing power are able to help delivering the users only with high quality information.

Sandra Gisin, former head of Swiss Re's Group Knowledge and Records Management unit, says[66]:

> "Information Quality is a strategic approach that enables us to consistently deliver highly useful products and services. We are not far away from having a Group-wide culture where expectations for high quality information are the daily norm. Time is money. IQ can help everyone from employees to executive board members get better results."

Business leaders need high quality information in order to run the business properly, and e.g. not to waste resources because of making wrong decisions. Lack of high quality information or issues with trusting information within own company is a characteristic of dysfunctional learning organization [25].

The aim of the current thesis is to choose the right quality dimensions and methods to measure the quality of text documents and their sources and implement a Java *Application Programming Interface* (*API*) that enables texts to be assessed according to the selected dimensions. Implementation of *quality assessment* (QA) API will become a part of a larger *business intelligence* (BI) *text mining* (TM) system called DAVID (Section 2.1). The system was developed in a research project entitled "Towards e-leadership: higher profitability through innovative management and leadership systems"[40]. DAVID processes large quantities of texts and data instead of human workers. The aim of this approach is to help business leaders in making decisions more effectively, i.e. right decisions made faster than by competitors.

1.1 Motivation

Not all the information is as good and of high quality as others. No single person can ever process all the information on the internet by himself to find out which information is of high or low quality. Simply put, the amount of information is too large. Hence, there is a need to automatically process, distinguish and filter information, in order to save the decision-maker's time and allow them to base decisions only on high quality information.

Business leaders, especially in global corporations, have to make tough decisions on a daily basis. These decisions do not only affect the business leader's life but usually, a single decision has an influence on the whole business and its employees. To make these decisions properly and in timely manner, it is required to have enough background information about the issue at hand. That usually includes documents from several sources obtained through multiple channels. Unfortunately, the relevant information are not always easy to find and access. The information might be difficult to locate or even split to several documents, and again it takes time to put the pieces together. The main idea of the project is to make the knowledge presented in text documents more clear and unambiguous by utilizing information technology. This can potentially improve the management of resources in an enterprise [40] .

The current thesis focuses on the quality of information in the context of BI. Every business acquires and collects information and makes decisions upon them. Wrong and confusing information may lead to a wrong decision. Because of the extraordinary amount of information available in modern business environments, business decision-makers need tools to efficiently and accurately collect and process the information. This enables them to access correct information at the right time. This way, business people are more likely to reach well-informed, correct decisions on time, i.e. before the competition does. Savings of time and resources makes business more efficient, and therefore the business gains advantage for competition.

On a practical level, the aim of this thesis is to improve the existing IQ assessment component of the DAVID system.

1.2 Research Problem

The DAVID system extracts knowledge from text based documents using various *natural language processing* (NLP) and TM techniques. NLP enables computers to acquire meaning from a human language input. TM refers to a process of extracting relevant and high quality information from a text source. However, NLP and TM are quite resource-consuming [56]. Moreover, the accuracy of NLP and TM based analysis is dependent on the quality of the inputs. For these two main reasons, we need to handle all the input documents and their data coming into the system differently, so only usable documents are further processed. For example, data can be:

- filtered out completely
- used only for certain type of analysis
- given more or less priority based on the reliability of the source

IQ is not just a single value assessed from a document. IQ consists from several IQ dimensions where each dimension represents a different aspect of quality. However, not all the quality dimensions are available when checking a document. Dimensions such as availability, believability or understandability are impossible or at least very difficult to measure from a text. Hence, the main research problem in this thesis is to figure out which quality dimensions are measurable and how to measure them, and prioritize them to use these information properly. Even with priorities of selected dimensions, it might not be an easy task to calculate predicative overall quality.

1.3 Research Objectives

The IQ assessment API developed in this research will be a component of a sophisticated TM system for BI. The API serves as a gateway for all documents that are processed by the DAVID system. In order to design and implement the API, the thesis seeks answers to the following two main research questions:

A) **How to assess quality of documents and their sources?**

Two main approaches are considered in this thesis:

Linguistic Metrics give the amount of spelling and grammar errors, and also other information such as word-diversity. This approach does not really consider what information is contained in the document, but how it is written. It may give us a rough estimation about the overall quality of the document.

User Ratings tell us what users think about the document. As this is an evaluation done by humans, we can think of it as an accurate, although subjective, evaluation of the quality of a document.

The research question can be further divided into the following sub-questions:

A1) **Which quality dimensions are measurable?**

To find out which dimensions are we able to measure is a starting point. Processing documents and measuring certain quality dimensions from plain text might not be an easy task. With the **linguistic metrics** approach we could measure readability, lexical diversity, and free-of-errors dimensions. **User ratings** could give us more insight about how people perceive the information in documents. Are there going to be several ratings for different aspects of a document? That way we could measure certain dimensions more accurately. Or is there going to be just single rating expressing users' satisfaction about the document? In that case we could have a combined measurement of dimensions such as accuracy, relevancy, timeliness, and since the sources of documents are on-line, also accessibility.

A2) **How to automatically measure each of the selected quality dimensions?**

It would take a lot of resources to implement all the tools that are needed for assessing the selected IQ dimensions. Some tools for this purpose are available as open source, so there is no need to reimplement them. It is necessary to study the available tools, select the appropriate ones and adapt them to be reused as components of FAQAD API.

A3) **How much weight has the reliability of the source?**

The quality of a source of documents is rated in FAQAD based quality of documents originating in that source. The other way around, new documents' quality will be affected by quality of the source. What happens when an excellent article is published on a web site that is not usually considered as a reliable source? And what happens when a terrible article, even by mistake, is published on a well-rated site? Is the article good enough? Most probably, these situations will not occur - at least not often. However, they might occur in real life, and hence, they need to be addressed.

B) **How to utilize the defined IQ measures?**

Once the IQ assessment measures are chosen and implemented, we need to define how to handle the assessment results. The results do not necessarily lead to clear and unambiguous conclusions about IQ of the assessed documents. This research question consists of the following sub-questions:

B1) **How to prioritize the IQ dimensions in order to best utilize the information they provide?**

Since we have to calculate overall quality, prioritizing different quality dimensions and creating a formula to process the different dimensions' scores is a must. A basic arithmetic mean is not sufficient for our purpose. The assessment scores for each IQ dimension are not equally important and, hence, need to have different weights.

B2) **Where is the line deciding whether to use the document for certain analysis or filter it out?**

When the final quality score of a document is delivered, and it is not the simple 0 or 1, how do we recognize if the document is good enough? Research literature on existing IQ frameworks could provide some answers. However, FAQAD is not based on exactly the same dimensions as any of the existing frameworks as some of them are not available in our case. Therefore, practical testing and evaluation of FAQAD is required to answer this subsection.

1.4 Structure of the Thesis

The first aim of this thesis is to research existing IQ frameworks and available text evaluation tools in Java programming language. Secondly, based on the analysis of the collected information, we propose a new framework, FAQAD, for assessing the quality of documents and their sources. Finally, we implement the FAQAD framework in Java and evaluate its performance on realistic input data. The aim of the framework is to be able to assess the quality of documents and distinguish which are of high quality and which of low quality, hence, probably useless as sources of business information. Filtering out low quality document saves time for the business leaders. Being provided with information of higher quality, the business leaders potentially make better decisions based on the information. Additionally, preventing information systems such as DAVID from processing low quality documents saves computational resources and enhances the quality of the analysis results.

The thesis is organized as follows:

- In **Chapter 2**, the background of this work, the DAVID system and the role of the current work in it is described in more detail.
- In **Chapter 3**, a few existing IQ assessment frameworks are described and compared. The analysis of the existing assessment frameworks forms the basis to define a new QA framework which is one of the main goals of this thesis.
- In **Chapter 4**, we discuss what quality dimensions should be used and what freely available Java tools can be reused in this work. Thus, we define a new QA framework: FAQAD
- In **Chapter 5**, we report experiments in which FAQAD was used as a part of a larger software system. The aim of these experiments is to show that FAQAD is able to provide meaningful results from real data.
- The last chapter is dedicated to the conclusions and ideas for further improvements.

Chapter 2

Background

In this chapter, we provide an overview of the DAVID system and its components. First, we introduce the "Towards e-leadership" project (Section 2.1) and clarify the purpose of the DAVID system (Section 2.2). In Section 2.3, we outline the structure of DAVID and shortly describe every major component of the system. In Section 2.4, the process of fetching new documents is explained in more detail. QA tools that have already been discovered and partially integrated into DAVID are discussed in Section 2.5. Finally, in Section 2.6, we discuss the tools and mechanism used for the *graphical user interface* (GUI).

2.1 Towards E-leadership Project and DAVID

DAVID is developed as a part of a research project entitled "Towards e-leadership: Higher profitability through innovative management and leadership systems" which is a joint effort by School of Computing and Department of Business at the University of Eastern Finland. The research groups participating in the project focus on the scientific and educational aspects. In DAVID, various TM and NLP techniques are utilized in order to process text content of miscellaneous documents [40]. The main research question of the project is:

> "How to obtain, convert, and represent existing and invariably increasing information used in decision making in a way that enhances strategic leadership and reduces information overflow?"[40]

The project was funded by Finnish Funding Agency for Technology and Innovation (TEKES), European Regional Development Fund and seven partner companies that also contribute by business expertise and enable the software to be tested in a real enterprise environment. The participating companies are: Connexor (provider of language analysis tools) (http://www.connexor.com/), Futuremissions (a non-profit consultancy and management organization) (http://www.futuremissions.fi/), Johtamistaidon opisto (leadership and strategic management development institute) (http://www.jto.fi/), Metalliset Group (international contract supplier of metal parts) (http://www.metallilaite.fi/), Outotec Filters (leading company in designing and manufacturing industrial filters) (http://www.outotec.com/), Pohjois-Karjalan Osuuskauppa (retail chain) (http://www.s-kanava.fi/pko), and Valtra (tractor manufacturer) (http://www.valtra.fi/) [40].

2.2 Purpose of DAVID

Developing a TM system for collecting and analyzing BI was one of the main objectives of the project. The TM system analyzes text documents in order to help business leaders to reach the right decisions easier. It gathers and analyzes information from the internet, e.g. feedback, customer opinions, or BI to examine competitors. With the obtained results, it is able to assist the business leaders with making decisions [38].

The representation of information may vary. A basic numerical representation might be accurate. However, it might not always be useful and usable for the leader. Instead, a textual or a visual form can be more understandable. Additionally, the working environment in modern companies is constantly changing, and so is the information available. Thus, the data representation should dynamically capture those changes. The existence of dynamic representation and analysis aids to reach a proactive leadership. [40]

DAVID system is mainly used in the following manner:

- A business decision-maker **defines a project**. It has to be clear what is the intention of the analysis. The information sources, from which documents are gathered, need to be set.
- Once the project is running, new documents are **automatically fetched** from the internet sources.
- When the fetched documents are considered as being of **high enough quality**, they are further processed and several different TM techniques

(e.g. information extraction and sentiment analysis) are applied. These TM techniques are used for processing the input texts. To save the newly found pieces of information as well as storing the known facts, we need a domain-dependant knowledge base (i.e. *ontology*).

- Finally, as a result of the analysis, DAVID system provides **intelligence reports** about the business environment in textual format as well as in visual [38].

2.3 Structure of DAVID

To develop the whole DAVID system for analyzing textual BI from scratch would be an excessive amount of work. Creating and testing certain lower-level functionality such as converting documents from various formats or indexing them is indeed not the ultimate goal of the system. Moreover, developing such tools from scratch would be a too ambitious goal for a single research project.

Because of the large scope of the DAVID software project, it is crucial to reuse other software that is available. Using and integrating components that have already been implemented, tested, and used by other developers ensures that the components are reliable. Furthermore, system design that is based on reusing components allows us to spend more time on development of advanced analysis and decision-support capabilities instead of developing something that has been previously implemented [38].

There are many components with implementations of various web mining, TM and *Semantic Web* (SW) technologies freely available for Java programming language, and many of them are, in fact, open source [38]. The structure of DAVID system is shown in Figure 2.1. The scheme is explained in more detail below [40]:

Document Fetching

The document fetcher finds documents on the internet within specified sources and collects them. It is essential to be able to feed the system with new information. In Figure 2.1 on page 11, you can see it as component 1 in the top left corner. The external sources are not only web pages, but also news feeds and search engine queries. The fetched documents are converted to ASCII text

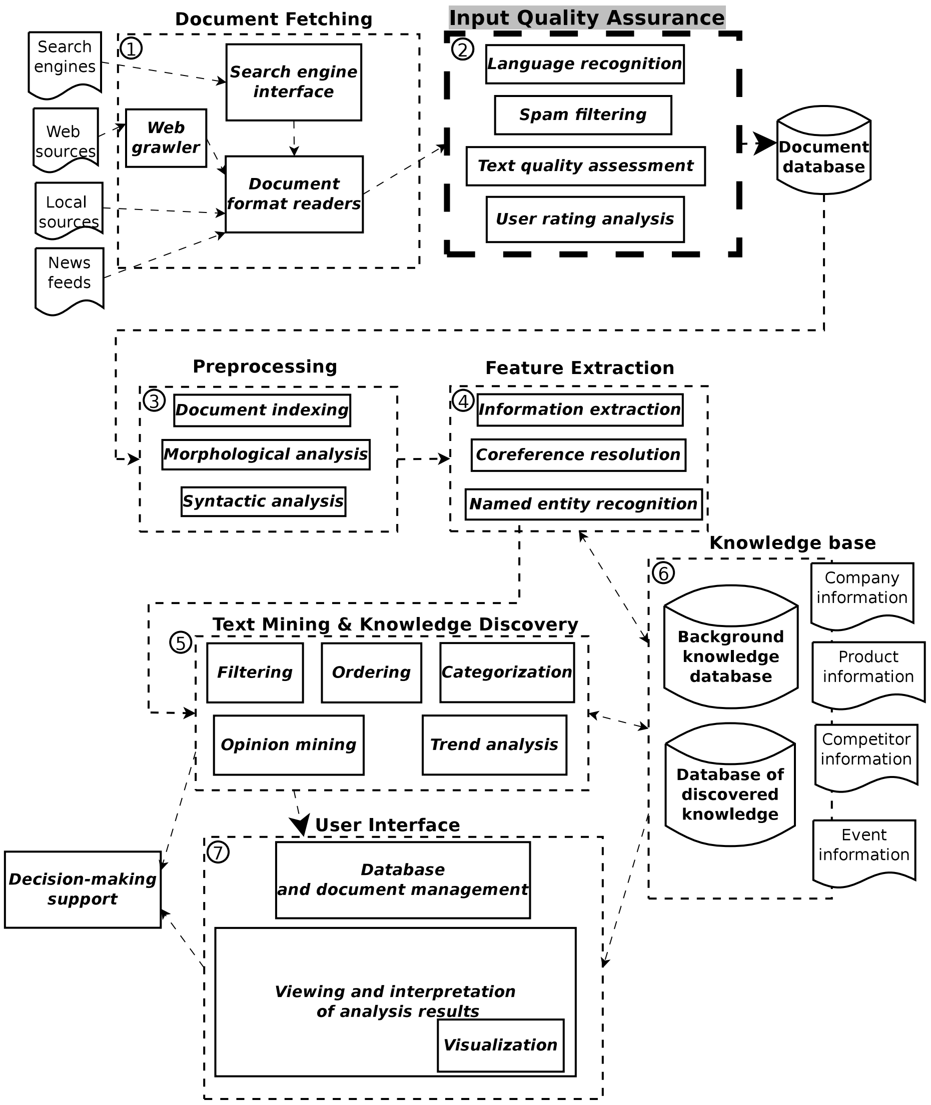


Figure 2.1: Architecture of DAVID system. Boxes with dashed lines show the main parts of DAVID marked with numbers in circle which consist from components shown by boxes with full lines.

from various file formats, e.g. HTML, PDF, MS Word, and PowerPoint. More information about fetching the documents can be found in Section 2.4.

An indivisible, yet important component of document fetching is the quality assessor. This component ensures that the input documents are worth further processing, either fully or partially. Processing all the fetched documents would be too resource-consuming. And many of them would be completely useless due to being either irrelevant or of poor quality. Developing a fine quality assessor framework and component to distinguish quality documents is the aim of this thesis. The quality assessor framework has its own chapter(4).

A list of the open source Java packages used in developing this component follows: BING API (`http://www.bing.com/developers`), Yahoo! Search API (`http://developer.yahoo.com/search/`), Heritrix web crawler (`http://crawler.archive.org/`), Web-Harvest (`http://web-harvest.sourceforge.net/`), YARFRAW (Yet Another RSS Feed Reader And Writer API) (`http://yarfraw.sourceforge.net/`)

Preprocessing and Feature Extraction

After a document passes the quality assurance component, which is the main focus of this thesis, and it is saved to a *database* (DB), it is further processed by components 3 & 4 in Figure 2.1. The document preprocessor examines the fetched documents linguistically (e.g. by part-of-speech tagging, morphological analysis, syntactic parsing) and decomposes documents into meaningful segments. The feature extraction components extract concepts (such as companies and products) and events (such as launching of a new product, bankruptcy of a company) by using the background knowledge base as the basis. This process is referred to as *ontology-based information extraction* (OBIE) and performed with using a purpose-built system called BEECON (*Business Events Extractor Component based on Ontology*) [17].

The preprocessing and feature extraction component uses a convenient open source software GATE (General Architecture for Text Engineering) (`http://gate.ac.uk/`). Additionally, the documents are indexed for efficient searching and retrieval. The name of Java component for indexing and searching is Lucene (`http://lucene.apache.org/core/index.html`).

Text Mining and Knowledge Discovery

In order to discover useful knowledge, this component, marked as component 5 in Figure 2.1, enables the extracted features to be processed in various ways, e.g. filtered, organized, categorized.

Knowledge Base

The knowledge base is used to store relevant background knowledge and also the new automatically discovered pieces of information about the companies and products included in documents that are being analyzed. The implementation of knowledge base is applying an ontology and semantic web technologies using Jena semantic web framework (`http://incubator.apache.org/jena/`). The framework offers the functionality to store, access and infer over the information contained in the knowledge base [59]. In Figure 2.1, knowledge base is marked with number 6.

Ontology

Ontologies are SW technologies that accommodate the resources to form concepts, properties, and relationships within a specific domain [38]. In the relation with DB systems, ontology can be seen as a level of abstraction of data models, analogous to hierarchical and relational models, but dedicated for modeling knowledge about individuals, their attributes, and their relationships to other individuals. Ontologies are considered to be at the "semantic" level. In contrast, DB schema are data models at the "logical" or "physical" level [49].

