Master Thesis

Ubiquitous Knowledge Processing Lab

Computer Science Department

Darmstadt University of Technology

Combining Answers from heterogeneous Web Documents for Question Answering

Christian M. Meyer

Supervisor: Prof. Dr. Iryna Gurevych

Coordinator: Dr. Delphine Bernhard
Kateryna Ignatova

Darmstadt, April 28, 2009

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Master Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 28. April 2009

Christian M. Meyer
Combining Answers from
heterogeneous Web Documents
for Question Answering
Masterarbeit

Ubiquitous Knowledge Processing Lab
Fachbereich Informatik
Technische Universität Darmstadt

Abstract

Currently, the information of the World Wide Web is mainly accessed with search engines. Recent studies showed that the usual keyword-in-context lists are not always the best choice for presenting the results. Additionally, an increasing amount of people uses search engines, without knowing how to formulate good queries. This master thesis therefore describes the design and implementation of a question answering system that generates a summarized answer for open-domain natural language queries. The system aims to increase the quality of existing systems by using heterogeneous documents from Wikipedia, Yahoo! Answers and Frequently Asked Questions.

Three main tasks have been identified: The first is passage extraction, which relies on semantic similarity and Hidden Markov Models for identifying irrelevant passages. Passage extraction obtains an average precision of 98% and recall of 81%. The second task calculates different clusterings for assigning a topic to each document. The best results have been found by combining k-means and Newman's community clustering, which results in an average clustering purity of 88%. The final step combines three different rankings and selects the top ranked sentences for composing the summary. Besides the textual summary that is particularly useful for answering definition questions, a list of frequent n-grams and URLs is created to support also factoid and list questions. While working with heterogeneous data, combining different approaches has been observed to be crucial for benefiting from the individual advantages and alleviate differences in format, length, style, focus, relevance as well as problems of ambiguity and redundancy within the documents.

An evaluation of the resulting summaries has been done by comparing the system's ROUGE scores with the two systems MEAD and START. User-generated answers from ask.com and Answerbag are used as a reference corpus. The evaluation shows that the system obtains the highest F-measure scores and leads to overall useful summaries. A t-test showed that the system's ROUGE score improvements are significant.

Zusammenfassung

Suchmaschinen sind heutzutage die gängige Methode, um Informationen im Internet zu finden. Häufig werden die gefundenen Seiten mit Titel und kurzer Beschreibung aufgelistet, was laut neueren Studien nicht immer die beste Art der Darstellung ist. Da eine steigende Zahl der Nutzer über keine oder wenig Erfahrung im Umgang mit Suchmaschinen verfügt, wird außerdem eine intuitive Benutzerschnittstelle zunehmend wichtiger. Ziel dieser Masterarbeit ist daher die Entwicklung eines Question-Answering-Systems, das eine Zusammenfassung zu natürlichsprachlichen Fragen generiert. Dabei soll die Qualität existierender Systeme verbessert werden, indem heterogene Daten aus Wikipedia, Yahoo! Answers und Frequently Asked Questions (FAQ) zum Einsatz kommen.

Das System besteht aus drei Teilschritten: Zunächst werden im Passage-Extraction-Teil irrelevante Abschnitte mittels semantischer Ähnlichkeit und Hidden-Markov-Modellen identifiziert. Dabei werden Precision- und Recallwerte von 98% bzw. 81% erreicht. Anschließend werden verschiedene Clusterings berechnet, um jedem Dokument ein Thema zuzuordnen. Die besten Ergebnisse zeigten sich bei einer Kombination aus k-Means- und Newmans Community-Clustering, welche zu einer mittleren Clustering-Purity von 88% führte. Im letzten Teilschritt werden die relevanten Sätze absteigend nach Relevanz geordnet und in einer Zusammenfassung kombiniert. Neben der Antwort im Fließtext, die insbesondere zur Beantwortung von Definitionsfragen nützlich ist, wird eine Liste häufiger n-Gramme und URLs generiert, um auch Fakten- und Listenfragen zu unterstützen. Insbesondere die Kombination unterschiedlicher Ansätze hat sich als wichtig erwiesen, um mit den Unterschieden in Format, Länge, Stil, Tiefe, Relevanz, sowie Mehrdeutigkeiten und Redundanz von heterogenen Daten besser umgehen zu können.

Die erzeugten Zusammenfassungen werden mit Ergebnissen der Systeme MEAD und START verglichen. ROUGE wird zur quantitativen Evaluation eingesetzt, wobei Beiträge der Plattformen ask.com und Answerbag als Referenzkorpus dienen. Die Ergebnisse zeigen, dass das System die höchsten F-measure-Werte erzeugt und die generierten Zusammenfassungen im Wesentlichen informativ und nützlich sind. Die erreichten Verbesserungen im F-measure-Wert sind, wie ein t-Test zeigte, statistisch signifikant.

Contents

Chapter 1

Introduction

The chapter addresses motivation and background of this master thesis and then defines the goals of the project. Finally, a structural outline of the following chapters is given.

1.1 Motivation

The size of the World Wide Web has been constantly increasing during the recent years. While Bharat and Broder (1998) calculated over 200 million pages in November 1997, Gulli and Signorini (2005) present already over 11.5 billion web pages as of January 2005. This promotes the usage of the web as a common knowledge base. According to the empirical study in Möller (2004) p. 50, the amount of people, using the web as knowledge base, has increased by 25% since 1999 through 2003; in the age group 20–29 the number of people even grew from 19% to 59%.

Currently, information is usually found and accessed by a search engine. Common search engines display a list of search results, hyper linking to a certain web page and maybe showing the search keywords in the context of the page. Jansen et al. (2000) however found out that 58% of the users tend to only look at the first page (usually 10 results), although there are plenty of pages with further search results. Kaisser et al. (2008) evaluated the expected text length of a search result, ranging from a single phrase to a whole article. They found out that it is best to provide a short answer for precise questions (like asking for a certain fact) and a long answer for broader questions (like definitions, advices or opinions). Having these arguments in mind, a summary of multiple search results, consisting of the most important information, could be useful. Yahoo!, as the first of the biggest search engine publishers, is actually working on a summarization technique of search results.[1]

Not only the size of the web is changing, but also its structure. Especially the idea of collaborative content (or user-created content) is getting more and more popular, according to Vickery and Wunsch-Vincent (2007). The so called Web 2.0 platforms offer the possibility

[1] PCWorld online news, http://www.pcworld.com/article/154929, December 4, 2008

to create and edit web content without any technical understanding of HTML and other web terms or techniques. As more users appear on the web, there is a need for smart and easy to use interfaces. This also leads to the idea of processing natural language queries instead of a plain keyword search and to provide the search results in natural language texts.

1.2 Goals

The Goal of this Master Thesis is the implementation of a question answering system that combines different data sources on the web to provide a natural language answer for a natural language search query. In a document segmentation and passage extraction step, candidate snippets should be identified that most likely contain relevant information. To achieve that, semantic similarity scores should be calculated and used in a relevance classification algorithm.

The answer should be an extractive multi-document summary of the candidate web snippets. Sentences are to be ranked, to support the summarization task. A graphical user interface will also be provided, which allows submitting the query and adjusting settings for the returned answer summary.

Initially, the system is set up for answering definition questions, like *"What is flash media?"*. An extension for list and factoid questions should be possible but will not be discussed in detail within this thesis.

A qualitative and quantitative evaluation of the main steps of the system will be given. Evaluation will focus on queries of the computer science domain to allow an easier judgment of the resulting summaries; the system is however not domain specific and should be able to handle arbitrary (definition) queries.

1.3 Thesis Organization

The thesis is organized as follows:

Chapter 2 gives an overview on the system architecture, the evaluation data and the input data, before the project is set in the context of related work.

Chapter 3 describes the preprocessing steps, which include segmentation, part-of-speech tagging, word sense disambiguation and the calculation of semantic similarity.

Chapter 4 addresses the extraction of candidate snippets and the filtering of irrelevant passages. Each passage is classified to be relevant or irrelevant by checking its semantic similarity to the query for a certain threshold. A Hidden Markov Model is then used to refine this classification by finding natural boundaries between relevant and irrelevant passages.

Chapter 5 describes clustering methods to determine the topic of a document. A hierarchical, an iterative and a graph-based clustering approach will be tested and then improved by merging the results of the individual techniques.

Chapter 6 introduces the summarization techniques that are used to create the system result. The selected sentences are ordered according to the results of three different ranking algorithms. The identified topics are then also ranked and automatically labeled, before the resulting summary is formatted for the output. The user interface is finally introduced and the summaries are evaluated.