Ontologies in the field of computer science can be seen as dictionaries, categorization schemata, or modeling languages [27]. A specific ontology describes what is considered to exist in reality for a specific purpose. The ontology developed as component of DAVID is called *Company, Product and Event* (CoProE) ontology. Thus, CoProE deals with products and events concerning a certain company.

User Interface and Information Visualization

Users can easily use the system through a GUI. It provides the user capabilities to e.g. set up the system, to run analysis, and of course, to browse and search analysis results. The results can also be displayed in a graphical way in forms of graphs using JUNG (Java Universal Network/Graph Framework)(`http://jung.sourceforge.net/`). In Figure 2.1 on page 11, visualization component has number 7. The overall user interface (UI) is implemented using Eclipse RCP (`http://wiki.eclipse.org/Rich_Client_Platform`); more information about UI in Section 2.6 .

Support for Decision-making

The decision-making support module aims at using the information collected and analyzed by the other components of the system to support business decision making. The module combines the TM results with "traditional" competitive intelligence analysis models (such as the Five Forces Framework shown in Figure 2.2), to help leaders to track, understand and predict competitors' activities. The ultimate goal is to assist leaders to make smarter business decisions faster [22].

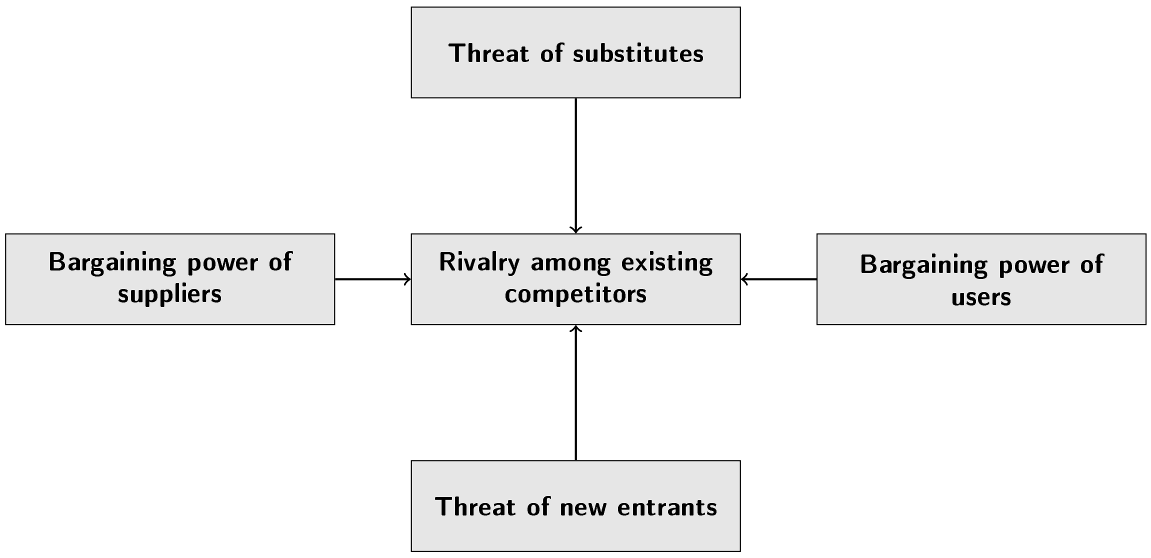

Threat of substitutes
Bargaining power of suppliers
Rivalry among existing competitors
Bargaining power of users
Threat of new entrants

Figure 2.2: Five Forces Framework[64] shows the forces that determine the competitive intensity and overall profitability of an industrial company

2.4 David Document Fetching Component

Fetching a document and preparing it for further processing consists of several steps as shown in Figure 2.3. The key component is *DocumentFetcher* which takes care of this process. The original fetcher component was developed by

Tuomo Kakkonen and Shukrat Nekbaev. Juho Heinonen had integrated some IQ assessment tools into DocumentFetcher before the current work on FAQAD was started [59, 36].

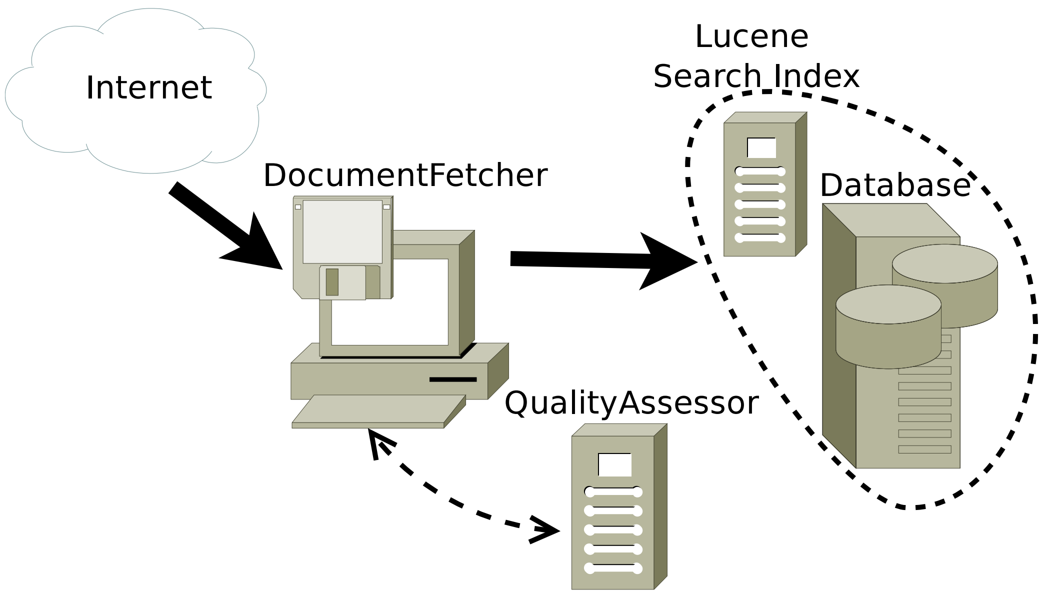


Figure 2.3: Fetching Process

As explained above, DAVID fetches documents automatically from internet data sources defined by the user. There are several types of data sources:

Web sites refer to HTML web pages. HTML pages mostly consists of text. Therefore, it is possible to process and extract information from them. Some web pages are made e.g. purely with Adobe Flash technology where the source is not available and it is not possible do analyze such documents. An example of a web page where BI can be obtained is `http://www.bbc.co.uk/news/business/`. There, we can find launches of new technologies, lawsuits and other information about competing companies.

News feeds refer to textual data format often used by content distributors on the internet where the content is frequently updated. A common example is *RDF Site Summary* (RSS) feed. Users can chose to subscribe to a desired news feed, and then download the news from the feed using a news reader. It might seem very similar to e-mail subscription. However, the news feeds have several advantages: users are not disclosing an e-mail address or any other personal information, therefore there is no threat of spam or viruses that could be regularly seen in e-mail inboxes.

`http://feeds.bbci.co.uk/news/business/rss.xml` is location of news feeds concerning business provided by British Broadcasting Corporation (BBC).

Search engine queries are the search phrases submitted to search engines by DAVID. The engines like Google and Yahoo! give for the same keywords different results over time. Currently DAVID supports Bing and Yahoo! via the Java API they provide.
A search query can be e.g. "valtra tractor". In a web search it would create an HTTP request for the server engine that you can usually see in a browser's address bar - it could look similar to `q=valtra+tractor`. The provided APIs accept the keywords as their input, so there is no need to manually create HTTP requests. The mentioned query finds web sites mentioning Valtra tractors.

Once a document is fetched, the text is extracted with *strippers*. Strippers are responsible for ripping ASCII text from various file formats, such as HTML, MS Word PDF, RTF and PowerPoint. The package uses freely available tools such as Apache POI (`http://poi.apache.org/`) and Apache PDFBox (`http://pdfbox.apache.org/`).

Extracting text from HTML is not as straightforward as from other formats. The process has 3 steps:

1. The class first strips the ASCII content of the whole HTML document (stripped text).
2. Then, it iterates through all the elements of the HTML document and check, by using certain heuristics, that each element contains "proper text" (i.e. full sentences) rather than garbage, such as ads and menus.
3. Finally, the elements deemed to contain garbage are removed from the stripped text.

2.4.1 Filters

Filters are applied to make sure that the input document fulfills all the criteria for a document that is considered valid by DAVID. Filters can be enabled or disabled by changing project settings. The filters need to access information

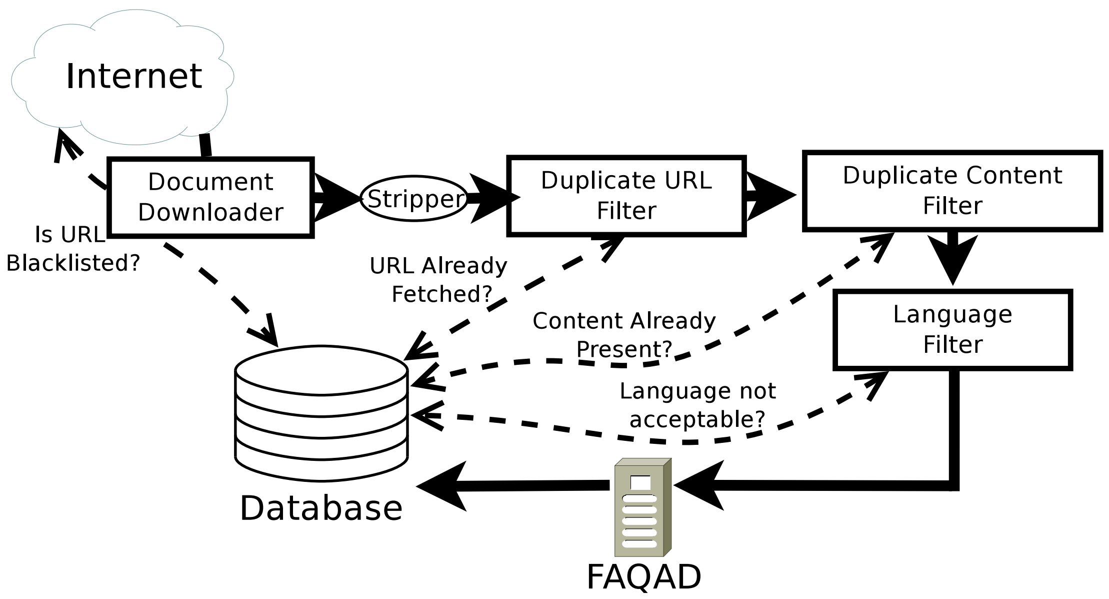


Figure 2.4: Filters applied on input documents. Each component can filter the new document out. Full arrows show the possible way of documents, striped arrows represent communication between components and the DB.

about the project settings as well as the access to previously fetched documents, i.e. the filters communicate with the DB. The document flow through the filters is visualized in Figure 2.4.

URL blacklist enables the user to block certain web pages or domains, thus preventing them from being processed by the system. URL blacklist filter is applied in the DocumentDownloader component [59]. For efficiency reasons, the filter is run before the document is actually fetched because the content of the document is not needed for this filter to work. For simplicity, that is not shown in Figure 2.3. Nevertheless, once the document passes the filter, it is downloaded, extracted to plain text using various stripper components, and it continues to the next filter.

Duplicate URL address filter filters out a new document if a document with the same URL address already exists in the system i.e. it prevents duplicates from being stored in the system.

Duplicate content filter filters out a new document if the content matches an existing document in the system, i.e. it does not allow content duplicates. Content filtering is based on Lucene [1] search index. Lucene is a full-featured text search engine library providing high-performance search capabilities over the fetched documents [1, 59]. Lucene is used for finding near matches. Once a near match is found, the two documents are compared in case-insensitive manner whether or not they are the same [59]. This prevents from duplicates located in different URL addresses to be stored in the system.

Language filter automatically detects the language of a new fetched document using a Java implementation of a library [5] developed for language recognition. A new document is either rejected or accepted based on the language settings of the current project [59]. The NLP and TM components of the DAVID system currently support only English, which means that the language filter is used at the moment for filtering out documents that are written in any other language.

FAQAD framework proposed in the thesis aims to bring these existing filters and new types of QA features into a unified QA framework. FAQAD will be implemented as a Java tool that allows a text document to be evaluated with several language processing tools. The DAVID system will then decide depending on the quality of a document, if it is filtered out or stored in the system for further analysis. The design and implementation of FAQAD are described in Chapter 4.

Once a fetched document passes all filters and is evaluated by FAQAD as of high quality, DocumentFetcher saves the document and the assessed quality to DB. Additionally, the document is indexed by Lucene.

2.5 Previous Quality Assessment Component of DAVID System

Juho Heinonen, student of linguistics, worked in the e-leadership project at the University of Eastern Finland during the year 2011 on finding document quality measurement tools to be used in DAVID. His work resulted in implementing a system that is capable of performing several types of linguistic measurements of document quality. These are listed in the following subsections.

2.5.1 Language

FAQAD evaluates documents in English and Finnish language. Because these two languages are from different language families and have no linguistic relation whatsoever, for certain analysis, different tools have to be used to evaluate documents written in these languages.

To distinguish what language a document is written in, DAVID uses *Java Text Categorizing Library* (JTCL). JTCL is a Java implementation of libTextCat which is a library that was created mainly for guessing the language of text documents. According to the web page [7], libTextCat performance is almost flawless in recognizing the language of text documents. JTCL was implemented at Knallgrau New Media Solutions, and at present time, it is used by tagthe.net which is web-service that can be used to provide tags for textual contents both on- and off-line [5].

2.5.2 Correctness

The frequency of misspelled words can be used as a measure of the correctness of a document. A high frequency of spelling mistakes indicates a lack of thought or diligent work from the author. We may argue that it is also possible that the document contains correct information but is not in author's native language. Nevertheless, in the context of DAVID, we do not consider a document or a web page as a reliable source of business information if it contains multiple spelling errors. As mentioned above, it is difficult to perform an accurate NLP and TM on documents that contain a high frequency of errors.

On the other hand, there are many spell checking tools available for common people. Therefore, it might happen that we have a document with no spelling mistakes that the common tools can discover, but the IQ is low. The spell checking tools are, for example, not able to find a misspelled word that appears to be another word spelled correctly (there$\neq$their). The weight and importance of spell checking in the overall IQ assessment is evaluated in Chapter 5.

Voikko

Voikko is an NLP tool for the Finnish language [14]. In addition to spell checking, it has ability to do other things as well: checking grammar, hyphenate words and collect related linguistic data for Finnish language [14]. In FAQAD, the spell checking module uses Voikko to tokenize Finnish documents and find spelling errors in it.

The Voikko libraries are programmed in C and C++ languages. FAQAD is implemented in Java. Fortunately, the developers of Voikko has provided a Java interface which makes it possible to use Voikko in Java applications. However, it means that compiled native libraries for different operation system need to be included in FAQAD. Currently, the libraries for MacOS, Windows (32-bit JVM), and Linux are included [36].

JMySpell

The MySpell spell checker under the LGPL license is the basis to JMySpell which is implemented in pure Java[6]. Using JMySpell, we can use the dictionaries from OpenOffice.org in Java applications. It does not matter whether they are J2EE web applications or J2SE applications. The module is able to check documents in both English and Finnish, even though the performance for Finnish documents is not good. It marks many composite words and inflected forms of words as misspelled [36]. FAQAD uses JMySpell to check spelling of English words. For FAQAD, it was utilized in a way that the component returns the ratio of

$$correctly\,spelled\,words/all\,words\,in\,document$$

To check texts in Finnish language by JMySpell, it is probably not the best choice. However, it is used for Finnish texts as a second choice if Voikko (see above) is not supported by the operation system [36].

2.5.3 Readability and Understandability

Readability and understandability are values indicating how pleasant a text is to read. To obtain those values, there are several tools we can use to evaluate text for readability and lexical diversity. Finnish texts tend to show high lexical diversity, because of many suffixes Finnish words can gain. In order to get more realistic values, we use the package Snowball to create stems of the words. Using the stems instead of the inflected words makes it possible to utilize standard readability and lexical diversity measurement to texts written in Finnish.

Snowball

Snowball is a string processing system that was designed for creating stemmers used in information retrieval [11]. It supports several languages including Finnish. Because Finnish words tend to have many different suffixes in written text, to get more realistic result about lexical diversity, Snowball is used. Otherwise, all the forms of the same word would be considered as different words. Therefore, the test would give excellent scores for lexical diversity analysis [36].

2.5.4 Spam Detection

We consider spam as unwanted bulks messages, such as product advertisements or phishing e-mail. These documents do not contain any reliable information and we can think of as “trash that accidentally got on our table”. Most probably, everybody who uses e-mail has seen some kind of spam and possibly even a spam filter. Because the spam has no information value whatsoever, it is required to use a spam filter in order to prevent DAVID from spam overload and misleading information.

Classifier4J

As the name suggests, Classifier4J is a text classifier for Java, i.e. it is implemented in Java. The system uses a Bayesian classifier [2]. A naive Bayesian classifier is based on Bayes' theorem. It is called *naive*, because it considers all the features to be independent. The assumption of independence makes the classification much easier. However, it seems to work well in practice even when the independence assumption is not genuine[12]. A more clear and understandable term for the essential probability model could be *independent feature model* [9].

In a more understandable way, the naive Bayes classifier expects that the presence (or absence) of a certain feature is not related to the presence (or absence) of another features. For example, a fruit could be recognized as an orange if it is orange, round, and about 5cm in radius. Although these features are related, or could be related to other existing features, naive Bayes classifier assumes that all the features are independent, and contribute separately to the probability that the fruit is an orange[9].

Naive Bayes classifier works in two steps to be able classify data[12]:

1. Training - by using training samples, the probability distribution of different features is estimated.

2. Prediction - new test samples are classified using the calculated probability based on the training data. It is so called *posterior probability*.

Classifier4J provides tools to save training results, and make classification decisions based upon the training[36]. However, Classifier4J has two major drawbacks:

- It is not able to add new training results to the one currently saved. All training data have to be used at once.

- It does not give a value about how certain it is that a text matches a category. It simply returns 0.01 or 0.99. On the other hand, it makes the implementation of FAQAD easier.

2.6 Graphical User Interface

An inevitable part of every sophisticated software is the GUI. Today, no end-users do really want to use *command-line interface* (CLI), although it might be faster in some cases. CLI demands careful reading of some kind of manual. GUI is easier to understand because the user visually sees what options he/she has, i.e. GUI makes operations more intuitive.

Users can access and work with the DAVID system using a GUI. Hence, the QA framework also needs to have its own GUI. Therefore, we review the current GUI of DAVID and discuss the technologies and tools used for implementing it.