Chapter 7 sums up the results of each processing step and draws a conclusion. The usage of merging and combination techniques turned out to be useful when working with heterogeneous data and will be discussed in detail. An outlook of possible improvements is also given.

Appendix A lists the 60 example queries that have been used for the evaluation. Some of them have been manually annotated to measure the system's performance; the statistics of these annotations is described here.

Appendix B describes the system components and the annotations that are created in each step.

Chapter 2

Project Overview

This chapter defines the term *Question Answering System* and describes the heterogeneous data sources that are used to retrieve answers from the web. The main scientific and technical challenges are then addressed, which lead to the objectives of the thesis. Finally, the system architecture is presented and the project is set in the context of previous work.

2.1 Question Answering System

Like a search engine, a question answering system returns results (i.e. answers) for a specific query that is provided by the user. The difference is however that the query can be an actual question in the user's natural language, while the resulting answer also consists of natural language text.

If for example the user wants to know who invented the printing press, an appropriate search engine query would possibly be *"printing press invention"* and result in a number of web links, each coming with a title and a short description. The corresponding query to a question answering system could however be *"Who invented the printing press?"* and result in a short summary about the history of printing.

Figure 2.1 shows a comparison between the Google results for the search query above and a fictive question answering system, whose answer has been taken manually from Wikipedia. The search engine acts as a hub that leads the user to a page, which hopefully contains the answer. The question answering system however works as a source for information itself. An overview of existing question answering systems is provided in section 2.6.

Typical components of a question answering system include question analysis, document and passage retrieval as well as answer extraction. Question analysis aims to understand the natural language question and formulate queries, which can be used during document retrieval. The result of the document retrieval task is a collection of documents that most likely contain the answer. Relevant passages (often called candidate snippets) are finally identified and the answer is composed.

(a)

Who invented the printing press?

A printing press is a mechanical device for applying pressure to an inked surface resting upon a medium (such as paper or cloth), thereby transferring an image. The mechanical systems involved were first assembled in Germany by the goldsmith Johannes Gutenberg around 1439, based on existing screw-presses used to press cloth, grapes etc., and possibly to print woodcuts, which were printed in Europe before Gutenberg.

(b)

Figure 2.1: Comparison of search engine and question answering system

2.2 Answer Retrieval

In order to determine an answer for a natural language query, this thesis combines several documents from the World Wide Web. The *Apache Lucene project*[2] is used to retrieve these documents, following state-of-the-art techniques, like Gospodnetić and Hatcher (2004). As the retrieval task is not part of the project, it will not be described in detail. Documents are taken from three different data sources, which are briefly described in the following. The system is however designed to add other data sources to improve the results.

The first data source are *Frequently Asked Questions* (FAQs), taken from arbitrary web pages. Each FAQ document consists of a single question and the corresponding (single) answer. The answer is typically 1–2 paragraphs long and specific to the domain of the including web page. It is a redacted text, mostly written by a specialist, that contains only few errors and is usually of formal language. Looking through the data, some entries have been found, that focused on a single product and contained advertisement—thus objectivity will be an issue here. Besides that, webmasters tend to copy their FAQ from other pages, resulting in duplicate or slightly modified entries from different URLs.

FAQs have been used for question answering earlier: As of 2005 a corpus of 293,000 pages with about 2.8 million pairs of question and answer has been created in Jijkoun and de Rijke (2005) at the University of Amsterdam. The corpus is available from the Web[3] and is used for this thesis.

As second data source, *Yahoo! Answers*[4] was chosen. The platform provides 52,148,802 questions (total search results as of 11/2008). Each question can hold multiple replies with

[2]The Apache Lucene project – http://lucene.apache.org
[3]ILPS group at University of Amsterdam – http://ilps.science.uva.nl/resources/webfaq
[4]Yahoo! Answers – http://answers.yahoo.com

What is Flash?

Macromedia Flash is a graphic format that allows for complex, animated web graphics with relatively small file sizes. The users browser will need the Flash player plug-in to view it. According to Macromedia, over 90% of internet browsers already have the player installed. More information about the Flash format and Flash players can be found at www.macromedia.com.

(a)

what is the flash player 8 - macro media inc.. tell about all information?

its a player that plays most of the web based application and some other PC application. Macromedia has been bought be Adobe. Most web ads reguires flash player because they were developed with Macromedia Flash or it developed in flash format

for more info bisit http://macromedia.com

(b)

Figure 2.2: Example documents from (a) the FAQ collection and (b) Yahoo! Answers

a dedicated "Best Answer" amongst them. Currently, only the "Best Answer" is retrieved by the retrieval engine, as it should be the most informative answer to the user's question and could successfully be used in Liu et al. (2008) before. The exact number of answers or best answers could not be found on the platform, however a Yahoo! press release[5] from 05/2006 states more than 10 million answers. For this thesis a subset of 240,000 pairs of question and answer from the computer category is used.

While the FAQ collection contains redacted text, Yahoo! Answers provides user-generated content in both questions and answers. Answers vary in length—from short one-sentence answers up to several extensive paragraphs. The language is often informal or lax, leading to error-prone and incomplete sentences. Figure 2.2 shows example documents from the FAQ collection and from Yahoo! Answers. Note the misspellings like "reguires" and "bisit" as well as the misuse of "its" and the lack of commas in the Yahoo! Answer.

Finally, *Wikipedia*[6] is used as third data source. Unlike the question/answer pairs from the previous data sources, Wikipedia offers encyclopedic articles about a certain topic, usually structured in multiple paragraphs, lists, tables and section headers. As Wikipedia has a large community, constantly proof-reading and correcting articles, the language is formal and has only few errors. The English language version contains 2,652,606 article pages as of their statistics page of 12/2008.[7] The retrieval engine however works with a local database dump to speed up the process. It contains 1,660,067 article pages and is a complete dump of the English Wikipedia from June 2007. An example excerpt of the Wikipedia article on the printing press can be found in figure 2.1 (b).

[5] Yahoo! press release – http://pressroom.yahoo.com/ReleaseDetail.cfm?ReleaseID=196681
[6] Wikipedia, the free encyclopedia – http://en.wikipedia.org
[7] Size of Wikipedia – http://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia

2.3 Objectives

The previous section described the differences in the data sources. The question answering system needs to deal with this heterogeneity. This applies to differences in

- *Format:* textual articles, tables, lists, single pairs of question and answer, questions with multiple answers.
- *Length:* single facts (e.g. *"on 02/26/2009"*), single sentences, several paragraphs, multiple sections with headers.
- *Language style:* formal, technical, colloquial, spelling/grammar errors.
- *Focus:* texts on a single or multiple topics, common/broad vs. in-depth knowledge.
- *Relevance:* whole document is relevant, only some passages, nothing.
- *Ambiguity:* unique or multiple different interpretations of a query.
- *Redundancy:* duplicate documents, paragraphs or sentences, slight reformulations, different compositions of the same information.

Given this issue list, several processing steps are required to provide an appropriate answer. *Preprocessing* steps will be used to overcome different document formats. Large documents are segmented to gain access to the individual linguistic levels and units. The question parts of the retrieved documents are ignored as the system should return answers rather than new questions. They can however be used to learn about a document's topic and relevance.

To distinguish relevant and irrelevant parts in the retrieved documents, a *passage extraction* component will be used, which evaluates the similarity between the document's passages and the query. As the most important information should be presented first, the relevant passages are ordered according to their importance by using a *ranking* algorithm.

Different topics can be found by *clustering* the document collection. A label for each cluster will allow the user to select which topics are of interest. To achieve that, ambiguity needs to be handled—usually the context of a word helps finding out its meaning.

The final result will be composed of the best passages by *summarizing* them. Redundancy needs to be avoided by removing duplicate facts. Besides that, quality assurance is necessary to ensure the correctness of the sentences and to produce good results. Identifying different writing styles, like the usage of formal or colloquial language, could be useful, but will not be considered here. The user will also be allowed to define several parameters that adjust the resulting summary.

2.4 System Architecture

The project is implemented on top of the *Unstructured Information Management Architecture (UIMA)*.[8] The framework focuses on the analysis of large amounts of unstructured data. Unstructured data is e.g. plain text, html files, images, audio or video files. The main idea is to first annotate the data, for instance by identifying relevant passages or tagging each token with its corresponding part-of-speech. The annotations can then be used in so called consumers, which produce the analysis results, like in this case the summarized answer.