Nowadays, there are multiple options to choose from when selecting the platform to implement a GUI. Web-based interface is widely used. You can easily access the system remotely and you don't need any extra desktop client to access the system. Since the whole DAVID system is developed in Java programming language, a web-based GUI would need an extra framework to communicate with the system and display the data. Again, because the whole system is developed in Java, the easiest solution would be to implement the user interface in Java. The main advantage of Java is that it is platform independent, unless you use any platform specific components.

Java has two standard GUI tool kits:

Abstract Windows Toolkit (AWT) is the original Java GUI tool kit. The main advantage o AWT is that it is available in every common version of Java Technology - that means it is also included in Java implementation in very old or obsolete web browsers, and it is stable. That means you do not have to install anything further, you can just rely on any Java runtime environment, and it will support your AWT application with all the features you expect. However as the original toolkit, the amount of AWT's GUI components is very limited. Components, such as Tables or Trees are not supported. In application where you need more components, you have to implement them from scratch. That might become a problem [28].

Swing also known as a part of the *Java Foundation Classes* (JFC), was an effort to resolve most of the AWT's drawbacks. Nevertheless at the same time, Swing is built on parts of AWT. All the Swing components are also AWT components. "In Swing, Sun created a very well-engineered, flexible, powerful GUI tool kit. Unfortunately, this means Swing takes time to learn, and it is sometimes too complex for common situations."[28]

With these GUI toolkit, there is still lot of programming to do, especially when you want to use components such as wizards or editors, because those components are not implicitly available. Fortunately, *Rich Client Platforms* (RCP) for Java exist, so we do not have to implement every single widget we need. For DAVID system and the QA API, Eclipse RCP was chosen.

2.6.1 Eclipse Rich Client Platform

Eclipse platform is an open source platform that provides many components that the developers can use and benefit from the tested features of the framework. Thus, they do not have to implement everything from scratch using the basic Java GUI tool kits such as AWT or Swing. Eclipse platform is designed in a way that using its components, we can be build simply any client application [10].

Eclipse and Eclipse applications are built using a plug-in architecture. Plug-ins are software components, and they are the smallest deployable components of Eclipse [72]. The essential collection of plug-ins required to build a rich client application is commonly known as RCP[10]. Of course, rich client applications are able to use and be extended by third party software or API to enhance their functionality [72].

Eclipse RCP is the basis for Eclipse - one of the most successful Java IDE. It uses native GUI widgets to provide native look and feel as much as possible. It allows us to relatively quickly build a professionally looking application for multiple platforms. With its intense modularity approach, we can conveniently design component based systems [72].

Many companies including corporations like IBM and Google use the Eclipse platform frequently for their products. Thus they ensure, that Eclipse is fast, flexible and continues to evolve [72]. Eclipse RCP is stable and broadly used and allows the developers to use the Eclipse platform to create flexible and extensible desktop applications [72]. It also allows them to easily reuse and integrate components that are already implemented.

Chapter 3

Quality Assessment Frameworks

Over the past few decades, several frameworks have been developed for assessing IQ in text documents. The focus of these frameworks has been, in particular, on the QA of web pages. According to Strong et al.[69] high quality data is data that is fit for use by the data consumer. The quality or usefulness of data is dependent on the individual who is going to be using it. Good quality data would therefore meet requirements of its intended use. The concept of quality is therefore relative, depending on the different perceptions and needs of the users of the data[62].

In the following Section (3.1), we discuss the different types and categories of quality dimensions. In Section 3.2 , we compare and discuss the existing QA frameworks.

3.1 Categorization

ISO defines *quality* as "the totality of characteristics of an entity that bear in its ability to satisfy stated and implied needs" (ISO 8402, 1994). In context of web pages, the definition implies we need two different approaches and kinds of requirements for web document quality evaluation[32]:

1. *Technical requirements*: These deal with the structure of web documents. This category is concerned with technical design aspects, thus takes into consideration criteria which indicate objective and quantitative characteristics of the documents. That includes web page code quality, broken links, but also structure of document in sense of clear order of information.

2. *Content requirements*: These consider the extent to which the web documents meet the specific user needs. The evaluation criteria in this category takes into consideration subjective and qualitative characteristics of documents. That includes, e.g., accuracy, relevance, consistency.

IQ assessment frameworks are defined using a series of quality dimensions. In order to compare different approaches, the quality dimensions can be grouped into four categories. The following categorization schema was introduced by Wang and Strong [73].

Intrinsic Dimensions are independent of user's context. Intrinsic dimensions indicate that a piece of data possesses quality in its own right, i.e. the data have objective attributes and are not affected by user's needs for a particular task. The common intrinsic sub-dimensions are briefly explained in Table 3.1 on page 27.

Contextual Dimensions are based on user's context and subjective preferences. The quality of data is considered within the context of the task user needs to accomplish. Because the context and tasks are changing over the time, it is quite a challenge for researchers to measure the contextual quality dimensions accurately with fixed assessment methods[70]. User's subjective preferences indicate what makes an information of high quality, i.e. which quality dimensions are the most significant for the particular user and the user's task at hand. The frequently used contextual sub-dimensions are briefly described in Table 3.2.

Representational Dimensions are concerned with representation of information within *information systems* (IS). Representational dimensions consider aspects regarding the format of the data as well as the meaning of the data. Thus, the IS must present the data in consistent, interpretable and easy to understand manner. Sub-dimensions of this category are briefly explained in Table 3.3.

Accessibility Dimensions consider aspects involved in accessing information. This category emphasizes the role of IS, i.e. the IS must be accessible and at the same time secure. Nowadays, users mostly access the internet for their information needs and are not looking for a hard-copy version so often anymore. Thus, accessibility dimensions need to be considered as inseparable part of IQ. The common accessibility sub-dimensions are described in Table 3.4.

Each of the quality dimensions listed above can be further divided into sub-dimensions [19]. The different quality sub-dimensions are briefly explained in the tables underneath:

Table 3.1: Intrinsic Dimensions

Sub-dimensions	Description
Accuracy	Is the degree to which the information content of a web page is correct and reliable[62]. In fact, many people consider accuracy to be the same as quality. Nevertheless, accuracy is only a single component of quality[71]. Information, whether electronic or on paper, is a representation of real world objects or events. Data elements hold values that are facts representing some attribute of a real world object or event. Therefore, accuracy is the extent to which data properly matches the actual object or event being explained [24].
Consistency	Indicates that values in a document do not conflict with each other. Information on web-sites might be perceived as inconsistent, since they have been created by multiple authors that might have different level of knowledge and different perception of reality [19].
Objectivity	Is the extent to which the information is unbiased, not prejudiced and is fair so no missing fact would significantly change the meaning of the information [63]. The objectivity of certain types of information, such as product description, could be affected by the information provider's interests or goals [19]. Objectivity is closely related to the accuracy sub-dimension.
Timeliness	Is the degree to which information is up-to-date for the activity intended [37]. Timeliness can be recognized in an objective fashion, meaning that information reflects the current state of the real world [57]. At the same time, timeliness can also be recognized as task-dependent, meaning that the information is timely enough to be used for a specific task [63].

Table 3.2: Contextual Dimensions

Sub-dimensions	Description
Believability	Degree to which is the content on a web page true and trustworthy [63].
Completeness	Degree to which are information in the content not missing, and the depth of information is adequate [63]. Because we are talking about contextual dimensions, the perception of completeness of a certain information may differ between users. For example, list of students might be complete for a professor giving lectures, while the list is incomplete for the head of the department.
Understandability	Degree to which is the data smoothly apprehended by the end user [19]. Understandability is somewhat related to interpretability. However, interpretability refers to technical aspects, such as usage of appropriate notations, while understandability refers to the subjective capability of the user to perceive the information.
Relevancy	Degree to which information is appropriate and beneficial for the task at hand [63]. It is an essential IQ dimension in the context of web-based systems and search engines, as end-users are frequently challenged with large amounts of possibly relevant information [19]. Search engines assess relevance in order to sort results accordingly.
Reputation	Quality sub-dimension that measures trustworthiness and significance of a source. The content of a web page gets user's attention because of information the user gathered previously from that web page [62].
Verifiability	Is the degree and comfort to which the information on a web page can be easily verified for correctness [57].
Amount of Data	In the context of task at hand, "amount of data" is the degree to which the quantity of information is suitable and the user does not get astonished by too detailed information [19].

Table 3.3: Representational Dimensions

Sub-dimensions	Description
Interpretability	Is the degree to which information in a document is presented in appropriate language, using relevant units and symbols. Of course, also the definitions need to be clear [63]. The availability of key material to endorse correct interpretation, such as summaries, figures, guides, etc., are crucial. Interpretability is an essential component of quality as it allows the information to be appropriately utilized and understood.
Representation	Is the degree to which information is represented in the same format [63]. In general, the representation of information on the web is not very consistent, because there are not any restrictions for the representation [34]. An example of inconsistency in representation is to use different document formats, such as HTML, PDF, and Microsoft Word, within a single web page [19].

Table 3.4: Accessibility Dimensions

Sub-dimensions	Description
Accessibility	Refers to availability of the information or how quickly and simply it is to fetch the document [63]. The main factor of the success of World Wide Web is the possibility to supply numerous information sources with an on-line access. Enhancing accessibility of the information on-line is the primary motivation behind the technologies standardization of the web [19].
Response Time	Measures the time interval between a user's request sent to the server and the response obtained from server. The response time might be affected by various factors, such as complexity of the request, network traffic, or the server's workload [19].
Security	Is the degree to which access to information is adequately limited to keep it protected [63].

3.2 Comparison

Comparison of IQ frameworks is shown in 3.5[62].

Table 3.5: Comparison of IQ frameworks[62]

	Information Quality Dimensions	Zeist & Hend-ricks (1996) [75]	Strong et al (1997) [69]	Alexan-der & Tate (1999) [15]	Kater-attana-kul et al (1999) [41]	Shanks & Corbitt (1999) [67]	Nau-mann & Rolker (2000) [58]	Zhu & Gauch (2000) [76]	Dedeke (2000) [23]	Leung (2001) [48]	Kahn et al (2002) [37]	Eppler & Muenz-enma-yer (2002) [26]	Klein (2002) [45]	Liu & Huang (2005) [50]	FREQUENCY
Intrinsic	Accuracy	X	X	X	X	X	X		X	X		X	X	X	11
	Consistency		X			X			X		X	X		X	6
	Free-of-error			X	X				X	X	X	X			6
	Objectivity		X	X		X	X	X			X	X	X		8
	Timeliness	X	X	X		X	X	X	X	X	X	X	X	X	12
Contextual	Appopriateness	X	X	X					X		X	X	X		7
	Believability		X	X	X	X	X	X			X	X			8
	Completeness		X			X	X		X		X	X	X	X	8
	Ease of manipulation	X							X	X		X			6
	Relevancy	X	X	X	X		X	X	X	X			X	X	10
	Reputation		X	X	X	X								X	5
	Verifiability														0
	Understandability	X	X				X		X	X	X	X			7
	Amount of Data														0
Rep.	Interpretability														0
	Representation		X	X	X		X				X	X		X	7
Acc.	Accessibility	X	X	X	X	X	X	X	X	X	X	X		X	12
	Security	X	X				X			X		X			5
	Source			X	X		X	X			X		X	X	7
	Value-added		X				X								2

The analysis of the information quality frameworks in Table 3.5 reveals common dimensions between the existing IQ frameworks. The most frequent quality dimensions used in those frameworks are: accessibility, accuracy, relevancy and timeliness. The reason for this is that different researchers considered them to be most useful and relevant ones.

Accessibility dimension addresses technical accessibility, and the problem with accessibility is realized quickly by every user. Unlike other quality dimensions, users are able to notice that accessibility is poor even before they start reading the document. Additionally, when a user knows a document with certain information exist, but it is not possible to access it at the moment[62], it might be even more agitating for the user than spending lot of time time by looking for the information. Consequently, poor accessibility may lead to bad reputation of the web page.

Accuracy is probably one of the most important quality dimensions for the majority of users when searching information, because inaccurate data are mostly useless and potentially misleading. Lack of accuracy may, again, lead to poor reputation and also to believability problems[62]. Ultimately, inaccurate information is useless or harmful and should not be used as a basis for decision-making.

Relevance is a task-specific quality dimension. When users seek information, they usually use search engines in order to locate the information on the web. Because of the enormous quantity of documents on the internet, search engines sort the search result according to relevance or popularity[34]. In this sense, relevance is the resemblance between the search key words and the text in the documents that were returned. If the search engine does not find relevant documents for user's task, the user has to try to search with different or more specific keywords. In many cases, the user does not eventually find what he was looking for. In contrast to the dimensions mentioned above, not-finding relevant documents usually does not lead to poor reputation of web pages, but rather the search engine.

IQ is commonly perceived as the fitness for usage of the information[19]. According to this definition, the IQ is task-dependent and subjective. Although, the intrinsic dimensions indicate that data posses quality of objective nature, it is hardly enough for any user to evaluate documents without any context. IQ is a concept of multiple dimensions. Which dimensions are important and which quality levels are needed is resolved by the task at hand and the subjective preferences of the user.

Chapter 4

Developing FAQAD – New Framework for Quality Assessment

In the following section (4.1), we discuss the measurement of some of the mentioned IQ assessment dimensions introduced in Chapter 3. Our focus is on the dimensions that are important in the context of a BI system such as DAVID, however at the same time, it is very difficult to assess their quality within our settings. In Section 4.2, new QA tools and components are introduced. Implemetation details, such as the DB structure (4.3.3) or tools used for development, are described in Section 4.3.

4.1 Measurement

The assessment of contextual dimensions, as mentioned before, is based on the user's context and subjective preferences. FAQAD does not have a straightforward way to communicate with a user to find out his or her preferences, thus, it cannot really work with contextual quality dimensions.

For example, **relevance** is one of the contextual dimensions. Relevance ranking is used by search engines to estimated what is user is looking for. The average size of a web search query is two terms[54]. Obviously, such a short query cannot specify precisely the information search of web users, and as a result, the response set is large and therefore potentially useless (imagine getting a list of a million documents from a web search engine in random order). One may argue that users have to make their queries specific enough to get a small set of all relevant documents, but this is impractical. The solution is to rank documents in the response set by relevance to the query and present to the user an ordered list with the top-ranking documents first. Therefore, additional information about terms is needed, such as counts, positions, and other context information[54]. DAVID is able to access data via search engine queries which return result sets ordered by relevance ranking. Therefore, FAQAD obtains documents that are, according to the search engine, the most relevant for the used keywords. However, FAQAD does not have access to the actual ranking values of the search engine. That implies that relevance, as in user context, cannot be directly used by FAQAD for assessing the overall quality of documents.

Accessibility could be measured using criteria such as amount of broken links, orphan pages, code quality, or navigation on a web page, i.e. visual structure of the document. FAQAD obtains the documents from DAVID's DocumentFetcher component as plain text along with the URL address from which the document was obtained; an internet connection is needed to be able to measure accessibility. Nevertheless, in case a web server has a long response time, and FAQAD needs to process large amount of documents from that web server, time consumption would increase enormously. Additionally, at the moment, FAQAD itself does not use direct internet connections, as these services are provided by DAVID. Therefore, the current system design does not provide means for measuring accessibility. Instead, DAVID skips a document after a preset time has elapsed from the moment the attempt to access the document started. This prevents the document downloader from getting into a deadlock.

Accuracy, in the sense of correctness and reliability of the texts that have been fetched is the main focus of the FAQAD framework. Reliability is obtained by ranking each document source based on the quality of the documents that have been previously retrieved from it. More information about this technique is provided in Subsection 4.2.3.

4.2 Designing FAQAD

In addition to the QA tools introduced in Section 2.5, FAQAD includes ones for measuring the readability of document (4.2.1), the quality of data sources (4.2.3), user rating mechanism (4.2.4), and a spam filter (4.2.2).

4.2.1 Readability

Lexical diversity and readability give an approximate value about the overall linguistic quality of a document. Lexical diversity measures the size of vocabulary used in a document. Lexically diverse text, i.e. one with a richer collection of different words, is usually considered to be more convincing about its content than an low diverse equivalent of the same text; more commonly used words tend to be shorter than the words that are used, for instance, in science or by specialists in some specific field[39]. Readability implies how easy the text is to read. Length of words is a significant factor in evaluating readability[36].

We use two packages, TexComp and Fathom, to evaluate readability and understandability.

TexComp

TexComp is a component that analyzes texts and calculates readability and lexical diversity values. TexComp can be adjusted to better suit for the analysis of different languages. In Tuomo Kakkonen's article[39], it is stated that TexComp was tested on two different corpora:

1. English speaker students in the Department of English, Uppsala University, Sweden [74]

2. Native English speaker students from Oxford Brookes, Reading and Warwick University [33]

The evaluation results by Kakkonen indicated that the system can be reliably used for assessing the readability and lexical diversity of the texts in the two test sets. Native English speakers were given higher scores for lexical diversity and readability on average than non-native speakers.

Fathom Package

Fathom package includes three reading level algorithms that can be helpful in determining the readability of the content [47]. George Klare (1963) defines readability as "the ease of understanding or comprehension due to the style of writing."[44] However, we cannot assume that good readability of a content always means it is easy to understand. As explained above, documents that contain a relatively high number of long words are potentially more exact, specific and reliable. On the other hand, they tend to be difficult to read for common people.

Reading level algorithms only provide a rough guide to measure readability, as they tend to reward short sentences made up of short words. Fathom package works only with English texts, and it is basically a Java implementation of two Perl packages [61]: Lingua::EN::Fathom by Kim Ryan and Lingua::EN::Syllable by Greg Fast [3].

Gunning-Fog Index roughly indicates how many years of schooling it would take somebody to conceive the text content [31]. The larger the index score is, the more sophisticated the text is. Gunning-Fog Index was developed by Robert Gunning to help the writers and newspaper editors to write to their audience by removing the "fog" - unnecessary complexity that most common people do not understand. The algorithm uses the average quantity of words per sentence and the ratio of complex words in the text to calculate the score:

$$(Words\, per\, sentence + Percentage\, of\, complex\, words) * 0.4$$

In this algorithm, the complex words are considered to be words with more than two syllables.