Currently, the retrieved answers are stored in XML files and read using a UIMA reader component. As there is not yet a connection to a retrieval engine, the data file for the given query needs to be already present. After reading the retrieved answers, the preprocessing steps are performed, including segmentation, word sense disambiguation, POS tagging and stop word detection. Chapter 3 describes these preprocessing steps.

Further processing includes the three main tasks *Passage Extraction*, *Topic Clustering* and *Summarization* as introduced in the previous section. The chapters 4–6 describe these tasks in detail. The last task, summarization, passes the resulting data to the *Graphical User Interface* that formats and displays the summary. The GUI itself is written in JSP and can for example be run on an Apache Tomcat.[9] Section 6.5 presents the structure and appearance

[8] UIMA – http://incubator.apache.org/uima
[9] Apache Tomcat – http://tomcat.apache.org

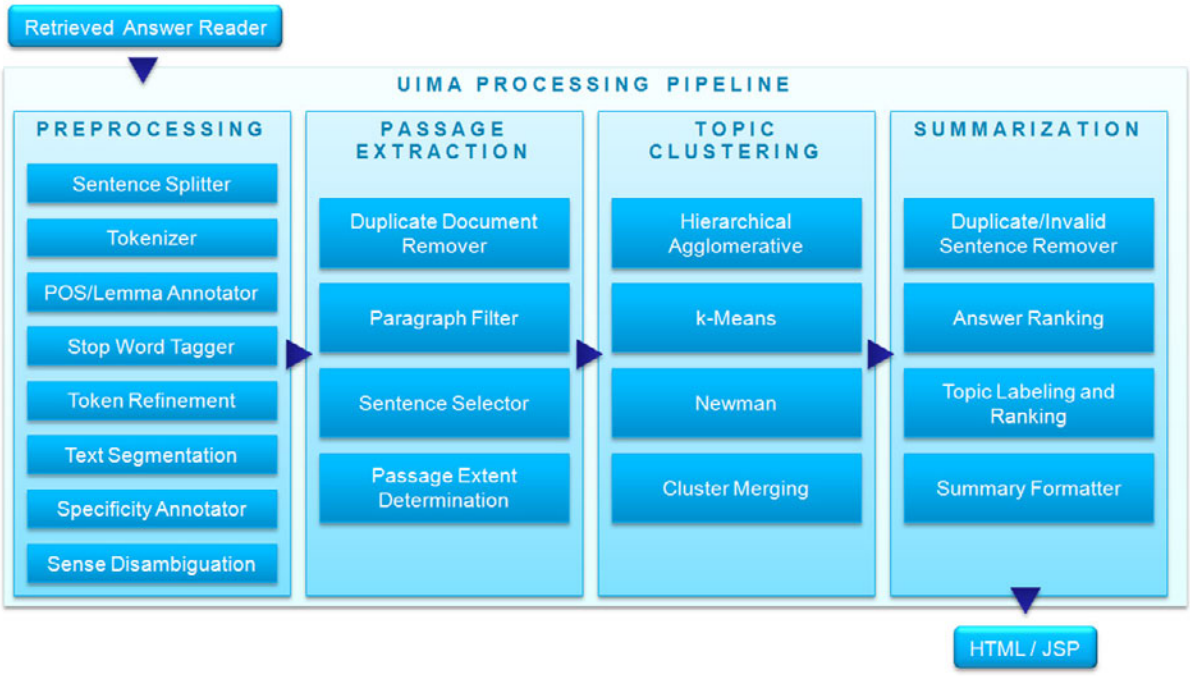


Figure 2.3: Overview on the system's components and architecture

of the JSP pages. The main processing will be deployed as a web service and can thus be accessed by the JSP application.

Figure 2.3 shows the different components and their composition. There are several libraries and supporting data sources included, to solve the tasks above. Where possible, an adapter class has been written, to allow an easy exchange, e.g. of a certain corpus or library.

2.5 Evaluation Data

A set of evaluation data has been created to allow a quantitative analysis of the project's algorithms. A total of 60 natural language queries have been manually selected from the *Answerbag*[10] platform. For all queries, the computer science domain has been chosen, allowing a better decision of what is relevant and which summary is appropriate. The query collection consists of 20 definition questions, 20 factoid questions and 20 list questions that can be found in appendix A.1. The definition of the three question types follows Liu et al. (2008).

Up to 20 answers per query and data source have been retrieved using the retrieval engine. Every answer document comes with the answer text, the type of data source, a ranking score and the URL of the source page. The corresponding question title and text (only for FAQs and Yahoo! Answers) are also saved. The query and the corresponding answers are stored in an XML file. Figure 2.4 shows an example document from Yahoo! Answers for the query *"What is flash media?"* (`defQ-0`).

Some of the answers have been manually annotated to create an evaluation corpus. The relevance of each sentence in the answer text has been annotated with 0 if it is irrelevant, 1 if it is possibly relevant (i.e. interesting) and 2 if it is definitely relevant. Definitely relevant sentences provide crucial information for the query, like the definition that Flash is a "multimedia plug-in for viewing rich media content on the internet" (sentence 3 of the evaluation query *"What is flash media?"*). Interesting sentences contain supporting information, like that the installation of flash is "quick and easy" (sentence 6).

Besides that, the topic of each document has been annotated by assigning the same integer to similar documents and different integers to two documents of opposed topics. Table 2.1 shows two annotated example documents, again for query `defQ-0`. *Part* determines if the sentence appears in the question title (1), question text (2), answer title (3) or answer text (4). Only the parts 3 and 4 have been annotated, since only they can be used in the summary. The first document is about *Adobe Flash* and got cluster id 1, the second is about *flash memory sticks* and is assigned to cluster 2. Appendix A.2 shows the complete list of queries that have been manually annotated for the evaluation.

[10] Answerbag – `http://www.answerbag.com`

```
<retrieved_answer>
  <question_id>396545676-ComputerNetworking_370-18</question_id>
  <question_title>when i go to some sites on pc its says i need to
    download media flash player ive tired this and still
    noting?</question_title>
  <answer_id>20060730135058AAhnqGF</answer_id>
  <answer_text>Go here to download the Flash Player (but uncheck the box
    to install the Yahoo toolbar): http://www.adobe.com/shockwave/download
    /download.cgi?P1_Prod_Version=ShockwaveFlash

    Once you have it installed correctly, you should see a Flash image
    saying "Adobe Flash Player Successfully Installed".</answer_text>
  <source_id>ya</source_id>
  <source_name>Yahoo!  Answers</source_name>
  <source_url>http://answers.yahoo.com/question/
    ?qid=20060730135058AAhnqGF</source_url>
  <rank>16</rank>
  <score>0.28571427</score>
  <category>Computer Networking</category>
</retrieved_answer>
```

Figure 2.4: Example document for the query “What is flash media?” in its XML representation

No.	*Part*	*Relev.*	*Cluster*	*Text*
1	1			What is Flash?
2	1			Where can I get it?
3	4	2	1	Macromedia Flash is a powerful multimedia plug-in that allows you to view rich media content on the internet.
4	4	2	1	While most computer systems and browsers come pre-packaged with Flash, your configuration may vary.
5	4	0	1	EyeCron requires that you have Flash 5 installed on your computer.
6	4	1	1	Installing Flash from Macromedia’s web site is quick and easy.
7	4	0	1	Simply Click Here and follow the instructions to install the latest version of Flash on your computer.
8				
9	1			Q1.
10	1			How is Memory Stick superior to other flash memory media?
11	4	0	2	A1.
12	4	2	2	Memory Stick is designed to be a network media that enables off-line connection of a variety of different devices.
13	4	1	2	Compatible with MagicGate copyright protection technology, it also provides a secure platform for enjoying copyright-protected content.
14	4	2	2	With these advantages, Memory Stick media is fit for use with multitudes of applications, from PC (IT) to CE products, mobile phones, car AV devices and other products, and will stimulate the development of many more attractive products in the future.

Table 2.1: Example annotations of the evaluation corpus for the query “What is flash media?”

2.6 Related Work

There are plenty of Question Answering systems that have been developed both in scientific and commercial domains. The *START*[11] platform claims to be the first web-based Question Answering system. It was developed at the MIT Artificial Intelligence lab in 1993 and is described in Katz (1997) and Katz et al. (2002). Each query is transformed by the system to a structured request (an Object–Property–Value expression) and then processed by OmniBase, which looks up certain data sources for information—both in local databases and on the Web. Due to deep linguistic analysis, the system usually produces high quality answers, but fails to answer questions that could not be analyzed by the system. The query *"What is a 'mashup'?"* (`defQ-3`) for instance does not return a result at all. Apart from that, the system is able to answer factoid, definition and list questions. If multiple data sources contain relevant information, their result is presented in individual sections. For ambiguous questions like *"What is flip-flop?"* (`defQ-13`), an excerpt of the Wikipedia disambiguation page is shown (if available).

Another system is *AnswerBus*[12] as introduced in Zheng (2002). *AnswerBus* is able to process queries in multiple languages, while the answer is a ranked list of English sentences. The system uses 5 different search engines and web directories to retrieve the answer data. The rate of correct answers is claimed to be 70.5% for the TREC-8 question answering evaluation corpus. As only individual sentences are returned, the system is not capable of answering definition questions.

Although many similarities between both the systems and this thesis exist, there are also some differences: Answer data for the thesis should be taken from heterogeneous data sources, which is not the case in *AnswerBus* (all the search engines and web directories offer web links in the same manner—based on a keyword search). Answers from different sources are presented individually as sections (*START*) or list items (*AnswerBus*). The thesis however aims to combine the results and present an interweaved summary of all the answers from different data sources. The system should also be able to answer arbitrary open-domain questions, which is not the case in *START* due to deep linguistic analysis. Ambiguity will be coped with topic clustering that is neither applied in *START* nor *AnswerBus*.

Ask.com[13] and *True Knowledge*[14] are two commercial systems. The Question Answering application of *Ask.com* works like a search engine and displays a list of possible web links that may contain the answer to the given natural language query. The engine processes the whole web index for equal or similar questions and tries to extract the part of the web page that contains the answer. For the example query *"What is flash media?"* (`defQ-0`) the top result is for instance:

[11] SynTactic Analysis using Reversible Transformations (START) – `http://start.csail.mit.edu`
[12] AnswerBus – `http://www.answerbus.com`, currently out of order (as of 03/2009)
[13] Ask.com – `http://www.ask.com`
[14] True Knowledge, The Internet Answer Engine – `http://www.trueknowledge.com`

> A multimedia package that allows the user to create animation, images and short graphical films for a website. Flash can bring your website to life and create a more entertaining experience for the visitor. Visually, Flash can provide a web...
> http://www.impact-direct.com/seo-faq.asp

A list of related topics is also displayed, allowing to further define the requested result of ambiguous queries. The main difference to this thesis is the absence of a multi-document summarization of the search result and the usage of a broader retrieval corpus (the *Ask.com* search engine index).

True Knowledge is a new platform that has not yet been published. Its goal is to provide a perfect inline response for the natural language query (i.e. a single fact, phrase or sentence) by using a large knowledge base. The knowledge base is both populated from existing web resources and user inputs or corrections. If the query could not be analyzed, a list of hits is displayed like in common search engines and the user has the possibility to add the necessary information to the knowledge base. For ambiguous queries, the system guesses the most likely sense. Obviously, the difference to this thesis' system is the short inline response. As this is not very useful for complex or definition questions, a summary of about 2–3 paragraphs will be tried here, together with a list of the most frequent n-grams (for factoid and list questions). Apart from that, the user will be offered the possibility to choose the data sources, which are used in the summary.

Another type of system is the so called *Social Q&A* platform. These Web 2.0 applications allow the users to submit arbitrary questions to be answered by the community. Although these platforms are not directly Question Answering systems, they allow a search for similar questions and thus also present the corresponding answers. Passage extraction, topic clustering or summarization as addressed in this thesis is not part of these systems. Many other systems have been developed, like *askEd!*, *TellMe QA* or *Wikiferret*, but will not be explained in detail.

Some Question Answering systems (or components of those) have been patented. Murata (2008), US patent 7,444,279, claims a Question Answering system that selects answer candidates based on the frequency of contained phrases that are ranked by their evaluation points. In Masuichi et al. (2008), US patent 7,461,047, a system is claimed that transforms the query to an answer pattern and searches for suitable results matching the pattern. Query expansion is also used here to ensure a broader basis of results. Although both the patents describe complete systems, none of the specific methods are used in this thesis' system. Answer ranking will be performed by exploiting the position within the document and the similarity between answer and query as well as answer and answer. As regards answer retrieval, only Lucene is currently used to retrieve suitable documents—there is no in-depth linguistic analysis of the query.

A different system is introduced in Yoshimura et al. (2008), US patent 7,418,443, which claims the generation of a tree-structure that consists of a list of retrieved candidate answers. This structure is then compared to the corresponding tree-structure of the query and

their similarity is evaluated. The resulting similarity scores are used to rank the candidate answers. This system again relies on the deep linguistic analysis of query and answers, which is not performed in this thesis. Although a graph representation of the candidate passages is built during answer ranking and topic clustering, the structure is not related to the claimed method.

US patent 7,454,393, Horvitz et al. (2008), addresses information extraction of large unstructured corpora like the World Wide Web. A retrieval and extraction method is claimed that is based on query reformulation and statistical models. The main idea is to predict the accuracy of the retrieval engine for a reformulated query, aiming in an optimized number of reformulations. Even though statistical models will be applied during passage extraction, neither query reformulation nor optimization of the document retrieval based on the statistical model is used here.

A passage extraction method is claimed in US patent 7,236,968, Seki et al. (2007). The technique uses a likelihood scoring of relevant passages. As the passage extraction component of this thesis includes filtering based on the calculation of semantic similarity and afterwards refining the result using a Hidden Markov Model, the claimed method is not affected. In Vanderwende et al. (2008), US patent 7,430,504, a graph representation of a text fragment is built and then scored. The graph consists of word nodes that are linked with their relation in the text fragment. The relation set involves linguistic constituents like subject or object, as well as semantic relations like first name and academic title. The scores are derived from the number and targets of the directed edges within the graph. They can then be used for summarization or ranking tasks. The basic idea of this patent follows the Biased LexRank method that is applied in the answer ranking task. Biased LexRank also builds a graph out of a source document and relies on the relations between the different graph nodes. The method however works on sentence level and includes only relations based on the semantic similarity of the sentences.

Chapter 3

Document Preprocessing and Text Similarity

This chapter describes the preprocessing steps that are used for further analysis of the retrieved answer documents. This includes segmentation, specificity annotations and word sense disambiguation. Moreover, several methods for the calculation of semantic text similarity are discussed.

3.1 Linguistic Levels and Segmentation

To allow an in-depth analysis of the documents, different linguistic levels need to be considered:

Definition 1 (Linguistic Levels) Each *document* (i.e. the *retrieved answer*) may consist of a question title, question text, answer title and answer text that will be referred to as *retrieved answer parts*. Every part can contain multiple *paragraphs* (divided by two or more line delimiters), every paragraph can contain multiple *sentences* (divided by sentence terminators) and every sentence consists of one or more *tokens*, which are the smallest unit that is considered in this thesis.

Sometimes it is useful to analyze also syntactic structures, so called *syntactic constituents*. The main focus lies on *noun phrases*, which consist of a head noun or pronoun and corresponding articles or adjectives, as of the definition in Carstensen et al. (2004). After segmenting the text, every token is reduced to its base form, called *lemma* in linguistics. The word class, i.e. the *part-of-speech* is also annotated, allowing to concentrate on certain classes only—especially nouns will be analyzed in more detail.

Figure 3.1 shows some examples for the above linguistic levels. The document is part of the FAQ data of query *"What is flash media?"* (`defQ-0`). The upper retrieved answer part is the question text; the lower is the answer text (no titles in this document). There are a total of 3 paragraphs, 7 sentences and 77 tokens, of which example annotations are shown.