Flesch Reading Ease outputs an index score that evaluates the text on a 100-point scale [29]. The higher the score, the easier the document is to understand. The best score range is considered to be approximately from 60 to 70. The author, Rudolph Flesch, proposed his formula to improve writing styles. Several US institutions use the Flesch Reading Ease test as a standard tool to validate readability of forms and documents [30, 65]. Similar to Gunning-Fog Index, this algorithm uses the average quantity of words per sentence and the average amount of syllables per word for the calculation:

$$206.835 - (1.015 * Words\, per\, sentence) - (84.6 * Syllables\, per\, word)$$

Flesch-Kincaid grade level is based on Flesch Reading Ease but gives a different score. Flesch-Kincaid grade level is similar to Gunning-Fog index in a sense it is a rough measure of how many years of schooling it would take someone to understand the text content. It was developed by J. Peter Kincaid and his team for the US Navy [43]. Later, it was used by the US army, for example, to assess readability of technical manuals. For the calculation, the algorithm uses the already mentioned statistics about the text:

$$(11.8 * Syllables\, per\, word) + (0.39 * Words\, per\, sentence) - 15.59$$

4.2.2 Spam Filter

Most people consider spam to be electronic junk mail [35]. It arguably is the most widely used form of spam. Spam mostly consist of product advertisements that seem to most persons suspicions, to say the very least. For example, it could be one of the get-rich-quickly-and-easily scheme, or it could be drugs, e.g., to enlarge different parts of the body or to loose weight effortlessly. Unfortunately, spam is not used only in e-mails, but also in other media, such as on-line forums, blogs, classified advertisements, Wikipedia and web search engine results.

The ultimate goal of spam is to get money from the users. If the advertisement is not directly asking the user to buy something or “make deposit”, it asks for more contact information, so the user can be bothered more then just by e-mail. However, not all the spam is used to rob users of money. Instead, some spam infects posts with ideas and opinions such as religion views.

It is quite obvious that spam does not offer valuable BI, and it appears when user is looking for something else, i.e. usually, there is not any relevance between the spam and the user needs. Spam only consumes user’s precious time and does not give any benefit in return. In the DAVID system, spam has a similar impact: consumes time for nothing in return. In the worst case, introducing spam into the analysis processes of the system causes erroneous analysis. Therefore, spam needs to be filtered out.

ABCV API

Anti-social behavior, conflict and violence (ABCV) is an ontology and tagging tool for detecting various types of misbehavior and conflicts from text. It is developed by Dr. Tuomo Kakkonen in the context of the “Detecting and visualizing changes in emotions in texts” project (`http://cs.joensuu.fi/~mmunez/emotion_detection/index.html`) that is funded by the Academy of Finland. Among other things, the Java tool that is based on the ontology is able to detect swearing words and profanations. ABCV API outputs only values 1 or 0 to determine whether or not the text contains socially inappropriate content, and if it

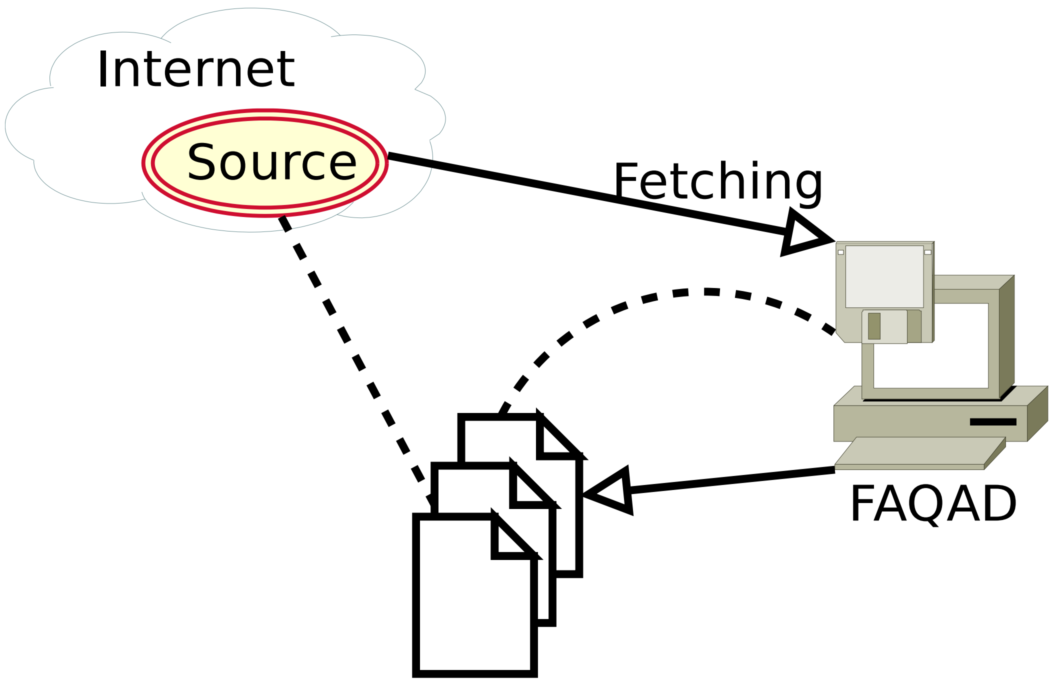


Figure 4.1: Quality of Data Sources

should be rejected. We assume that a document including swearing words does not contain reliable business information. On the other hand, swearing words in a user feedback indicate anger and disagreement, especially from users with extrovert personalities [53]. Those contributions should not be ignored, because it points out a potential negative attitude towards a product or company. If the DAVID system is applied on customer feedback data, ABCV-based filtering should be disabled by using the appropriate parameter setting.

4.2.3 Quality of Data Sources

Evaluating the content of single documents is not the only way how to distinguish high and low quality of documents. We can also assess the quality of the source from where the document was downloaded. Quality of data source can be estimated based on the average of the quality of documents that were previously fetched from that source. The process is shown on Figure 4.1.

1. Once we have documents saved in the DB with the assessed quality, we can calculate quality of the documents' data source.

2. When a new document is fetched by DocumentFetcher, it is evaluated by FAQAD.
3. Additionally, we can use the quality of the source where the new document is fetched from to decide if we want to accept the document or not.

4.2.4 User Rating

Evaluating documents by automated tools makes life easier for the user and can provide a wealth of useful QA data. Nevertheless, including a human opinion is very important. We may assume that an evaluation done by human is correct in most of the cases. At least, it is correct in more cases than the automated tools are able to be. Similar to quality of data source used to partially evaluate newly fetched documents, user rating can also be used for this purpose.

Therefore, users are given the option to evaluate a document by themselves. User ratings are values that indicate how useful the document is in the users' perspective. Mainly, there are two ways in which the users could report their opinion on how good a document is to the system:

Star Rating is a widely used method e.g. on web pages. Users are offered the possibility to select stars to indicate, for example, how much they liked the document or how useful it was. In case the user considers the extremely useful, he selects the maximum number of stars. When the document is not useful at all, the user selects no star. Using this approach, it might be difficult to establish the mean value. If the user thinks the document is neither good or bad, what value does he/she select? It could get even more complicated when there are more users evaluating the documents, and each of them has a different idea about neutral value.

Scale is an alternative to the star rating approach that alleviates some of the issues related to start ratings. Scale is usually implemented by widget called scale or slider which has a minimal and a maximal value. The minimal value on the left side of the scale is for very bad documents, and the maximum value is for very good documents. The mean value is simply in the middle.

The star rating seems to be more popular than the scale rating [16]. This is mainly due to the fact that users have gotten used to it since it has been used on many on-line services, such as on-line auction sites and pools. As mentioned before, the main advantage of the scale compared to the star rating is that we know exactly where the neutral value is. That is the major attribute of the scale that guided us to choose it as the user rating method to be used in FAQAD. However, the rating scale is a replaceable GUI component, hence it could be easily exchanged for another type of rating component in the future.

4.3 Implementation Details

4.3.1 Java

The entire DAVID system, with the exception of a few external libraries, is developed in Java programming language [59, 36]. In the following text, we discuss a few features of Java that makes Java different from other programming languages such as C/C++.

Java Virtual Machine

The major advantage of Java is that we write one code and the application works and has the same functionality on any supported platform. Compiling a Java source code results in having a so called "byte code". To be able to run the byte code (Java application), we need Java *Virtual Machine* (VM) present in an operation system. Java VM interprets the byte code and runs it on the system. Unlike C and C++, in which the code is compiled for the target platform, Java byte code can be run on any system for which a Java VM is available. [68].

Another fact about C and C++ is, that the primitive data types, such as integer or float, have different sizes on different systems[68]. Having enough space for computing is crucial in large calculations or simulations. Therefore, the C/C++ source code has to be changed for different platforms. Java does not face this problem. However, the Java VM is continuously updated, and it could face other problems with each update such as a security breach [60].

FAQAD uses multiple external packages and libraries. Using only one programming language for implementing a relatively complex software system has many benefits: it makes the system development simpler and allows for any member of the project team that knows Java to participate in implementing any part of the system. Additionally, using Java for FAQAD and also DAVID makes migrating to another operating system easier, because there is no need to rewrite the source code or recompile it for the target platform.

Java Exceptions

Java implemented a convenient way to handle errors. In older programming languages, we have to check what every method or function returns in order to discover an error (for example, trying to open file that does not exist). In Java environment, methods are able to throw exceptions. “An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.”[13] When an error takes place within a method, the method creates an *exception object* (EO) and hands it over to the run-time environment. The EO contains various information, such as type of the error and state of the program when the error occurred. After the run-time environment retrieves an EO, it tries to find a block of code to handle the exception which might be located in another part of the program. Using this approach, it is easy to separate the program and error-handling logics. The following pseudo-code demonstrates how error-handling works in Java [13]:

```
readFile {
    try {//any of the methods within try clause
         //may throw exception
        open the file;
        determine its size;
        allocate that much memory;
        read the file into memory;
        close the file;
    //catch clauses handle exceptions
    } catch (fileOpenFailed) {
       handleException;
    } catch (sizeDeterminationFailed) {
        handleException;
    } catch (memoryAllocationFailed) {
        handleException;
    } catch (readFailed) {
        handleException;
    } catch (fileCloseFailed) {
        handleException;
    }
}
```

It is important to note, however, that exceptions do not, in any way, free one of detecting and handling errors. It does, however, help to organize the code in a more effective way.

For example, FAQAD uses a DB connection to save or update certain data. Exception handling makes the programming easier, because it allows us to write a code that makes sure that everything runs smoothly even if an error occurs.

In order to make sure that FAQAD does not crash due to DB errors, the code that uses the DB connection is placed inside a try-catch block. We do not need to check the return values of methods that are called within the *try* block for a possible DB error. In case an error occurs, e.g. the DB is not available, an exception is thrown, and we can handle that exception in a *catch* block. This way, the error handling is separate from the normal run-time code. That helps to keep the source code clear and more organized.

Garbage Collector

Garbage collector (GC) is a form of automatic memory manager within the Java run-time environment. Once an object in a Java program is no longer used, the GC finds it and frees the unused memory space by destroying and removing the object from the memory. From the programmer's point of view, it makes work much easier since the programmer does not have to remove all the unused objects manually[1].

The main argument and drawback of the traditional GC is that it consumes computing resources in order find out what is considered "garbage" and needs to be removed. The program must pause for the GC to reclaim any unused memory. Users usually do not notice the pause since it is often just a fraction of a second, however, it is unacceptable for real-time systems[18].

FAQAD also takes advantage of the GC. FAQAD is processing a fetched document and creates object containing information about the document and its various QA scores. Those information are processed and saved to a DB. From that moment, the created object is not needed anymore nor used by FAQAD in any way. GC destroys the object to ensure there is enough memory for new documents going to be fetched.

[1]In C/C++, manual memory management was often source of the hardest-to-find bugs[68]. Nowadays, there are implementations of GC for C/C++ programs as well[20]. However, not all the C/C++ programmers actually use it.

4.3.2 Eclipse IDE

Eclipse *Integrated Development Environment* (IDE)[4] was chosen to develop DAVID. Java code can be written in a basic text editor, however, Eclipse IDE has many features that makes the programming easier especially for large projects. These features include:

Syntax highlighting to recognize e.g. what are variables and what Java language keywords. That helps read and manipulate the code easier and faster.

Code assistance comes in handy when a developer does not remember an exact name of a certain method or parameters it accepts. Eclipse is able to offer a list of methods for a certain object type. That list usually includes documentation for those methods. Therefore, you do not necessarily have check the manual every time you are not sure about a method. Also, when you need to check the code of a method used in your code, Eclipse allows you to display implementation of that method by a simple click.

Code validation on-the-fly helps to detect common mistakes and typographical errors. That way, one can easily correct the code before you try to compile the code. In most cases, Eclipse also offers a list of possible solutions.

Concurrent Versions System (CVS) support is very convenient to use especially in projects with more participants. Programmers save their code to a CVS server with relevant comments about the changes that were made, and others can easily download the code, see the changes, and possibly change or extend the code. In case there are any errors accidentally saved to the CVS server, we can retrieve an older version of the code. CVS support is integrated in Eclipse, so there is no need to have another external application. And because there are several participants working on DAVID project, CVS is a good choice how to make programming easier for everyone.

JUnit is quite a simple Java open source framework originally written by Erich Gamma and Kent Beck based on the xUnit architecture for unit testing [42]. JUnit is designed to create and execute a set of tests in order to make sure that the developed Java application works as expected. Additionally, JUnit can be used as a tool to manipulate testing data, such as import or export them from DB. Nevertheless, the main features of JUnit include:

- Automated Execution of a set of tests
- Assertions to test expected results
- Test Fixtures to share and possibly slightly modify testing data for each test
- Test Runners to be able to run the tests in various ways such as from IDE or CLI

FAQAD uses JUnit in order to verify that QA tools are working and not failing. Additionally, JUnit is used by FAQAD to automatically process and evaluate large amounts of documents and save the assessment scores into a DB.

4.3.3 Extending DAVID Database

The storage component of the DAVID system is responsible for saving information allowing to gather and analyze BI from specific data sources on the internet. All the settings needed for such a process are encapsulated within a project. Users can define several projects for different purposes. The DB structure is shown in Figure 4.2.

When user sets up a project, it is necessary to save the following data in order to be able to gather and analyze information from the internet:

Information needs determine what types of competitive intelligence is DAVID going to gather and analyze[59]. They could be: customer opinions, suppliers, subcontractors, or competitors.

Data Source is a source of input documents defined by the user (see Section 2.4). There are three types of data sources: web sites, RSS news feeds, and search engine query. A data source contains required parameters, such as URL of the source.

Once the project is set up and running, the systems starts to fetch new documents from the data sources specified in the project. In the DB, there are two tables for storing documents:

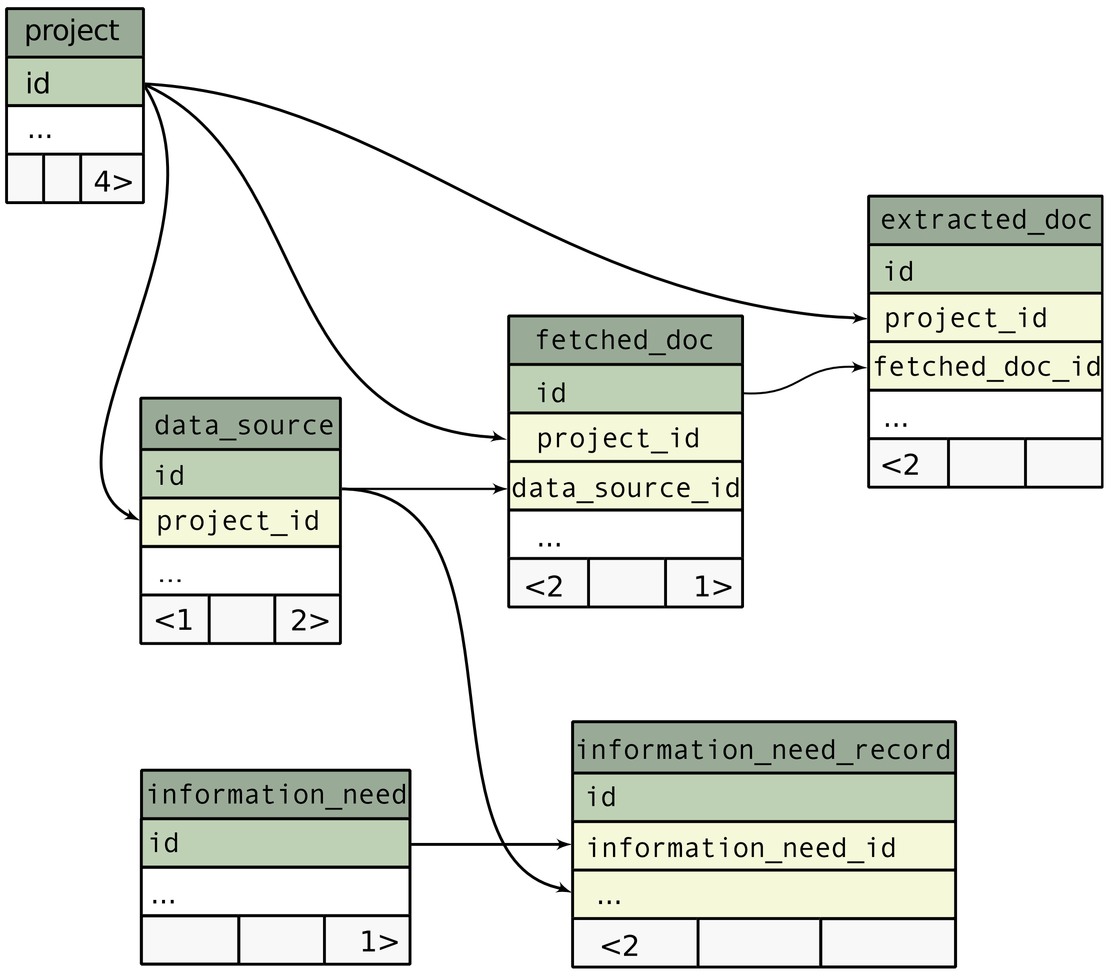


Figure 4.2: Part of the DAVID DB scheme relevant to document fetching. The arrows indicate parent-child relationship where parent can have zero or more children, and each child has exactly one parent.

Fetched Document contains documents that have been fetched from the internet and saved to the document storage after filters have been applied; i.e. have not been filtered out.

Extracted Document contains documents that were previously fetched, and selected for further processing.

The DAVID data storage consists of a MySQL DB [8] which holds all the saved data and Lucene search index[1] for efficient search capabilities over the fetched and extracted documents[59].

To ensure that the DAVID system is working with documents that are worth processing, FAQAD extends the DB to be able to store information about data sources and quality of the fetched documents. At the moment, the only additional information about the sources which FAQAD uses is which data sources are blocked and are avoided when fetching new documents. Once new documents are fetched, FAQAD evaluates the documents' quality and saves those scores into DB for later usage.

4.3.3.1 Blocked Sources

A blocked source defines a data source from which DAVID must not fetch any documents. It is convenient to define such a source in order to prevent fetching and overwhelming the system with large amounts of potentially useless documents. To specify which sources are blocked within a project is optional. An example of a blocked source in the BI context would be an on-line auction site. While these pages contain numerous mentions of brand and product names, they typically do not offer any valuable information for business decision making.

As you can see on Figure4.2, there could be more than one way how to save the information about which sources are blocked. We could, for example, extend the table `data_source`, and indicate which source is blocked. This table holds much more information about the sources. However, information such as fetching details are useless when the source becomes blocked.