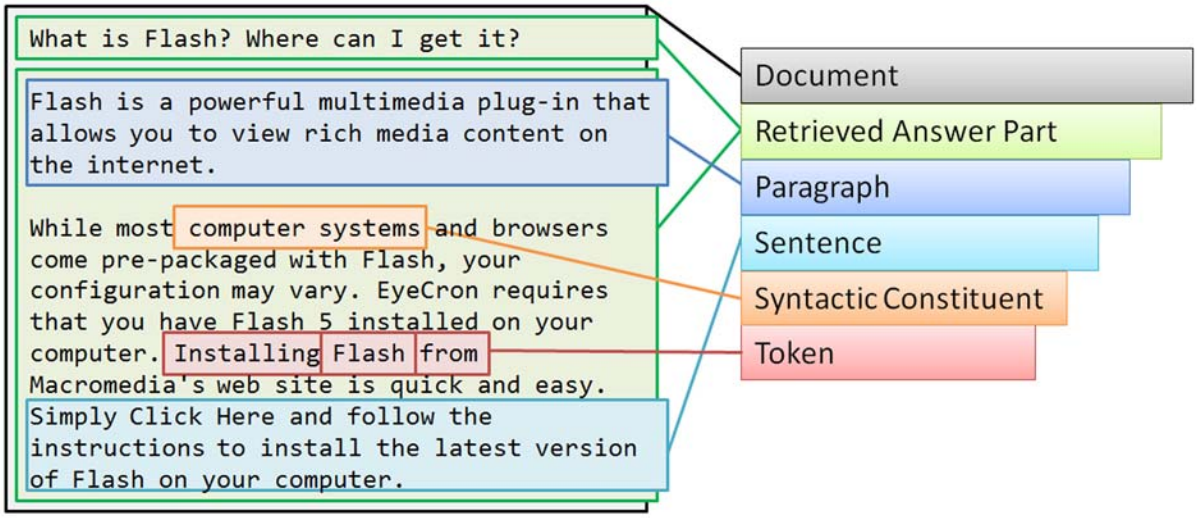


Figure 3.1: Example of different linguistic levels

For segmentation, both the sentence splitter and the tokenizer of the DKPro[15] implementation are used, which provide an extension of JAVA's break iterator. The annotators for part-of-speech tagging and lemmatization in DKPro are also used. These components incorporate Helmut Schmid's TreeTagger,[16] introduced in Schmid (1994).

To avoid noise in the analysis process, very common tokens like *a*, *is*, *I* etc. should not be considered. These words are commonly referred to as stop words. This project uses fixed word lists to remove stop word tokens.

3.2 Word Specificity

Taking into account the *specificity* of a word, often leads to better results in text comparison and similarity calculation. The more specific a term is, the better it describes the focus of a document. Specific terms are rare and usually restricted to a certain domain (e.g. *petunia* or *part-of-speech*), unlike generic terms (e.g. *plant* or *word*) that tend to appear in many documents of different type and topic. A general definition of specificity can be found in Spärck Jones (1972).

According to Salton et al. (1983), the *inverse document frequency* (IDF) can be used as a good measure for specificity. It is defined as the logarithm of the total number of documents N in a document collection, divided by the number of documents n_k that contain a word k. Since the document collection for a single query is very small (currently 60 documents per query), a large corpus will be used to calculate a specificity value like the one suggested in Mihalcea et al. (2006). More precisely, the Wortschatz[17] corpus, with 1

[15] Darmstadt Knowledge Processing Repository, Müller et al. (2008)
[16] TreeTagger – http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger
[17] Leipzig University "Wortschatz", Quasthoff et al. (2006) – http://corpora.uni-leipzig.de

million sentences from English newspapers, is used by the SPECIFICITY ANNOTATOR component to save a specificity value for every token of the document collection according to the following definition:

Definition 2 (Specificity) Let C be a large corpus and w a token. The specificity of token w is:

$$spec(w) = \frac{1}{\log |C|} \cdot \log \frac{|C|}{freq(w, C)} = 1 - \frac{\log freq(w, C)}{\log |C|}$$

with $freq(w, C)$ returning the number of occurrences of w and $|C|$ returning the corpus size, i.e. the total number of words in the corpus. Note that this formula uses word frequencies instead of document frequencies and thus differs from the classical IDF. The normalization factor $\frac{1}{\log |C|}$ is used to obtain a value in $[0, 1]$.

3.3 Word Sense Disambiguation

Some similarity measures require *word senses* rather than tokens as input data. These word senses (*word meanings* or *concepts*) are primitive semantic constituents of a document. A word can have multiple senses:

The word *"bank"* can e.g. be used in the sense of

- a financial institution (*"He transferred his money to another bank"*) or
- a slope next to a river (*"He sat on the bank of the river"*)

depending on the context.

Since the measures require the most suitable word sense, an automatic sense disambiguation needs to be done. The approach of Lesk (1986) will be used here. *Lesk's algorithm* determines the word overlap between a context and the sense gloss (i.e. definition) of possible concepts and takes the concept with the highest overlap. As a tie breaker, the more commonly used concept should be chosen.

The sense annotations are generated by the WORDNET SENSE ANNOTATOR component for every noun, verb, adjective and adverb, which is not a stop word. All lemmata of tokens in the surrounding document (again only non-stop words) are used as context, while the concepts and sense glosses are retrieved from the WordNet[18] dictionary. As the WordNet senses are ordered according to their commonness, the tie breaker chooses the sense with the smaller index.

[18] WordNet – http://wordnet.princeton.edu

3.4 Semantic Text Similarity

Text similarity is crucial for lots of natural language processing methods such as passage extraction, clustering and ranking. A variety of different similarity measures exist; cf. Mihalcea et al. (2006), Islam and Inkpen (2007) or Yang and Powers (2006) for a good overview. The following sections briefly describe the measures used in this thesis according to the following basic definition:

Definition 3 (Similarity Measure) Let s and t be tokens. A *Token Similarity Measure*

$$sim(s,t): (s,t) \mapsto d$$

returns the (normalized) similarity score $d \in [0,1]$ for the token pair (s,t). $d = 1$ means that s and t are equal, while $d = 0$ indicates that s and t have nothing in common.

Let $s_1, \ldots, s_m$ and $t_1, \ldots, t_n$ be tokens with $m, n \in \mathbb{N}$. The token sequences $S = s_1 s_2 \ldots s_m$ and $T = t_1 t_2 \ldots t_n$ are called *texts*. Analogously, a *Text Similarity Measure* is a function

$$sim(S,T): (S,T) \mapsto d,$$

that returns the (normalized) similarity score $d \in [0,1]$ for the text pair (S,T).

A text T has also a *set* character, with the element operator $s \in T$, indicating that s appears in the sequence T, the union operator $S \cup T$, returning a set of tokens that are in sequence S or T, and the intersection $S \cap T$, resulting in a set of tokens that are both in the sequences S and T.

3.4.1 Vector-based Similarity

A common set of similarity measures, used in Information Retrieval, is based on the Vector Space Model as defined in Salton et al. (1975) and Manning et al. (2008). The idea is to regard a text as a vector of word frequencies, a so called bag-of-words index. These vectors can be compared with standard approaches from linear algebra, like vector distance or the angle between the two vectors. The latter is known as *cosine similarity* and is defined formally as follows:

Definition 4 (Cosine Similarity) Let $S = s_1 s_2 \ldots s_m$ and $T = t_1 t_2 \ldots t_n$ be texts with $m, n \in \mathbb{N}$. Their *cosine similarity* score is the angle between their *TF-IDF* vectors:

$$sim_{cos}(S,T) = \frac{\sum_{w \in S \cup T} tfidf(w,S) \cdot tfidf(w,T)}{\sqrt{\sum_{w \in S} w^2} \cdot \sqrt{\sum_{w \in T} w^2}}$$

A TF-IDF value $tfidf(w,T) = freq(w,T) \cdot spec(w)$ of token w is the product of specificity (according to definition 2) and the number of times w appears in T, i.e. its word frequency $freq(w,T) = |\{t \in T : t = w\}|$. Words with a high TF-IDF value appear often in a certain document but seldom in the whole language and tend to provide good information about the content and the focus of the document. TF-IDF and cosine similarity have been defined e.g. in Salton and Buckley (1988).

3.4.2 Knowledge-based Similarity

Ontology-based similarity measures exploit the position of a concept within a given world-knowledge ontology. The basic idea of this similarity is that concepts gain a high similarity score if they are near to each other within the ontology hierarchy. As the ontology contains concepts rather than words, word senses need to be calculated as described in section 3.3.

Different token similarity measures are presented in the following. All the measures employ the ontology of the WordNet[19] dictionary. To provide a corresponding text similarity measure, the score of each token pair is maximized and weighted with the corresponding specificity (as of definition 2):

Definition 5 (Maximizing Token Similarity) Let S and T be texts and sim(s, t) a token similarity measure. The corresponding text similarity maximizes the similarity of each token pair and is called *Maximizing Token Similarity*:

$$sim_{max}(S,T) = \frac{\sum_{s \in S} \max_{t \in T}(sim(s,t)) \cdot spec(s)}{2 \sum_{s \in S} spec(s)} + \frac{\sum_{t \in T} \max_{s \in S}(sim(t,s)) \cdot spec(t)}{2 \sum_{t \in T} spec(t)}$$

This definition of maximizing token similarity has been taken from Mihalcea et al. (2006).