The other solution could be to create a new table indicating which sources are blocked within the project. This way the DB storage is used efficiently.

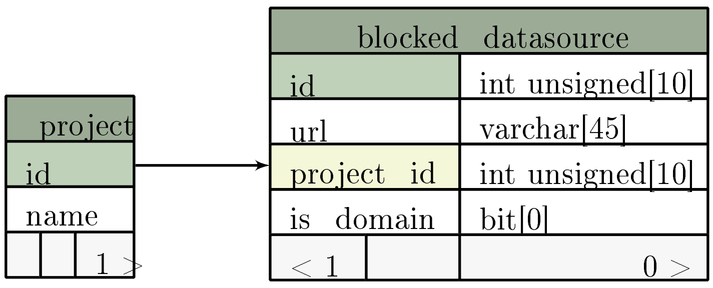


Figure 4.3: Blocked Sources within a project.

4.3.3.2 Blocked Sources Presets

Adding blocked sources to project settings could mean a lot of typing for the user. To avoid such an unnecessary time consumption, DAVID provides the users presets of blocked sources they can use. Various sources are grouped to different categories to make the manipulation easier. The groups can be, for example, auction sites or companies selling a certain type of products. The presets are not directly related to any data shown in the DAVID DB (Figure 4.2). At the moment, the blocked sources presets are stored in a separate DB shown on Figure 4.4. Naturally, it could be moved to the DAVID DB to make the DB schema more compact and easier to maintain.

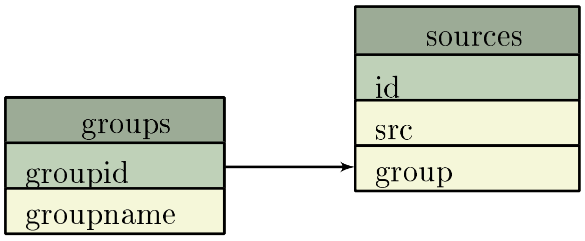


Figure 4.4: Database of blocked sources presets. The arrows indicate parent-child relationship where parent can have zero or more children, and each child has exactly one parent.

Each source in the DB belongs to a certain group and is specified by an URL. The user can choose items or the entire groups from the blocked sources presets through the implemented GUI dialog. A snapshot of the dialog is shown in Figure 4.7. Additionally, users can add or modify the existing presets of blocked sources using a GUI dialog shown in Figure 4.8.

4.3.3.3 FAQAD

Once a document passes all the filters (2.4.1), it is saved into the `fetched_doc` DB table. At the same time, QA scores of the document are saved to the DB by FAQAD. The original DAVID DB did not have a way of storing information on the quality of fetched documents. As a part of developing FAQAD, such a table, `fetched_doc_scores`, was added (Figure 4.5). The table is able to store all the QA scores for each document.

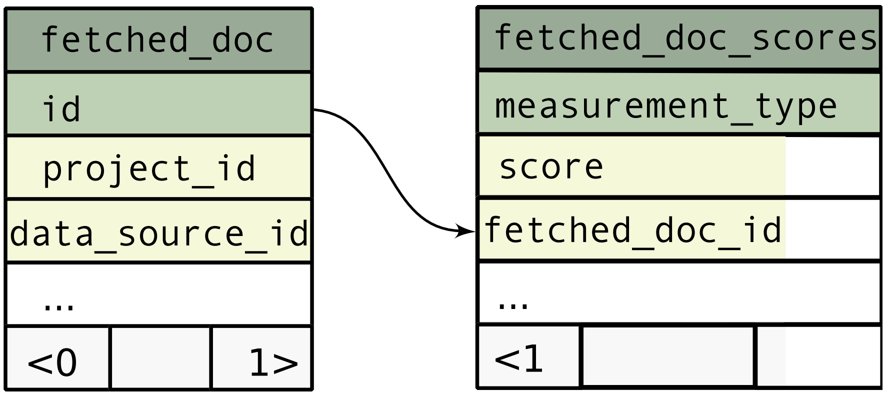


Figure 4.5: DB table for saving the assessment scores. Each combination of `fetched_doc_id` and `measurement_type` is unique.

Based on the multiple quality scores provided by its components, FAQAD has to provide a single QA score and make a decision whether or not to allow the document to be stored for further processing.

FAQAD has to make a single decision. The decision-making process is demonstrated in Figure 4.6 as FAQAD needs to determine whether the document qualifies for further processing or it should be filtered out.

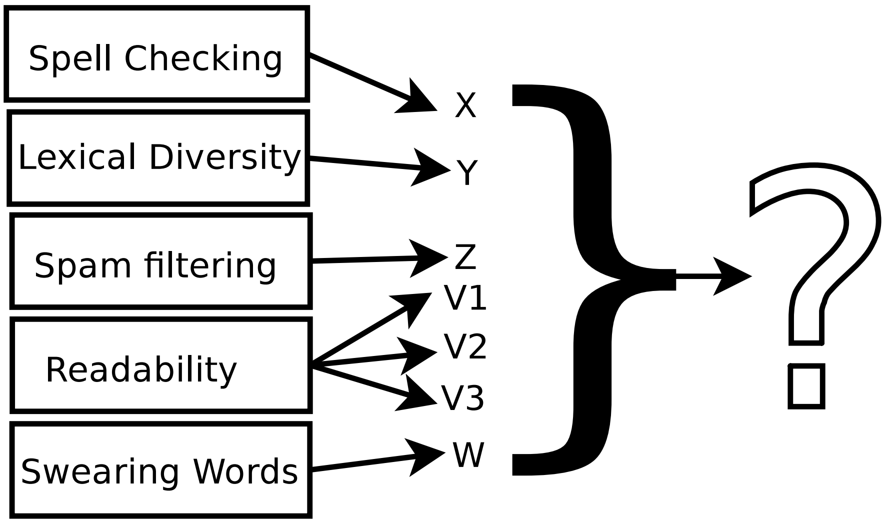


Figure 4.6: Make a single filtering decision upon various quality dimensions.

If all the QA results indicate that a document qualifies, there is no conflict, and we can assume that a document is of high quality. However, if the various scores contradict with each other, we need to designate a formula to calculate the final decision. Although the easiest method to combine numeric values is arithmetic mean, some tools are more reliable than others, which means their scores are more important than the scores from the less reliable tools. For this purpose, we can use weighted mean where each score has a designated weight. The formula (4.1) is shown below.

$$\bar{s} = \frac{s_1 w_1 + s_2 w_2 + \cdots + s_n w_n}{w_1 + w_2 + \cdots + w_n} \tag{4.1}$$

Each weight $(w_1 \ldots w_n)$ is assigned to a score $(s_1 \ldots s_n)$. We can simplify the formula by normalizing the weights, so that the fraction denominator is equal to one, i.e. $\sum_{i=1}^{n} w_i = 1$. The simplified formula (4.2) is shown below. We will pair specific weights with scores in the following chapter.

$$\bar{s} = s_1 w_1 + s_2 w_2 + \cdots + s_n w_n \tag{4.2}$$

4.3.4 User Interface

FAQAD uses Eclipse RCP to relatively easily build a robust GUI, thus enable the user to set certain project options.

Selecting Sources to Be Blocked

User is able to select sources which will be ignored when the DAVID system is fetching new documents, i.e. documents from the selected sources will not be fetched. The preset of the blocked sources is saved in the DAVID DB. The GUI dialog that users can use is shown in Figure 4.7.

Editing Blocked Sources Presets

In order to use the blocked sources presets effectively, we need the possibility to modify them according to users' needs. Again, this is implemented through a GUI dialog. The GUI dialog is demonstrated in a snapshot in Figure 4.8. In the dialog, the user is able to:

- Add new items
- Edit items
- Filter the items in case there are too many of them
- Delete multiple items at once

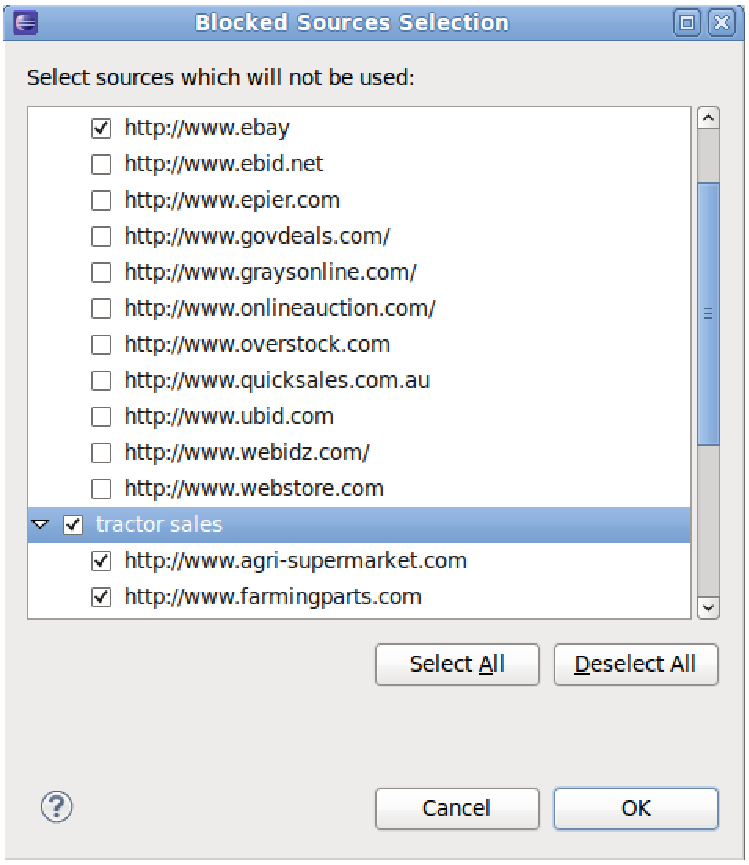
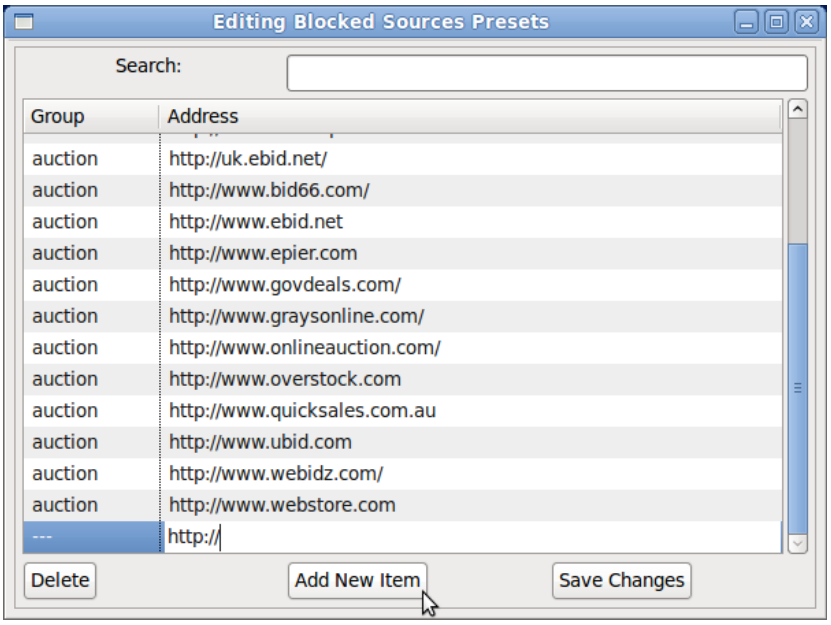

Blocked Sources Selection
Select sources which will not be used:
☑ http://www.ebay
☐ http://www.ebid.net
☐ http://www.epier.com
☐ http://www.govdeals.com/
☐ http://www.graysonline.com/
☐ http://www.onlineauction.com/
☐ http://www.overstock.com
☐ http://www.quicksales.com.au
☐ http://www.ubid.com
☐ http://www.webidz.com/
☐ http://www.webstore.com
☑ tractor sales
☑ http://www.agri-supermarket.com
☑ http://www.farmingparts.com
Select All
Deselect All
Cancel
OK

Figure 4.7: Dialog for selection of blocked sources from the presets.



Group	Address
auction	http://uk.ebid.net/
auction	http://www.bid66.com/
auction	http://www.ebid.net
auction	http://www.epier.com
auction	http://www.govdeals.com/
auction	http://www.graysonline.com/
auction	http://www.onlineauction.com/
auction	http://www.overstock.com
auction	http://www.quicksales.com.au
auction	http://www.ubid.com
auction	http://www.webidz.com/
auction	http://www.webstore.com
---	http://

Figure 4.8: Dialog to edit the blocked sources presets.

Chapter 5

Experiments

As described in Chapter 4, we have gathered numerous tools to evaluate text content of documents. In order to decide which tools to use in the QA component and how to weigh them in the overall QA, we need to analyze their performance on realistic data. The assessment tools that have been chosen for the evaluation, output varying types of values that reflect certain properties of the quality of the document. The returned values vary both in type and usage. For instance, a value 27 returned from the Flesch-Kincaid tool in Fathom package has a very different meaning than e.g. accuracy in percentage from JMySpell. The value can be a decimal number, a number between 0 and 100 or even a number in a totally different and unusual range. We need to evaluate and analyze the outputs of each of the tools separately in order to make a decision whether or not to include it in the FAQAD toolbox.

In order to get meaningful results, we need to feed the QA tools with real-world data. We have four major data sets that are further described in the following section (5.1). In order to gain a better understanding of how each of the tools work, we ran the tests multiple times after modifying the data sets. In Section 5.2, we describe the approach for running the series of tests and analyzing the results. The results of the experiments are shown and analyzed in Section 5.3.

5.1 Test Data Collections

Here, we describe the four data collections that are used to run all the information QA tools in order to evaluate the scores they return. A summary of the test data collections is located in Table 5.1.

Business Articles were originally gathered for testing other components developed in the e-leader project. The dataset has been previously used for developing and testing the BEECON tool (Arendarenko and Kakkonen, 2012 [17]) and the text categorization component (Machunik, 2012 [52]). The articles were gathered manually by five participants of the e-leadership project from the news portals on the internet. These news portals include popular news services, such as REUTERS (`http://www.reuters.com/`), The New York Times (`http://www.nytimes.com/`), YAHOO! Finance (`http://finance.yahoo.com/news`). The articles were manually cleaned from advertisements and other irrelevant information. The total amount of the business articles in the dataset is 840.

Flames is a collection of short texts containing vulgarities and profanations. One part of this collection was provided by Dr. Tuomo Kakkonen. The other part was manually collected from various web sites and forums on the internet by the author of this thesis, Radim Svoboda. The main purpose of this collection is to test the ABCV API (Subsection 4.2.2). The collection contains 106 entries.

Leipzig Corpora Collection presents monolingual dictionaries [46]. The main idea is to use the dictionaries for testing the language recognition tool. The text collections downloaded from the Leipzig Corpora Collection web site (`http://corpora.informatik.uni-leipzig.de/download.html`) are gathered mostly from newspapers and random web sites in various languages. All the collections are in a same format and similar in size and content. The texts are divided into single sentences; the non-sentences and parts of texts containing foreign language were removed. The size of the collections vary from thousand sentences up to hundreds of thousands. In these experiments, we used a subset of the collection that consisted of 10,000 sentences per language.

Spam data set was downloaded from a spam archive (`http://untroubled.org/spam/`). The author of the site has been collecting spam e-mails since 1998. Due to the large volume of the spam data, we used only a small portion of the spam archive. The e-mails contain e-mail headers which indicate where had the e-mail traveled from in more detail. For our purpose, we removed the headers from the emails in the collection and only used the message bodies in the evaluation. After excluding the duplicate entries, the collection contains 1834 spam messages.

Table 5.1: Summary of test data collections.

Collection	Quantity	Description
Business Articles	804	articles from news portals on the internet
Flames	106	texts containing vulgarities and profanations
Leipzig Corpora	70,000	monolingual dictionaries - single sentences
Spam	1,834	portion of online e-mail spam archive

5.2 Test Settings

Running the test data manually through the QA tools would take considerable effort and a lot of time. Nevertheless, the tools are implemented in Java and while being integrated, they were tested by JUnit [42] to check if they are working as expected. The JUnit tests that were used for checking the tools were also used for automatically running the experiments. Large amounts of test data might aggravate the ability to browse and to view the input data and the QA results. In order to enhance the ability to analyze the outputs, we saved all the data and the evaluation results into a new MySQL DB that was set up exclusively for the evaluation purposes. Storing all the data in a DB has several advantages: easier manipulation in sense of showing e.g. minimal, maximal and average values of the scores or the text length, batch altering or generating new data based on the previous, etc.. Additionally, we can generate graphs for visualizing the results and possibly their correlation with any programming language or tool that is able to access the MySQL DB.

In our case, we chose the programming language *PHP: Hypertext Preprocessor* (PHP) for handling the evaluation results. With PHP we can easily manipulate and load the values, and are able to generate and preview generated *Scalable Vector Graphics* (SVG) images[1].

5.3 Results

In the following subsections, we give and discuss the evaluation scores for each QA tool. All of the QA tools are designed for evaluating specific languages - in

[1]SVG defines the displayed objects in a similar manner as in analytic geometry, i.e. objects have position, size, and orientation. Unlike bitmap graphics, SVG can be easily resized without any visible corruption of the image. This is very handy especially when we generate the images or figures for electronic publications where the readers often zoom in or out.

our case English. In order to be able to use the tools, we need to test the language recognition component (JTCL) first.

5.3.1 Language Recognition

We used the Leipzig Corpora Collection for testing the language recognition component introduced in Subsection 2.5.1. In the first test, each sentence was evaluated separately. This method of evaluation is, in fact, more demanding than the use case in which the tool is used as part of DAVID. In DAVID, language recognition is always done at the level of whole documents. The longer the text, the easier it is to correctly recognize the language. JTCL was tested on the major world languages that use Latin alphabet, and the results are shown in Figure 5.1. Additionally, the English text collections are divided based on the region they come from, e.g. Australia, United Kingdom, USA, etc.