The first three ontology-based similarity measures evaluate the number of nodes and edges between two given concepts. The smaller the shortest path, the more similar two concepts are. The $depth(c)$ function can be used to retrieve the total depth of concept c in the hierarchy. Besides that, the *least common subsumer* (LCS) is considered, which is the nearest parent node within the ontology that both the concepts share; e.g. the LCS of *human* and *chimpanzee* is *primate* within the WordNet ontology.

Definition 6 (Quillian Similarity) Let s and t be concepts within an ontology, LCS their least common subsumer and D the maximum depth[20] of the ontology. The *Quillian Similarity* of s and t is the number of edges on the shortest path between the concepts, normalized over the maximum ontology depth:

$$sim_{Quillian}(s,t) = \frac{2D - (depth(s) + depth(t) - 2\,depth(LCS))}{2D}$$

[19] WordNet, Fellbaum (1998) – `http://wordnet.princeton.edu`
[20] set to $D = 16$ for WordNet, according to Yang and Powers (2006)

Definition 7 (Leacock/Chodorow Similarity) Let s and t be concepts within an ontology, LCS their least common subsumer and D the maximum depth of the ontology. The *Leacock/Chodorow Similarity* of s and t is the logarithm of the number of nodes on the shortest path between the concepts, normalized over the maximum ontology depth:

$$sim_{Leacock}(s,t) = \frac{\log(2D) - \log(depth(s) + depth(t) - 2\,depth(LCS) + 1)}{\log(2D)}$$

Definition 8 (Wu/Palmer Similarity) Let s and t be concepts within an ontology and LCS their least common subsumer. *Wu/Palmer Similarity* is defined as the ratio of the depth of the nearest common node to the depth of both the target concepts:

$$sim_{WuPalmer}(s,t) = \frac{2 \cdot depth(LCS)}{depth(s) + depth(t)}$$

The measures have been introduced in Quillian (1967), Leacock and Chodorow (1998) and Wu and Palmer (1994), while the definitions have been taken from Yang and Powers (2006) and Mihalcea et al. (2006).

Common problems of those techniques, relying on the shortest path between two concepts, are unbalanced relations within the hierarchy. The term *skibob* in WordNet is e.g. 2 edges away from *military vehicle*, while *aircraft* is 3 edges from *military vehicle*. Intuitively an *aircraft* should be more similar to a *military vehicle* (if seen in the context of fighter jet, cruise missile etc.), than a *skibob* is to a *military vehicle*. The disequilibrium in hierarchy usually occurs if some parts of the hierarchy are more detailed and finer elaborated than others.

To overcome this issue, Ross (1976) and Resnik (1995) define the *Information Content* of a concept as:

$$IC(c) = -\log p(c),$$

with probability $p(c)$ of encountering and instance of concept c or any sub concept within an arbitrary text. The idea is, to use shared information of two concepts for a definition of similarity. After giving some remarks on how to calculate the information content, three IC-based similarity measures will be introduced.

The root concept in the hierarchy has $p(c) = 1$ and therefore $IC(c) = 0$, which means, that the root does not contain any information. The information content generally increases with increasing depth. Since WordNet does not offer such a probability, a supporting database has been created that maps each WordNet concept to its cumulated frequency within the Wortschatz[21] corpus. The WordNet hierarchy is therefore traversed in a depth-first order to aggregate the frequencies of child concepts to their parents. The frequency of a single concept is measured by the sum of word frequencies of all words in its synset (i.e. the concept in WordNet). Note, that the words are not yet disambiguated for the sake of

[21] Leipzig University "Wortschatz", Quasthoff et al. (2006) – http://corpora.uni-leipzig.de

simplicity and calculation time. It is assumed that every concept appears at least once, the frequency is therefore incremented by one (Laplace correction).

Consider e.g. concept $c =$ `SID-02472293-N`, which is expressed by the words *human, homo, man* and *human being*. The corresponding word frequencies in the Wortschatz corpus are 2 755, 2, 6 081, 3 329 and the sum of child concept frequencies is 12 365, thus the concept frequency of c is 24 533 (the sum of above frequencies plus 1). The probability $p(c) = 0.00082646$ is the quotient of concept frequency and the sum of all concept frequencies (which calculates to 29 684 421). Thus, the information content is $IC(c) = 7.098358492$.

Definition 9 (Resnik Similarity) Let s and t be concepts within an ontology, LCS their least common subsumer. The corresponding *Resnik Similarity* score is the information content of the least common subsumer:

$$sim_{Resnik}(s,t) = IC(LCS)$$

Definition 10 (Lin Similarity) Let s and t be concepts within an ontology, LCS their least common subsumer. The corresponding *Lin Similarity* score is the information content ratio of the least common subsumer and both the concepts:

$$sim_{Lin}(s,t) = \frac{2 \cdot IC(LCS)}{IC(s) + IC(t)}$$

Definition 11 (Jiang/Conrath Similarity) Let s and t be concepts within an ontology, LCS their least common subsumer. *Jiang/Conrath Similarity* calculates the reciprocal value of the concept differences:

$$sim_{Jiang}(s,t) = \frac{1}{IC(s) + IC(t) - 2 \cdot IC(LCS)}$$

The definitions have been taken from Resnik (1995), Lin (1998), Jiang and Conrath (1997) and Mihalcea et al. (2006).

3.4.3 Corpus-based Similarity

Another possibility to define similarity, is to make use of a large corpus and consider how often two words appear near to each other. It is e.g. more likely that *banana* appears near *fruit* than near *linguistics*, leading to a higher similarity score for *(banana, fruit)* compared to *(banana, linguistics)*.

Church and Hanks (1990) introduced such a measure: *Pointwise Mutual Information* (PMI). In the following we will consider a slight modification for the use of a web-based corpus, following Turney (2001) and Mihalcea et al. (2006):

Definition 12 (PMI-IR) Let s and t be tokens and *WebSize* the estimated size of the World Wide Web resp. the size of the search engine index. The *PMI-IR* similarity of s and t is defined as

$$sim_{PMI}(s,t) = \frac{hits(s\ NEAR\ t) \cdot WebSize}{hits(s) \cdot hits(t)},$$

where $hits(s)$ returns the number of search results in a large web search engine.

PMI-IR calculations in this thesis will use data from AltaVista[22] and *WebSize* will be set to 7e11.[23] As of definition 5, the Maximizing Token Similarity measure can be used to calculate the similarity of two texts.

Finally, the Wikipedia[24] can be exploited to determine the semantic similarity of two texts. According to Gabrilovich and Markovitch (2007), one possibility is *Explicit Semantic Analysis* (ESA):

Definition 13 (Wikipedia-ESA) Let S and T be texts. The corresponding Wikipedia-based ESA score $sim_{ESA}(S,T)$ can be calculated by interpreting the TF-IDF vectors of S and T semantically, using an inverted index of Wikipedia articles. The resulting weighted vectors are then compared.

The DKPro[25] implementation of ESA will be used in the following.

[22] AltaVista – http://www.altavista.com
[23] according to Chklovski and Pantel (2004)
[24] Wikipedia – http://en.wikipedia.org
[25] Darmstadt Knowledge Processing Repository, Müller et al. (2008)

Chapter 4

Passage Extraction

The passage extraction task aims to find sections and regions in the documents that contain relevant information for generating an appropriate answer to a query. This is achieved by gradually improving the quality of the results in a sequence of filtering and selecting components. The chapter first provides an overview of terms and techniques, before the components themselves are described and quantitatively evaluated.

4.1 Background

In order to formulate the goals of the passage extraction task, the term *passage* needs to be defined more precisely:

Definition 14 (Passage) A *passage* is a document, paragraph, sentence, phrase or token according to definition 1 within a collection of documents. Thus, a passage can be an arbitrary part at a certain linguistic level. A passage can be *relevant* or *irrelevant* with regard to the user's query.

A common approach to express the relevance of a passage is its similarity score to the current context of interest, which is e.g. proposed in Mihalcea et al. (2006). If the score is above a certain threshold θ, the passage is considered relevant and will be called *selected*. Otherwise the passage is irrelevant and will be called *filtered*. Since the relevance to the query should be calculated, the query text will be used as context.

The relevance of sentences has been manually annotated for the training dataset as described in section 2.5. These annotations will be used for evaluating the extraction results. Note that the annotation distinguishes *irrelevant*, *possibly relevant* (i.e. interesting) and *definitely relevant* passages. A passage that is manually annotated as either possibly relevant or definitely relevant, will be considered *relevant*. A paragraph however is considered relevant if at least one of its sentences is relevant.

A quantitative evaluation of the passage extraction task can be achieved by comparing the number of manually annotated relevant (resp. irrelevant) passages with the number of automatically selected (resp. filtered) passages. The four possibilities relevant-selected r_s, relevant-filtered r_f, irrelevant-selected i_s and irrelevant-filtered i_f can be analyzed in a *confusion matrix*:

	filtered	*selected*
irrelevant	i_f	i_s
relevant	r_f	r_s

The definition of the confusion matrix follows Kohavi and Provost (1998). A more detailed evaluation can be done if the annotations for possibly relevant (r_{1f} and r_{1s}) and definitely relevant (r_{2f} and r_{2s}) with $r_{1f} + r_{2f} = r_f$ and $r_{1s} + r_{2s} = r_s$ are considered. The confusion matrix then changes to:

	filtered	*selected*
irrelevant	i_f	i_s
possibly relevant	r_{f1}	r_{s1}
definitely relevant	r_{f2}	r_{s2}

Common measures like recall and precision are used to score the result quality; their definitions can be found e.g. in van Rijsbergen (1974) or Kohavi and Provost (1998).

4.2 Duplicate Document Remover

Sometimes the retrieval engine returns two documents with the same text. This happens mostly for FAQ documents, since webmasters tend to copy their FAQ from another existing website. But also two answers from Yahoo! Answers may have been copied. To speed up the analysis process and to avoid duplicate sentences in the summarized answer, these identical documents are removed.

The DUPLICATE DOCUMENT REMOVER component calculates an MD5 hash[26] for every document and saves the hash value in a set. If the set already contains such a hash, the document is removed and will not be considered in the following components. This method has proven to be simple, robust and fast, according to Ye et al. (2006). As the method is exact, there is no need for an evaluation with precision or recall values.

Figure 4.1 shows the number and percentage of duplicate documents in 8 example queries, which have been removed by the DUPLICATE DOCUMENT REMOVER. Both the values have been analyzed independently for each data source. Most duplicates have been found in the FAQ collection, while Yahoo! Answers contains almost half as much and Wikipedia has no duplicates.

[26] Message-Digest algorithm 5—defined in RFC 1321, Rivest (1992)

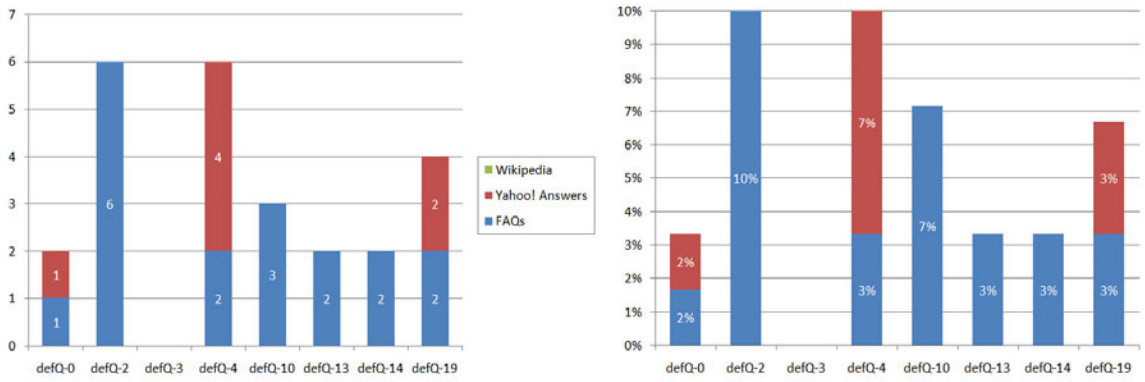


Figure 4.1: Number and percentage of duplicate documents per data source

4.3 Paragraph Filter

The PARAGRAPH FILTER component focuses on paragraph-level passages. Its goal is to remove as many irrelevant paragraphs as possible in both a fast and precise manner. It is not designed to remove *all* irrelevant passages, but rather to minimize the number of filtered-relevant paragraphs. The following table illustrates that goal:

	filtered	*selected*
irrelevant	i_f: MAX	i_s: don't care
relevant	r_f: MIN	r_s: don't care

A high number of filtered-irrelevant and a low number of filtered-relevant paragraphs lead to a high filter precision, which is chosen as evaluation measure for this component:

$$precision_{PGF} = \frac{i_f}{i_f + r_f}$$

As described in section 4.1, relevance classification can be done by calculating the similarity score between query and passage. If the score is above some threshold θ, the passage is regarded relevant (and irrelevant otherwise). Since e.g. not every word can be found in the WordNet database, it is useful to combine multiple similarity measures to get more stable results.

To provide a fast filter component, the calculation time of the similarity measures needs to be considered. Table 4.1 shows the calculation time[27] for the similarity measures described in section 3.4. All 458 paragraphs of the example query *"What is flash media?"* (`defQ-0`) have been used for the calculation.

It is not surprising that the PMI measure needs the longest time, since it has to fetch three web pages for every pair of tokens. Of course this can be further improved by caching the

[27] Calculation done on Intel Core2 Quad CPU, 2.4 GHz, 3 GB RAM, Windows Vista, Java 1.6.0.2

Similarity measure	*Calculation time*
Cosine	115ms
Quillian	1,566ms
Leacock/Chodorow	1,131ms
Wu/Palmer	1,029ms
Resnik	7,709ms
Lin	19,047ms
Jiang/Conrath	18,076ms
AltaVista-PMI	$\gg$ 200,000ms
Wikipedia-ESA	66,958ms

Table 4.1: Similarity score calculation time for 458 training data paragraphs

	Quillian	*Leacock*	*WuPalmer*	*Resnik*	*Lin*	*Jiang*
Quillian	1.00000	0.95478	0.95866	0.94391	0.92540	0.89830
Leacock	0.95478	1.00000	0.98730	0.98127	0.98404	0.97239
WuPalmer	0.95866	0.98730	1.00000	0.97327	0.98397	0.94952
Resnik	0.94391	0.98127	0.97327	1.00000	0.97961	0.97239
Lin	0.92540	0.98404	0.98397	0.97961	1.00000	0.98377
Jiang	0.89830	0.97239	0.94952	0.97239	0.98377	1.00000

Figure 4.2: Correlation coefficients for WordNet-based similarity measures

results—down to 431ms if all values are in the cache. For this simple filtering component however, PMI will not be considered. Also the ESA calculation, using a Wikipedia index, needs much time and is therefore excluded from the PARAGRAPH FILTER component.

Looking at the definition of the WordNet-based similarity measures, one can suspect a correlation between those relying on path length and those relying on information content. Figure 4.2 shows the correlation coefficient between the WordNet measures on the training data of `defQ-0`. As expected, the correlation is quite high for most pairs of measures. The PARAGRAPH FILTER component will therefore use only the three WordNet-based measures Quillian, Leacock/Chodorow and Resnik, while the measures by Wu/Palmer, Lin and Jiang/Conrath are omitted due to their high correlation to the others.

For the remaining similarity measures, the threshold θ is optimized on the paragraphs of the `defQ-0` training dataset. Table 4.2 shows the classification performance. With the restriction, that no relevant paragraph may be filtered ($r_f = 0$), θ is rather small and thus does not filter very much (irrelevant) paragraphs. If however one relevant paragraph may be filtered out, precision still is 99%, while the number of filtered paragraphs increases significantly. Because of that, θ is chosen to allow filtering a small number of relevant passages. Further increasing of the threshold can lead to a large decrease of precision, which is to be avoided in this component.