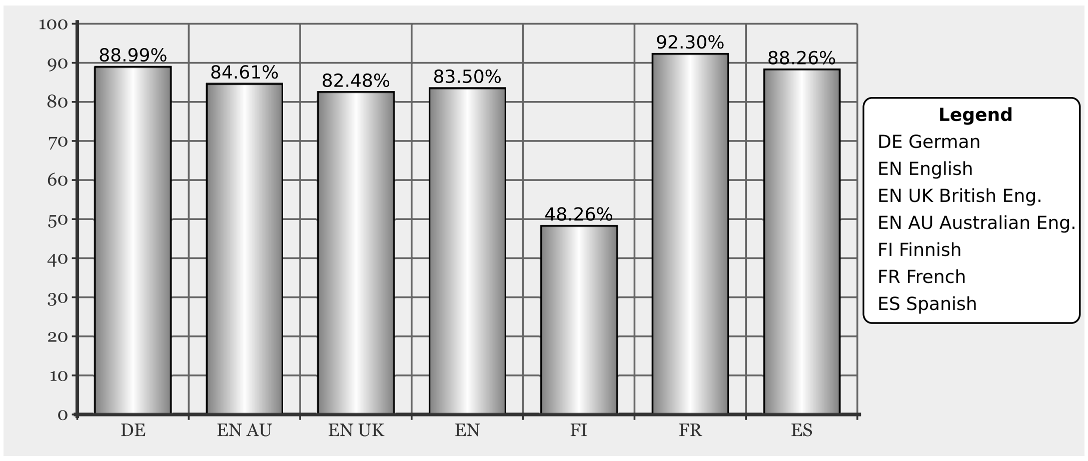


Figure 5.1: Accuracy of language recognition for seven languages (in percentages).

One can observe from Figure 5.1 that the tool had the worst accuracy in recognizing Finnish. It is because it is a difficult language, and Finnish words tend to change based on their usage, e.g. with suffixes. To recognize a language with such properties is not an easy task. On the other hand, English words change minimally. However, somewhat surprisingly, English texts do not have the best results.

Looking beyond recognition accuracy percentages, one may pose the question: what happens if a sentence is not correctly categorized? Is JTCL not able to associate the text with any language? Or is the text associated with a wrong language? Actually, both cases have occured in our experiments. The statistics in Table 5.2 indicate that the ability of JTCL is strongly affected by the amount of words that are being evaluated.

Language	**Correctly Recognized (%)/AWC***	**Wrongly Recognized (%)/AWC***	**Not Recognized (%)/AWC***
German	89.0/15.4	0.1/5.5	11.0/11.2
English	83.5/20.8	0.2/6.5	16.3/15.7
Australian English	84.6/19.5	0.5/7.3	14.9/12.4
British English	82.5/20.0	0.5/7.0	17.0/13.7
Finnish	48.3/13.1	0.0/–	51.7/11.4
French	92.3/20.6	0.2/8.1	7.5/13.4
Spanish	88.3/18.5	0.5/5.6	11.2/7.8

Table 5.2: Analysis of errors per language. ***AWC** stands for average amount of words per sentence being evaluated.

In Table 5.2, we can see that the absence of high enough number of words often results in wrong language recognition. This is indicated by the fact that the average amount of words per sentence where the language was not recognized is always less than the average amount of words in correctly categorized texts. In less then 0.6% cases, JTCL failed and associated text with incorrect language. These cases are outlined in Figure 5.2.

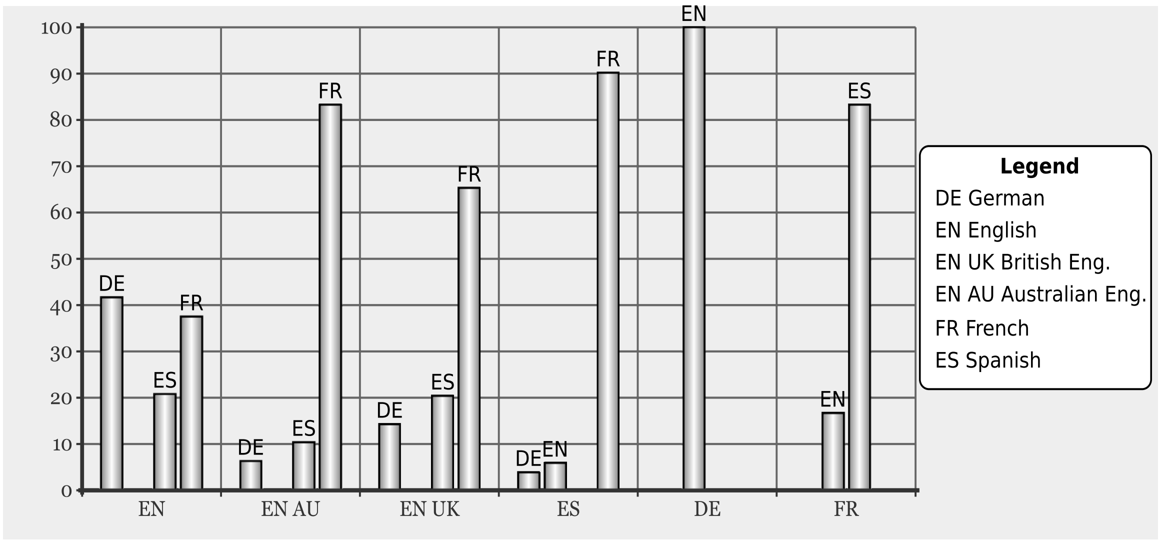


Figure 5.2: Incorrectly categorized languages and which languages were the texts associated with (in percentages).

As indicated by the results reported above, an incorrect identification of language appeared rarely. Nevertheless, we must not ignore them, and it would be worth to figure out the reason why they occurred. Let's look at some examples. The following sentences were categorized as English texts:

> "Le grand public est invité."
>
> "IT wird nie permanent funktionieren!"

At the first sight, we can see that these languages are not English - they are French and German. However, these sentences are quite short, and they contain words that can be easily found in an English dictionary - words such as "grand", "public", "it" or "permanent". In a similar manner, the following English sentence was categorized as French:

> "Contact us on 3290 7600."

All three words can be found in a French dictionary. Paradoxically, an English dictionary might not contain the word "us", because it is just another form of the word "we". The sentences where the language was not recognized have a related issue. More specifically, some words are just not found in any language dictionary. The sentences usually contain names, abbreviations, or quite a few non-alphabetic characters such as stars, brackets, colons or numbers. For example:

> "ALIA Core values statement (ALIA website)"
>
> "Posted by Pomerz on March 21, 2003 at 16:14:11"
>
> "EXO-3C GXO-U100 GXO-U101 GXO-U102 GXO-U103 GXO-U105 XO1 XO1H XO1HV XO1L XO1V denotes our 'Key Products', especially selected for value & availability."

All the sentences above are quite short, and JTCL most probably did not find enough key words to match the correct language. It makes sense that the right words could possibly be found easier in longer sentences. Figure 5.3 confirms this assumption. To make the figure more clear, all the regional types of English are combined together.

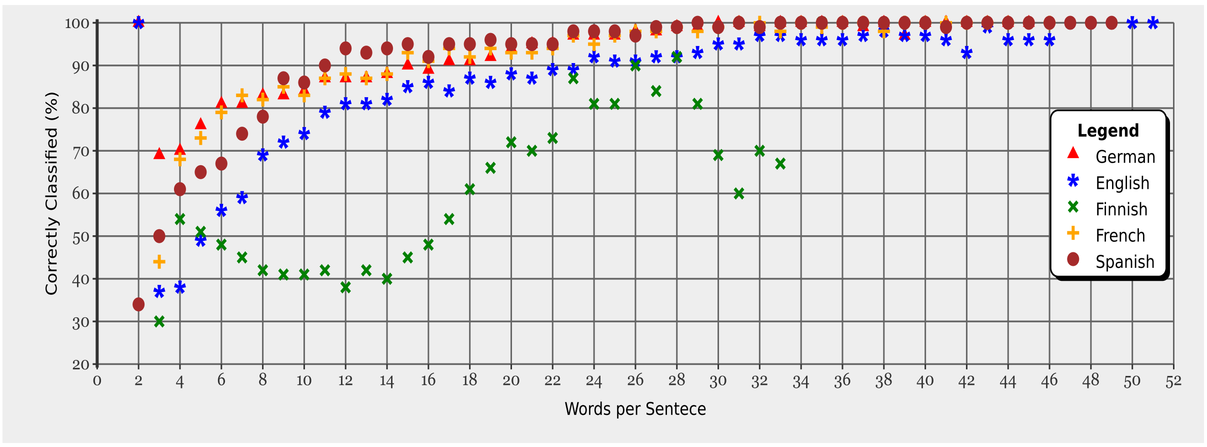


Figure 5.3: Percentage of correctly classified sentences for each language.

To make sure that the hypothesis about the effects of the length of input sentences is valid, we conducted an experiment. Let's create paragraphs only from sentences that were not correctly categorized and evaluate them. Previously, JTCL was not able to correctly recognize any of them. The sentences were chosen randomly, and they were concatenated in order to form paragraphs of at least 50 words.

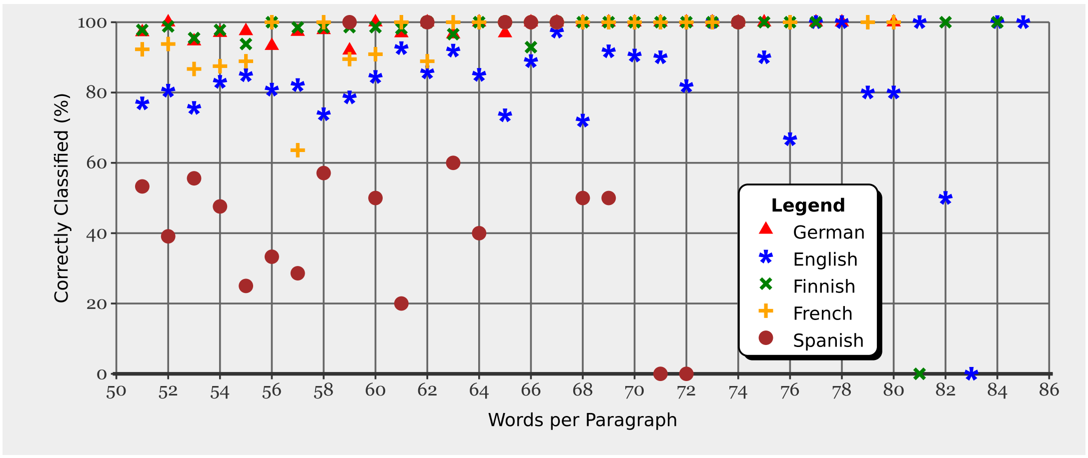


Figure 5.4: Percentage of correctly classified sentences for each language. Only the sentences previously not correctly classified were used. The mean averages of successful recognition are: German: 97.3%, English: 82.7%, Finish: 97.9%, French: 92.0%, Spanish: 49.1%.

As we can see in Figure 5.4, there is a significant improvement. The improvement of Finnish language recognition is outstanding. If we could ensure that the JTCL component is fed only with blocks of text that are long enough and possibly contain just few non-alphabetic characters, we would posses a remarkable and powerful tool for language recognition.

Business Articles

To confirm our conclusion, we tested JTCL using the business articles. Unlike the test items in Leipzig Corpora Collection, the business articles consist of more than one sentence. The articles contain higher amount of words from 24 up to 1899 with the average of 333.71 words per article. JTCL was able to assess that 838 out of 840 (**99.76%**) articles are in English. The two unrecognized business articles contain quite a few dashes, slashes and abbreviations with the word count of 24 and 53. Based on these results, we can conclude that JTCL is an accurate enough tool for detecting the language in the target domain.

5.3.2 JMySpell

JMySpell, introduced in Subsection 2.5.2, is used to check the spelling of English words using an English dictionary. If a word is misspelled, but at the same time it exists in the dictionary, JMySpell or any other standard spell-checking component is not able to recognize that as a mistake (e.g. two≠too), i.e. spell-checkers are not capable of recognizing grammatical errors. JMySpell checks each and every word, and according to the dictionary, it records how many words are correct and how many are misspelled. To get an overall result about the document, JMySpell was utilized in order to return a ratio of $correct\,words/all\,words$.

In the previous experiment (Subsection 5.3.1), the results indicated in the figures were grouped by the amount of words that were contained in the tested texts. It was very useful, as we could see significant differences in short and long texts results. We use the same grouping in the figures below.

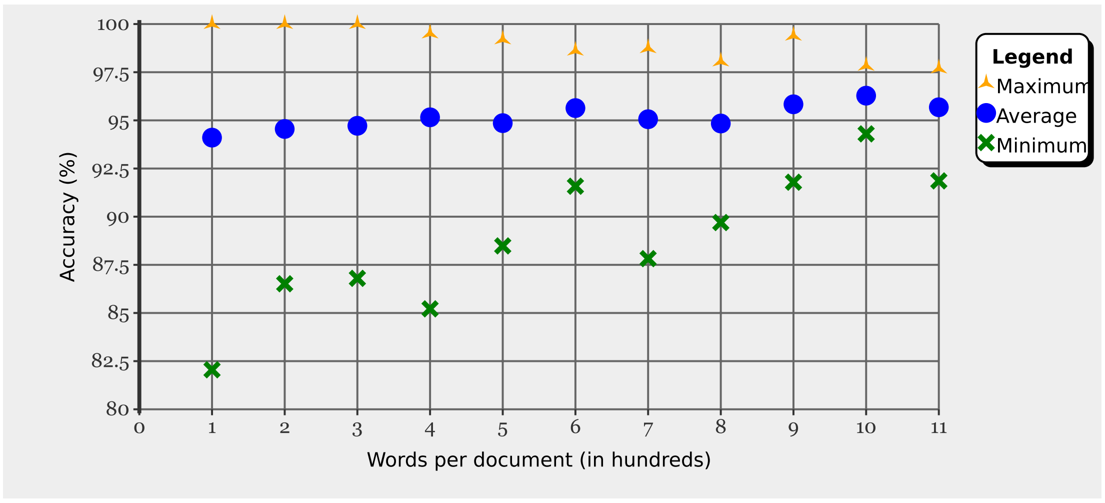


Figure 5.5: Spell-checking of business articles. Showing the minimal, maximal, and average values of the test results.

The average accuracy of misspellings in the business articles data set is 94,86%. At the same time, the worst result was 82,05%. After analyzing the worst results, it was obvious that the problem was usage of abbreviations, money amounts, and proper names, but also spelling variations, such as *online* instead of *on-line*, which are not recognized by JMySpell using OpenOffice dictionary. For example:

> "Inter-Alliance in four-for-one bonus
>
> By Reuters
>
> Last updated at 12:00 AM on 19th September 2000
>
> Independent financial adviser Inter-Alliance said it was proposing a four-for-one bonus issue. Shares in the AIM-listed group closed at 1875p on Monday."

Although the example above looks just fine for a human, it had the worst score in the business articles collection. One may observe quite a few occurrences of words merged by a dash (e.g. "four-for-one") or a letter appended to a number (e.g. "19th" or "1875p"). All of these occurrences were marked as misspelled words by JMySpell. Additionally, the shortness of the document contributed to its low recognition accuracy.

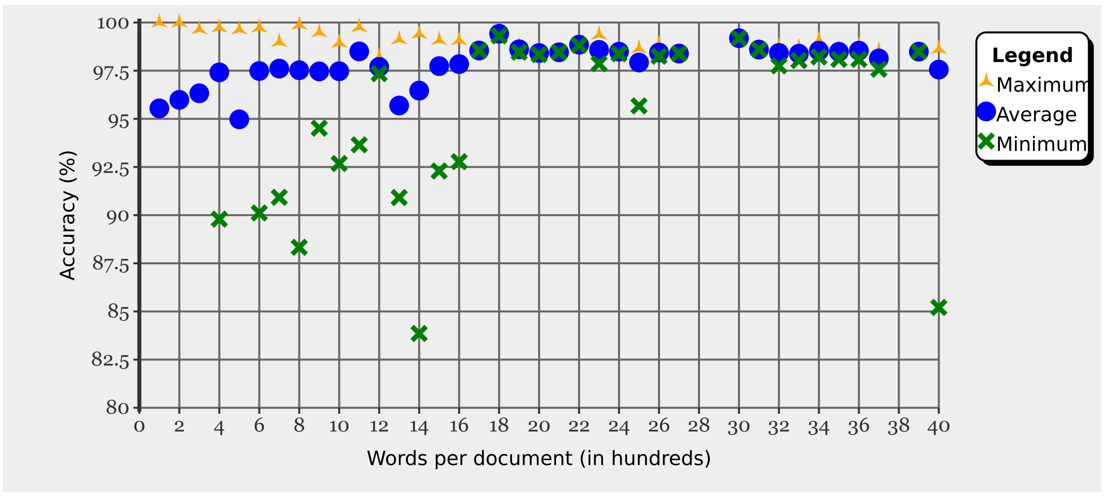


Figure 5.6: Spell-checking of spam. Showing the minimal, maximal, and average values of the test results. Note, that the minimal values on the left side are missing - the values are way under 75%.

Surprisingly, the average results for spam are slightly better than for the business articles: 96.60%. After all, the spamers might be smart enough to use a spellchecker which is, nowadays, integrated in most of the text editing tools. However unlike the business articles, spam has the minimal values much lower.

5.3.3 Readability

5.3.3.1 TexComp

TexComp, introduced in Subsection 4.2.1, evaluates a document for readability and lexical diversity, and returns numerical values representing those measures. Lower lexical diversity value indicates better diversity. On the contrary, higher readability value means a better result. Based on Juho Heinonen' experiments and notes, the thresholds indicating a high quality document are presented in Table 5.3.

Measure	Minimum	Maximum
Readability	40	80
Lex. Diversity	150	250

Table 5.3: TexComp thresholds for a high quality document.

We evaluated all the documents in the business article and spam collection. In Figure 5.7 showing the readability measures, we can see that most of the business articles really are within the mentioned thresholds, and most of the spam has slightly lower values. However, if we considered all the business articles of high quality, it would enable a lot of spam to be considered of high quality as well.

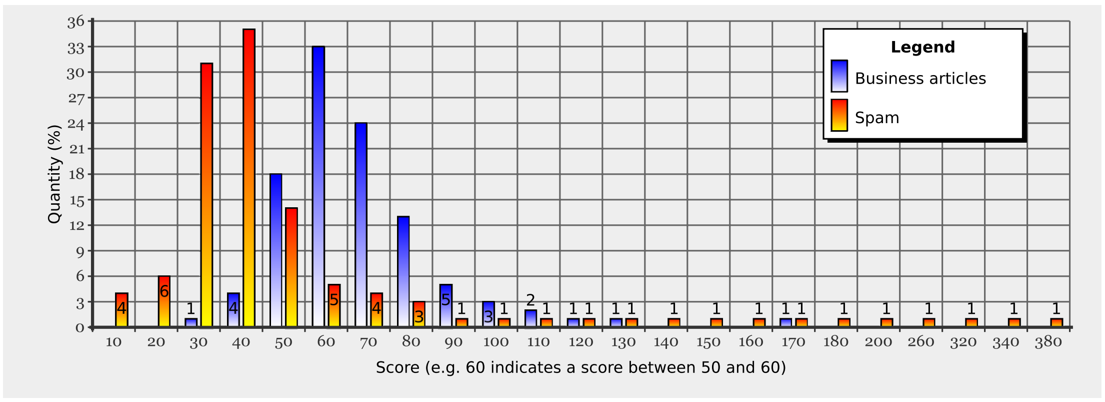


Figure 5.7: TexComp readability evaluation. Percentage quantity value equals to a proportion of the total number of articles.

The measurements of lexical diversity are presented in Figure 5.8. We can see that the values are more dispersed than for the assessment of readability. Yet again, most of the business articles scores are located within the mentioned thresholds.

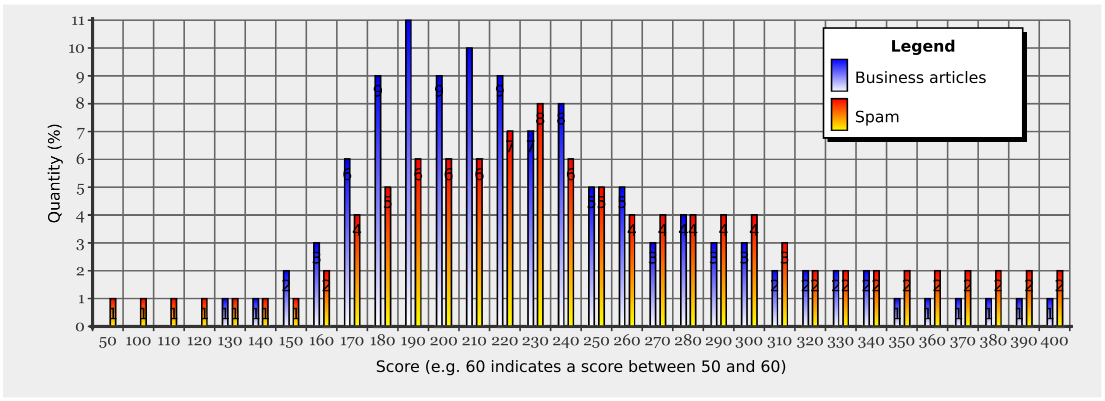


Figure 5.8: TexComp lexical diversity evaluation. Percentage quantity value equals to a portion from the total amount of articles.

Note that the presented figures did not include all the values. There were a few documents, both business articles and spam, with assessment values higher than 1000 or even negative values for lexical diversity. Those values were ignored and not displayed in the figures. The results are summarized in Table 5.4.

Collection	Readability	Lex. Diversity	Both	None
Business articles	86.43%	70.95%	63.10%	5.60%
Spam	23.83%	49.84%	12.88%	39.08%

Table 5.4: TexComp results summary. The values indicate how many documents passed each measurement considering thresholds given in Table 5.3. The column "Both" and "None" indicate how many documents passed both and none of the two measurements.

In Table 5.3, we introduced the thresholds used to distinguish high quality documents. Although it is mentioned that higher readability assessment scores and lower lexical diversity scores mean higher quality of a document, we keep the threshold boundaries in order to indicate if a document is of high quality or not. The main reason is that in some cases, TexComp returns excessive assessment scores, such as 731 for readability of spam and -446 for lexical diversity of spam. The threshold boundaries protect us from documents that obtained an assessment score which is too good to be true.

The business articles are written by professionals, and it would be safe to assume that all those articles are of high quality. Unfortunately, we can see that not all the articles passed the TexComp measurements. A few of the business articles did not pass either readability or lexical diversity assessment. There are two main approaches that we can use to improve the results. However, both of these approaches have their drawbacks:

1. Change the thresholds. It is probably the easiest thing to do. On the other hand, it would allow a lot of spam to pass this assessment tool as well (see Figure 5.7 and Figure 5.8)

2. Improve TexComp. Changing the implementation of TexComp might not be an easy task. Additionally, we cannot be sure that all the high quality documents that passed the test now will be able to pass it again after the evaluation rules become stricter and implementation changes are applied.

The results have more or less met our expectations: business articles have overall better results than spam. In order to get the best possible results, a compromise has to be made. We could either filter out some of the business articles in order to keep a lot of spam filtered as well. Or on the contrary, if we want all the business articles to pass, it would allow a large amount of spam to pass as well.

5.3.3.2 Fathom

Fathom package, used to measure readability in relation to US education system, was introduced in Subsection 4.2.1. The Fathom package contains three readability assessment algorithms, and the results for the business articles and spam collection are shown below: Gunning-Fog Index (Figure 5.9), Flesch Reading Ease (Figure 5.10), Flesh-Kincaid grade level (Figure 5.11).

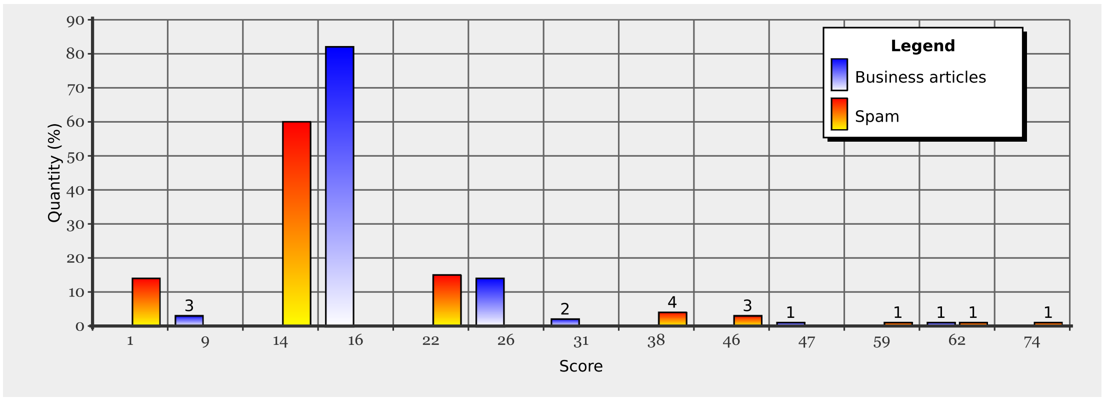


Figure 5.9: Gunning-Fog Index Assessment

Gunning-Fog Index indicates how many years of education need one to understand a text. As we can see in the figure above (5.9), the majority of both, business articles and spam, have a similar score: 14 and 16. Unfortunately, as we might think, not all the remaining spam has a lower score - quite the opposite. This could make our analysis to distinguish spam and high quality documents fairly difficult based on the Gunning-Fox Index scores. However, it could be used similar to TexComp, and the documents with excessive assessment scores, for instance 50, can be filtered out.

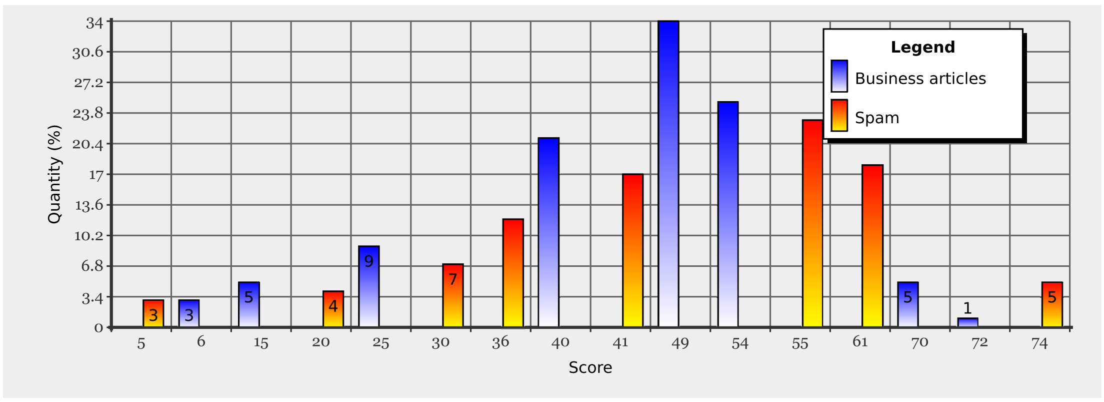


Figure 5.10: Flesch Reading Ease Assessment

Flesch Reading Ease algorithm evaluates a document with a score on 100-point scale. The documents having scores between 60 and 70 are supposed to be the easiest to read [29], understandable for people with 9 years of education. The lower the scores gets, the more difficult it might be to read. As we can see in Figure 5.10, the majority of spam has a score 55 or 61. The business articles are slightly more difficult to read. However, the same applies also to a significant part of spam collection. The fact that 5% of spam messages scored extremely high indicates that Flesch Reading Ease assessment could be used for filtering out some case of spam.

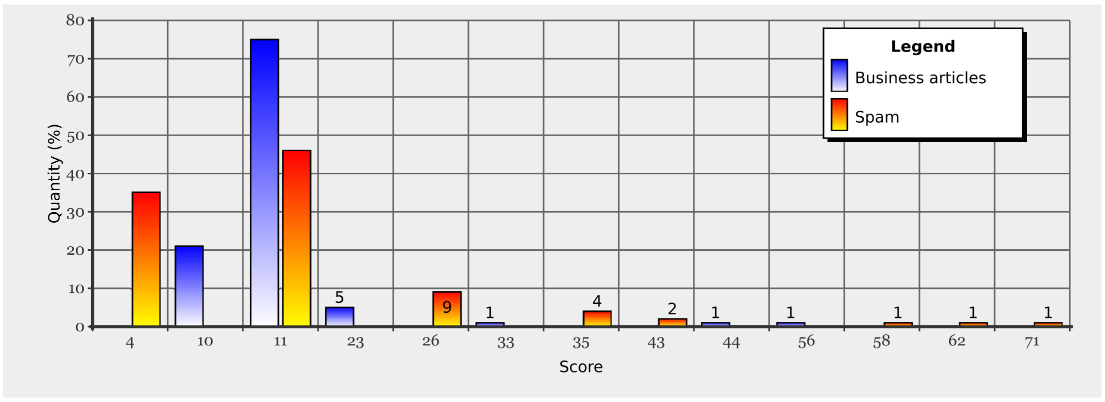


Figure 5.11: Flesh-Kincaid Grade Level Assessment

Flesh-Kincaid Grade Level algorithm uses the same attributes of a document as the Fleash Reading Ease algorithm, and translates the score to indicate how many years of education one needs to understand the text. The results of this algorithm are shown in Figure 5.11. This results do not satisfy our needs at all because the majority of both collections have the same score: 11. Hence, it is not possible to clearly distinguish spam and a high quality document. Nevertheless, the results can be used to filter out documents with very high scores.

In all three algorithms, we could see that the scores for both collections were often overlapping. Additionally, some documents had excessive scores, thus were not displayed in the figures. Table 5.5 shows how many documents from each collection obtained an excessive score, thus were not displayed in the figures.

Flesh Reading Ease algorithm did not perform the tests very well. More than 3% of business articles and 11% of spam obtained an assessment score that, according to the algorithm description, does not have a clear meaning. This fact linked with Fathom assessment scores makes the Gunning-Fog Index the only component from Fathom package that is considered as potentially usable for FAQAD.

Algorithm	Business Articles	Spam
Gunning-Fog Index	0.12%	4.31%
Flesch Reading Ease	3.21%	11.45%
Flesh-Kincaid Grade Level	0.12%	6.92%
Average	1.15%	7.56%

Table 5.5: Excessive assessment values of Fathom package.

5.3.4 Classifier4J

Juho Heinonen who discovered Classifier4J (Subsection 2.5.4) as a potential asset for FAQAD, trained the tool in order to discover spam. He used a couple of documents containing news and business announcements in order to classify high quality documents. We tested the classifier with our test data and got the results shown in Table 5.6.

Data	Classified as Spam
Business Articles	7.88%
Spam	83.48%

Table 5.6: Results of Classifier4J trained by Juho Heinonen.

More than 80% of spam was recognized. That is quite a promising result. However, almost 8% of business articles were also categorized as spam. Statistically speaking, Classifier4J did the job relatively well. In order to improve the results, we trained Classifier4J with our testing data: both business articles and spam. One hundred documents from each category were randomly chosen and used for the training. The results of newly trained Classifier4J are shown in Table 5.7.

Data	Classified as Spam
Business Articles	9.18%
Spam	88.66%

Table 5.7: Results of Classifier4J trained by some of the testing data.

It is clear that after a domain-specific training, the component recognizes more documents as spam. Unfortunately, it applies for both categories, business articles and spam, so we cannot really say that there is an overall improvement. Actually in the first test, more business articles passed the Classifier4J component. Because of this reason, we stick with the original training data model created by Juho Heinonen.

5.3.5 ABCV API

ABCV API, earlier introduced in Subsection 4.2.2, is a tool able to detect certain types of anti-social behavior and conflicts from text. To test the ABCV API, we used the flames collection and the business articles. The results are shown in Table 5.8.

Data	Passed	Rejected	Ratio
Flames	19	87	17.92%
Business Articles	831	9	98.93%

Table 5.8: Results of ABCV API

The ABCV tool rejected about 82% of the flames collection and about 11% of the business articles. After inspecting the rejected business articles, it seems that they did not pass because of elaborating crimes, business failures or simply quoting somebody with a bad language. On the other hand, some of the articles that passed talked about business failures as well.

Compared to the other assessment tools, ABCV API consumes quite a lot processing power. That might not be suitable when evaluating large amount of documents.

5.4 Assigning Overall Quality Scores

Based on the shown results, we can see that the most reliable and reflective assessment tools are TexComp and Classifier4J. In order to combine the results and make a final decision on which tools to include in the FAQAD toolbox, we need to assign weights to the scores of every assessment tool. After extensive experimenting, the weights chosen for FAQAD's assessment tools are presented in Table 5.9.

Combining the scores gives us a result which is used to make a final decision about the quality of a document. The result is actually a value between 0 and 1, but for simplicity it is shown on a 100-point scale (Figure 5.12) where the score 100 implies that the document passed all the assessment tools, thus the document is of a very high quality.

Tool	Weight	Acceptable Scores
ABCV API	17	0
Classifier4J	18	0 - 0.5
Gunning-Fog Index	13	15 - 30
JMySpell	4	92.5 - 100
TexComp - Readability	22	40 - 80
TexComp - Lexical Div.	26	150 - 320

Table 5.9: Score weights of assessment tools. The sum of weights in the table is equal to 100. In the actual implementation, the weights are normalized, and the sum of weights equals to 1.

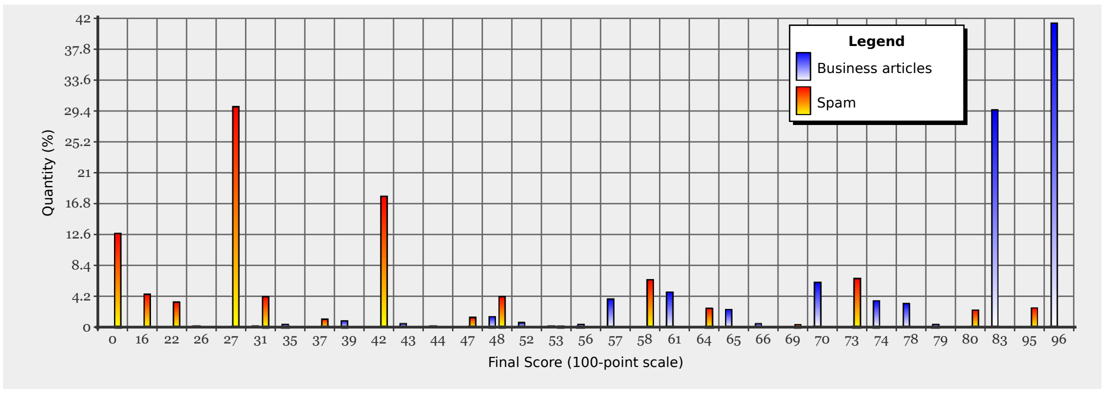


Figure 5.12: Aggregated scores based on the results obtained from each assessment tool.

Figure 5.12 clearly illustrates that FAQAD is able to detect the quality of documents. Business articles have higher scores on the right side, and spam with lower score is on the left side. The final task in designing the FAQAD overall quality scoring mechanism is to set a threshold that indicates the level of sufficient quality. Setting the threshold at value 50 allows 97% of business articles to pass FAQAD and 79% of spam to be filtered out. The threshold could, naturally, be modified depending on the situation. As a consequence, it would allow more documents, from both collections, either to pass FAQAD or to be filtered out.

5.4.1 Quality of Data Sources

The mechanism of using the quality of a data source to affect the score of a document originating from that source was introduced in 4.2.3. The quality of a data source is calculated as an average of the documents' scores previously obtained from that source. Unfortunately in our test data, we do not have the information which exact source was each document obtained from. We could, however, assume that all the business articles are from a single source and spam from another one in order to demonstrate the effect of applying the quality of data source.

Collection	Passed FAQAD	Average Score	Passed FAQAD Affected by Average Score
Business Art.	96.55%	82.82	99.88%
Spam	20.96%	36.13	14.41%

Table 5.10: Demonstrating how many documents are of a high quality either with or without applying the average score. The average score had a weight of 40% in the overall quality score.

In Table 5.10, we can see that the average document score is convenient for our purpose. The average score affected the scores in both collection the way it is supposed to. Figure 5.13 demonstrates how is the quality of data source affecting the documents' scores. In contrast to Figure 5.12, the new figure (5.13) looks "cleaner" because documents from each collection are grouped on one side of the figure. This indicates a clear separation between the business articles and spam.

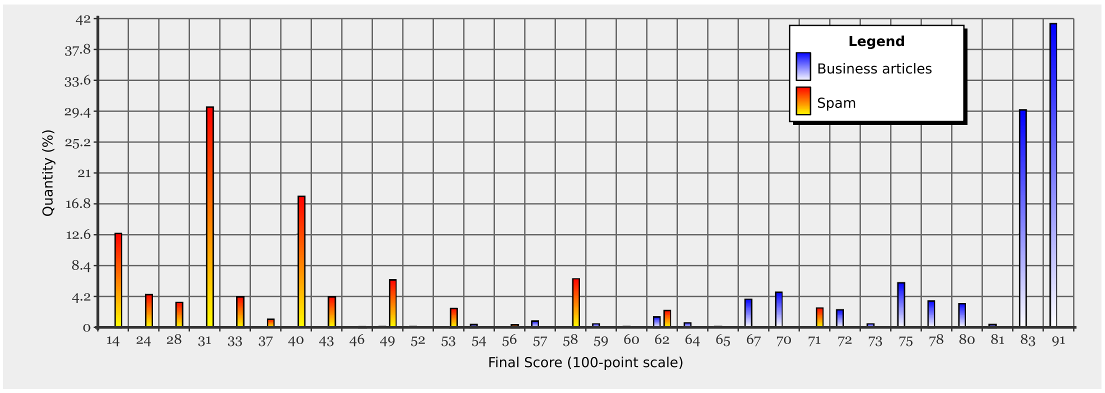


Figure 5.13: Aggregated scores affected by the average score of each collection.

Applying the average score to affect the score of a current document might not always have a positive effect, especially if we have just a little amount of documents from that one source. FAQAD should only use the source-based scoring for documents fetched from sources from which a certain number of documents, for instance 50, have been fetched.

5.5 Discussion

JMySpell was the most accurate tool in the evaluation in which each tool was assessed separately. However, the results of JMySpell for business articles and spam are very close, and mostly the spam results were even better. As an outcome, JMySpell cannot really be used to distinguish high quality data from spam. However, it could be used to filter out documents that are complete trash or documents in which the language was recognized incorrectly.

The accuracy of other tools is close to 90%. Problems that were common to all the tools were as follows:

Text length. The longer the text is, the better options for evaluation the assessment tool has. Text length can severely affect the assessment results.

Abbreviations sometimes confuse the assessment tools as they do not recognize the abbreviation as a correct word.

Technical terms and long words bear the same problem as abbreviations. That means that the words are not recognized as valid words, and are having negative impact on the assessment result.

Quotes are usually taken from another context. Although they are commonly used to give the reader more insight, the assessment tools might get confused as the quotations may contain bad language or even words that do not exist in any common dictionary.

Punctuation which determines, for example, where a sentence begins and where it ends. Some assessment tools, e.g. Fathom package, use the sentence length as one of the criteria to evaluate a document. When there are no periods in the document, the whole text is considered as one large sentence. In conclusion, the assessment tool returns absurd results that are not usable.

In order to improve the evaluation, we could:

Ensure that we consider only documents that are long enough. Based on the results above, we can see that the proper amount of words varies from tool to tool. JTCL had significantly better results with texts with 50 or more words. Other tools, such as TexComp for measuring lexical diversity, need at least 100 words.

Extend dictionaries that we are matching against with new words and possibly abbreviations.

Normalize abbreviations. Replace abbreviations in the input documents with the full words or phrases before processing them with the QA tools.

Train the spam filter with a larger collection of appropriate data, so the filter can easier distinguish which document are of high or low quality.

Improve ontology and scoring mechanism of ABCV API.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we created a prototype of *Framework and API for Quality Assessment of Documents* (FAQAD) which is a part of a large text mining system - *Data Analysis and Visualization aId for Decision-making* (DAVID). FAQAD is a *quality assessment* (QA) component that evaluates the *information quality* (IQ) of text documents in order to filter out low quality documents and prevent the DAVID system to process them further.

We started by reviewing the relevant previous research on QA frameworks. The results of the literature review indicated that in our setting, we are able to use automated QA tools to measure only intrinsic quality dimensions. We continued reviewing the documentation of the DAVID system, so we could follow up and extend the system in an effective way. In order, to build a QA component, we gathered a set of open source tools and tools developed in our research team that could be potentially used for QA in the FAQAD framework. Finally, we tested those QA tools on real-world data and experimented with ways of combining the scores returned by the tools in order to make a final overall QA. The majority of our test data consists of two collections: business articles and e-mail spam. With the optimal parameter settings for our test set, FAQAD was able to accept 99.88% of business articles and filter 85.59% of the spam. The expected values for sufficient accuracy were about 90%. Hence, the obtained levels of accuracy can be considered as good.

The first objective was to figure out how to assess quality of documents and their sources. We gathered numerous QA tools which are able to process fetched documents and save the quality scores to a database. Each QA tool measures the quality of documents using a different algorithm, i.e. evaluating a document from a different point of view. However, strictly speaking, we do not assess the exact dimensions mentioned in the previous researches on QA frameworks. We are forced to use the values returned by each QA tool.

The quality of a data source is calculated as an average score of documents that are retrieved from that source. In our experiments, we set the weight of a data source to 40%, i.e. for each new assessed document, the actual document score has the weight 60% and it is combined with the average score of documents from the same source with the weight of 40%. The results of combining the actual score with the average score had a positive effect, and they are illustrated in Section 5.4.

In our setting, there was not a straightforward way to test user rating as another aspect of the document's quality due to the lack of relevant business data with user ratings. However, we were able to combine the document's score with other scores, i.e. the quality of a data source, thus adding the average user score into the model is going to be relatively easy.

The second objective was to find a way "How to utilize the defined IQ measures". For each measure, we defined what value indicated a high-enough quality, and we assigned a weight to that measure. Using this mechanism, we were able to easily prioritize the various QA measures. Additionally, we utilized the assigned weights to calculate a final score using a weighted mean. The final scores are in a range from 0 to 1. Our experiments indicated that 0.5, i.e. in the middle of the range, was an appropriate threshold for deciding whether a document is of high quality or should be filtered out.

6.2 Future Work

The results of experimenting with our test data collections can be considered successful. There are quite a few shortcomings of the QA tools which were discussed in Section 5.5. In order to improve our results, we should consider those shortcomings.

The user rating for assessing a document was not tested. However, the idea is that a user can evaluate each document in his own perception and assign a score to it. This can be very overwhelming for the user if there is a large amount of documents. In order to make this process more efficient, we could consider use the user rating to assess directly a data source and not every single document. At the moment, we only allow this feature via the blacklist mechanism.

Documents are fetched from data sources, and the user has the ability to ban a specific data source by adding it to a blacklist in the DAVID system. We could consider creating a white list of data sources from which the documents are fetched without the need to be processed by FAQAD.

List of Abbreviations

ABCV	Anti-social Behaviour, Conflict and Violence
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange; i.e. the basic standard of text representation on digital systems
AWT	Abstract Windows Toolkit
BI	Business Intelligence
CLI	Command Line Interface
CoProE	Company, Product and Event
CVS	Concurrent Versions System
DAVID	Data Analysis and Visualization aId for Decision-making
DB	Database
EO	Exception Object
FAQAD	Framework and API for Quality Assessment of Documents
GC	Garbage Collector
GUI	Graphical User Interface
IDE	Integrated Development Environment
IQ	Information quality
IS	Information Systems
ISO	International Organization for Standardization
NLP	Natural language processing
OBIE	Ontology-Based Information Extraction
PHP	PHP: Hypertext Preprocessor, originally Personal Home Page
QA	Quality Assessment
RCP	Rich Client Platform
RSS	RDF Site Summary
SVG	Scalable Vector Graphics
SW	Semantic Web
TM	Text Mining

UI	User Interface
VM	Virtual Machine

Bibliography

[1] Apache Lucene. http://lucene.apache.org/. [Online; accessed 2.4.2012].

[2] Classifier4J. http://classifier4j.sourceforge.net/. [Online; accessed 2.3.2012].

[3] CPAN. http://search.cpan.org/. [Online; accessed 1.2.2012].

[4] Eclipse. http://www.eclipse.org/. [Online; accessed 13.1.2012].

[5] Java Text Categorizing Library. http://textcat.sourceforge.net/. [Online; accessed 1.12.2012].

[6] JMySpell. http://kenai.com/projects/jmyspell. [online; accessed 2.3.2012].

[7] libTextCat. http://software.wise-guys.nl/libtextcat/. [Online; accessed 1.12.2012].

[8] MySQL. http://www.mysql.com/. [Online; accessed 13.1.2012].

[9] Naive Bayes classifier. http://www.sccs.swarthmore.edu/users/08/ajb/tmve/wiki100k/docs/Naive_Bayes_classifier.html. [Online; accessed 12.12.2011].

[10] Rich Client Platform. http://wiki.eclipse.org/Rich_Client_Platform. [Online; accessed 11.1.2012].

[11] Snowball. http://snowball.tartarus.org/. [Online; accessed 11.5.2012].

[12] Statistics toolbox: Naive bayes classification. http://www.mathworks.se/help/toolbox/stats/br039qw-1.html. [Online; accessed 14.1.2012].

[13] The Java Tutorials: Essential Classes. http://docs.oracle.com/javase/tutorial/essential/. [Online; accessed 18.2.2012].

[14] Voikko. http://voikko.sourceforge.net/. [Online; accessed 2.11.2012].

[15] Janet E. Alexander and Marsha A. Tate. *Web Wisdom; How to Evaluate and Create Information Quality on the Web*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1st edition, 1999.

[16] Christopher Allen. Using 5-star rating systems. http://www.lifewithalacrity.com/2006/08/using_5star_rat.html, 2006. [Online; accessed 11.11.2012].

[17] E. Arendarenko and T. Kakkonen. Ontology-based information and event extraction for business intelligence. In *Proceedsing of the 15th International Conference on Artificial Intelligence: Methodology, Systems, Applications*, Varna, Bulgaria, 2012.

[18] Benjamin Biron and Ryan Sciampacone. Real-time Java, Part 4: Real-time garbage collection. http://www.ibm.com/developerworks/java/library/j-rtj4/. [Online; accessed 11.4.2012].

[19] Christian Bizer. *Quality-Driven Information Filtering- In the Context of Web-Based Information Systems*. VDM Verlag, Saarbrücken, Germany, Germany, 2007.

[20] Hans J. Boehm. A garbage collector for C and C++. http://www.hpl.hp.com/personal/Hans_Boehm/. [Online; accessed 11.4.2012].

[21] Internet System Consortium. Internet host count history. http://www.isc.org/solutions/survey/history. [Online; accessed 13.1.2012].

[22] Y. Dai, T. Kakkonen, and E. Sutinen. MinEDec: a decision-support model that combines text-mining technologies with two competitive intelligence analysis methods. *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)*, 3:165–173, 2011.

[23] Adenekan Dedeke. A conceptual framework for developing quality measures for information systems. In *Proceedings of the 5th International Conference on Information Quality*, pages 126–128, 2000.

[24] Larry English. Defining and measuring accuracy. http://www.information-management.com/issues/20030701/6954-1.html, July 2003. [Online; accessed 15.6.2012].

[25] Larry P. English. Information quality: Meeting customer needs. *DM Review*, 3(1), November 1996.

[26] Martin J. Eppler and Peter Muenzenmayer. Measuring information quality in the web context: A survey of state-of-the-art instruments and an application methodology. In *IQ*, pages 187–196, 2002.

[27] Joerg Evermann and Yair Wand. Ontology based object-oriented domain modelling: fundamental concepts. *Requirements Engineering*, 10(2):146–160, 2005.

[28] Barry Feigenbaum. SWT, Swing or AWT: Which is right for you? http://www.ibm.com/developerworks/grid/library/os-swingswt/, February 2006. [Online; accessed 12.1.2012].

[29] Rudolph Flesch. A new readability yardstick. *Journal of Applied Psychology*, 32(3):221 – 233, June 1948.

[30] Florida Laws: FL Statutes - Title XXXVII. Readable language in insurance policies. `http://law.onecle.com/florida/insurance/627.4145.html`. [Online; accessed 12.2.2012].

[31] Robert Gunning. *The Technique of Clear Writing*. McGraw-Hill, New York, 1952.

[32] Enrique Herrera-Viedma, Gabriella Pasi, Antonio Gabriel López-Herrera, and Carlos Porcel. Evaluating the information quality of web sites: A methodology based on fuzzy computing with words. *Journal of the American Society for Information Science and Technology*, pages 538–549, 2006.

[33] Alois Heuboeck, Jasper Holmes, and Hilary Nesi. The BAWE Corpus Manual. `http://www.engelska.uu.se/Forskning/engelsk_sprakvetenskap/Forskningsomraden/Electronic_Resource_Projects/USE-Corpus/`. [Online; accessed 16.2.2012].

[34] Norman Walsh Ian Jacobs. Architecture of the World Wide Web. `http://www.w3.org/TR/webarch/`. [Online; accessed 2.4.2012].

[35] Md Rafiqul Islam and Wanlei Zhou. An innovative analyser for email classification based on grey list analysis. In *Proceedings of the 2007 IFIP International Conference on Network and Parallel Computing Workshops*, NPC '07, pages 176–182, Washington, DC, USA, 2007. IEEE Computer Society.

[36] Tuomo Kakkonen Juho Heinonen. DataSourceQualityChecker - Technical Documentation, Unpublished.

[37] Beverly K. Kahn, Diane M. Strong, and Richard Y. Wang. Information quality benchmarks: product and service performance. *Communications of the ACM*, 45(4):184–192, 2002.

[38] T. Kakkonen and T. Mufti. Developing and applying a company, product and business event ontology for text mining. In *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, pages 24:1–24:8, Graz, Austria, 2011. ACM.

[39] Tuomo Kakkonen. TexComp - a text complexity analyzer for student text. In *Proceedings of the 12th International Conference on Interactive Computer-aided Learning*, Villach, Austria, September 2009.

[40] Tuomo Kakkonen. Towards e-leadership: Higher profitability through innovative management and leadership systems. `http://cs.joensuu.fi/~tkakkone/eleadership/`, 2011. [Online; accessed 28.1.2012].

[41] Pairin Katerattanakul and Keng Siau. Measuring information quality of web sites: development of an instrument, 1999.

[42] Erich Gamma Kent Beck et al. JUnit. http://junit.sourceforge.net/. [Online; accessed 6.12.2012].

[43] J. Peter Kincaid, Robert P. Fishburne, Richard L. Rogers, and Brad S. Chissom. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, National Technical Information Service, Springfield, Virginia 22151, February 1975.

[44] George Roger Klare. *Measurement of Readability.* Iowa State University Press, 1963.

[45] Barbara D. Klein. When do users detect information quality problems on the world wide web? In *Eighth American Conference in Information Systems, Dallas*, 2002.

[46] Universität Leipzig. Wortscahtz - Leipzig Corpora Collection. http://corpora.informatik.uni-leipzig.de/. [Online; accessed 07.09.2012].

[47] Gez Lemon. Readability test. http://juicystudio.com/services/readability.php. [Online; accessed 6.11.2012].

[48] Hareton K. N. Leung. Quality metrics for intranet applications. *Information & Management*, 38(3):137 – 152, 2001.

[49] Ling Liu and M. Tamer Özsu, editors. *Encyclopedia of Database Systems.* Springer New York, USA, 2009.

[50] Ziming Liu and Xiaobin Huang. Evaluating the credibility of scholarly information on the web: A cross cultural study. *The International Information & Library Review*, 37(2):99 – 106, 2005.

[51] David Loshin. What is high quality information? http://searchdatamanagement.techtarget.com/answer/What-is-high-quality-information. [Online; accessed 12.11.2012].

[52] Monika Machunik. Ontology-aided business document classification. Master's thesis, University of Eastern Finland, 2012.

[53] F. Mairesse, M. A. Walker, M. R. Mehl, and R. K. Moore. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of Artificial Intelligence Research*, 30:457–500, 2007.

[54] Zdravko Markov and Daniel T. Larose. *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage.* Wiley, New Jersey, 2007.

[55] Dragana Marković. The second industrial revolution. http://www.b92.net/eng/special/tesla/life.php?nav_id=36502. [Online; accessed 11.11.2012].

[56] Alessandro Mazzei, Daniele P. Radicioni, and Raffaella Brighi. *NLP-based extraction of modificatory provisions semantics.* International Conference on Artificial Intelligence and Law '09. ACM, New York, NY, USA, 2009.

[57] Felix Naumann. *Quality-driven query answering for integrated information systems.* Springer-Verlag, Berlin, Heidelberg, Germany, 2002.

[58] Felix Naumann and Claudia Rolker. Assessment methods for information quality criteria. In *Fifth Conference on Information Quality (IQ 2000), Boston, MA*, pages 148–162, 2000.

[59] Shukrat Nekbaev, Tuomo Kakkonen, and Ernest Arendarenko. DAVID Technical Documentation, Unpublished.

[60] BBC News. Apple computers 'hacked' in breach. `http://www.bbc.co.uk/news/technology-21510791`. [Online; accessed 21.2.2013].

[61] Larry Ogrodnek. Java Fathom. `http://www.representqueens.com/fathom/`. [Online; accessed 6.12.2012].

[62] Moleshe V. De la Harpe R. Parker, M. B. and G. B. Wills. An evaluation of information quality frameworks for the world wide web. `http://eprints.ecs.soton.ac.uk/12908/1/WWW2006_MParker.pdf`. [Online; accessed 06.11.2012].

[63] Leo L. Pipino, Yang W. Lee, and Richard Y. Wang. Data quality assessment. *Communications of the ACM*, 45:211–218, April 2002.

[64] Michael Eugene Porter. *On Competition.* Harvard Business School Press, Boston, Massachusetts, USA, 1998.

[65] Sandy Ressler. Perspectives on electronic publishing: standatds, solutions and more. `http://www.itl.nist.gov/iaui/ovrt/people/sressler/Persp/Views.html`, 1993. [Online; accessed 12.2.2012].

[66] Peter Muenzenmayer Ronald Hyams. How Swiss Re adds business value through quality information. `http://findarticles.com/p/articles/mi_m0FWE/is_8_10/ai_n26987153/`, August 2006. [Online; accessed 28.11.2011].

[67] Graeme Shanks and Brian Corbitt. Understanding data quality: Social and cultural aspects. In *Proceedings of the 10th Australasian Conference on Information Systems*, pages 785–797, Wellington, New Zeland, 1999.

[68] Harris Shiffman. Making sense of java. `http://www.disordered.org/Java-QA.html`. [Online; accessed 6.2.2012].

[69] Diane M. Strong, Yang W. Lee, and Richard Y. Wang. Data quality in context. *Communications of the ACM*, 40:103–110, May 1997.

[70] Diane M. Strong, Yang W. Lee, and Richard Y. Wang. Data quality in context. *Commun. ACM*, 40(5):103–110, May 1997.

[71] Howard Veregin. The NCGIA Core Curriculum in GISience. `http://www.ncgia.ucsb.edu/giscc/units/u100/u100_f.html`. [Online; accessed 26.1.2012].

[72] Lars Vogel. Eclipse RCP Tutorial. `http://www.vogella.de/articles/EclipseRCP/article.html`, January 2012. [Online; accessed 16.1.2012].

[73] Richard Y. Wang and Diane M. Strong. Beyond accuracy: what data quality means to data consumers. *Journal of Management Information Systems*, 12:5–33, March 1996.

[74] Margareta Westergren Axelsson and Ylva Berglund. Uppsala Student English Corpus. Department of English, Uppsala University, Sweden. `http://www.engelska.uu.se/Forskning/engelsk_sprakvetenskap/Forskningsomraden/Electronic_Resource_Projects/USE-Corpus/`. [Online; accessed 16.2.2012].

[75] R. H. J. Zeist and P. R. H. Hendriks. *Specifying software quality with the extended ISO model*, volume 5. Springer Netherlands, Dordrecht, 1996. 10.1007/BF00209185.

[76] Xiaolan Zhu and Susan Gauch. Incorporating quality metrics in centralized/distributed information retrieval on the world wide web. In *Special Interest Group on Information Retrieval '00*, pages 288–295, 2000.