Cosine Similarity

θ	i_f	r_f	$precision_{PGF}$
0.00	0	0	100.00%
0.06	189	1	99.47%
0.10	205	3	98.95%
0.13	230	5	97.87%

Leacock/Chodorow Similarity

θ	i_f	r_f	$precision_{PGF}$
0.28	63	0	100.00%
0.51	212	1	99.53%
0.55	235	3	98.73%
0.59	261	6	97.75%

Quillian Similarity

θ	i_f	r_f	$precision_{PGF}$
0.65	62	0	100.00%
0.76	147	1	99.32%
0.79	181	2	98.90%
0.82	208	5	97.65%

Resnik Similarity

θ	i_f	r_f	$precision_{PGF}$
0.01	28	0	100.00%
0.18	210	1	99.52%
0.27	224	3	98.67%
0.30	259	6	97.73%

Table 4.2: Optimized thresholds for the Paragraph Filter component

Since Quillian's method has the lowest number of filtered passages, it will not be considered for this component any further. After fine tuning on a second dataset (`defQ-4`) the following thresholds are used for the PARAGRAPH FILTER component:

- Cosine similarity: $\theta = 0.065$,
- Leacock/Chodorow similarity: $\theta = 0.513$,
- Resnik similarity: $\theta = 0.06$

Paragraphs with at least one similarity score below the corresponding threshold are considered irrelevant and will therefore not be used in the following components anymore.

4.4 Sentence Selector

As the name suggests, the SENTENCE SELECTOR component works on the sentence level of the retrieved documents. The goal is to select as many relevant sentences as possible, which means obtaining a high recall for the selection process. Moreover, the objective for a high filter precision introduced in the PARAGRAPH FILTER remains. The corresponding confusion matrix looks as follows:

	filtered	*selected*
irrelevant	i_f: MAX	i_s: don't care
possibly relevant	r_{f1}: MIN	r_{s1}: MAX
definitely relevant	r_{f2}: MIN	r_{s2}: MAX

AltaVista-PMI

	filtered			*relevant*			*definitely relevant*		
θ	i_f	r_f	$precision_{STS}$	r_f	r_s	$recall_{STS}$	r_{f2}	r_{s2}	$recall_{2,STS}$
0.000	0	0	100.00%	0	114	100.00%	0	58	100.00%
0.034	76	1	98.70%	1	113	99.12%	0	58	100.00%
0.152	129	2	98.47%	2	112	98.23%	0	58	100.00%
0.165	131	3	97.76%	3	111	97.35%	0	58	100.00%
0.190	142	4	97.26%	4	110	96.46%	1	57	98.25%

Wikipedia-ESA

	filtered			*relevant*			*definitely relevant*		
θ	i_f	r_f	$precision_{STS}$	r_f	r_s	$recall_{STS}$	r_{f2}	r_{s2}	$recall_{2,STS}$
0.002	35	0	100.00%	0	114	100.00%	0	58	100.00%
0.012	70	1	98.59%	1	113	99.12%	0	58	100.00%
0.018	104	2	98.11%	2	112	98.25%	1	57	98.28%
0.020	118	3	97.52%	3	111	97.37%	2	56	96.55%
0.024	139	4	97.20%	4	110	96.49%	2	56	96.55%

Table 4.3: Optimized thresholds for the Sentence Selector component

The optimization distinguishes between possibly relevant and definitely relevant. Definitely relevant sentences should not be filtered at all (i.e. $r_{f2} = 0$), where possible. Optimization performance is again monitored with precision and recall:

$$recall_{STS} = \frac{r_s}{r_s + r_f}, \qquad recall_{2,STS} = \frac{r_{s2}}{r_{s2} + r_{f2}}, \qquad precision_{STS} = \frac{i_f}{i_f + r_f}.$$

During paragraph filtering, calculation time was an issue to gain a fast filtering of irrelevant passages. These passages will not be processed in the SENTENCE SELECTOR component, thus allowing a more detailed and complex analysis method. Particularly, the measures, that have not been used in PARAGRAPH FILTER due to their time consuming calculation, can be considered now.

Gabrilovich and Markovitch (2007) introduce *Explicit Semantic Analysis* (ESA) using data from Wikipedia (cf. definition 13 for implementation details) and promise a correlation between 0.60 and 0.72 for computed relatedness with human judgments. Since this means a large improvement compared to WordNet-based measures, Wikipedia-ESA will be used for the SENTENCE SELECTOR.

As a second similarity measure, *Pointwise Mutual Information* will be used. The evaluation in Mihalcea et al. (2006) uses data from AltaVista resulting in an F-measure of 81.0% on the Microsoft paraphrase corpus,[28] which seems very auspicious. Implementation details can be found in definition 12.

[28] Microsoft paraphrase corpus – http://research.microsoft.com

AltaVista-PMI	*filtered*	*selected*
irrelevant	145	678
possibly relevant	2	53
definitely relevant	1	54

Wikipedia-ESA	*filtered*	*selected*
irrelevant	108	715
possibly relevant	1	54
definitely relevant	1	54

Sentence Selector	*filtered*	*selected*
irrelevant	190	633
possibly relevant	3	52
definitely relevant	1	54

Table 4.4: Confusion matrix of Sentence Selector results

Applying WordNet similarity measures once more does not seem to be a good idea, since they have already been used on the same data by the PARAGRAPH FILTER component and would thus not reveal much new information. The same applies to cosine similarity.

Table 4.3 shows the performance of AltaVista-PMI and Wikipedia-ESA with different thresholds θ. The example query *"What is flash media?"* (`defQ-0`) was used to generate the results. Wikipedia-ESA is able to filter 35 irrelevant sentences without filtering a single relevant when using a small threshold. Unfortunately, when increasing the threshold, the number of filtered definitely relevant sentences (r_{f2}) increases, such that $recall_{2,STS}$ drops first below 99% and then below 97%. Setting θ to 0.017, 2 relevant sentences (1 possibly relevant, 1 definitely relevant) and 101 irrelevant sentences are filtered. A total of 829 sentences is selected (of which 717 are still irrelevant).

AltaVista-PMI allows much higher thresholds before filtering a definitely relevant sentence. Not until a threshold of $\theta = 0.19$, the $recall_{2,STS}$ score drops below 100%. Setting θ to 0.15, only 2 relevant sentences (0 definitely relevant) and 127 irrelevant sentences are filtered. A total of 792 sentences is selected (of which 681 are irrelevant).

Table 4.4 shows the confusion matrix for the chosen thresholds:

- AltaVista-based PMI similarity: $\theta = 0.15$,
- Wikipedia-based ESA similarity: $\theta = 0.017$

The results are combined by considering a passage irrelevant (filtered) if one of the similarity scores is below its threshold. For the `defQ-0` training data set, these results can also be found in table 4.4.

4.5 Passage Extent Determination

So far only the similarity score between passage and query has been considered for classifying a passage relevant or irrelevant. Every passage is regarded independently from the context it appears in. A more natural approach would however find relevant passages near other relevant passages and irrelevant passages near other irrelevant passages, thus creating consecutive chains of relevant and irrelevant structures.

Table 4.5 shows 9 example sentences, that have been manually annotated relevant or irrelevant (column *"expected"*). After that, the similarity scores of AltaVista-based PMI and Wikipedia-based ESA have been calculated and used by the SENTENCE SELECTOR component to classify the sentences relevant or irrelevant (column *"actual"*—selected indicates relevant, filtered indicates irrelevant passages). Recall from section 4.4, that a sentence is selected if both the thresholds 0.15 for PMI and 0.017 for ESA have been exceeded.

There are two classification errors in the example: sentence 4 is relevant but was filtered (i.e. classified irrelevant) and sentence 8 is irrelevant but was selected (classified relevant). Such errors generally happen to appear if many stop words, prepositions or ambiguous words are used. The sentence for instance consists of the lemmata *"long"*, *"different"*, *"browser"*, *"equip"*, *"necessary"* and *"plug-in"*, which are not very much related to the query *"What is flash media?"*. Sentence 8 however contains the buzz words *"Java"* and *"applet"*, which lead to a high similarity score with *"flash"* and *"media"*.

The errors can possibly be resolved, by taking into account the sentence's context. This would include promoting sentence 4 to be selected because of the sentences 2–5 as well as filtering sentence 8 because of the sentences 6–9. Sentence 1 should however remain filtered, although the sentences 2 and 3 are selected.

No.	*Sentence*	*expected*	*PMI*	*ESA*	*actual*
1	Answer:	irrelevant	0.000	0.000	filtered
2	Flash was known as FutureSplash until 1997, when Macromedia Inc. bought the company that developed it.	relevant	0.630	0.057	selected
3	Flash animations will look the same in all browsers.	relevant	0.572	0.096	selected
4	As long as different browsers are equipped with the necessary plug-ins.	relevant	0.068	0.017	filtered
5	This can be downloaded here from the Macromedia web site.	relevant	0.471	0.053	selected
6	here are some good links about that:http://www.drivermagician.com/flashplayer/	irrelevant	0.236	0.001	filtered
7	-- http://www.proud-collector.com/	irrelevant	0.067	0.002	filtered
8	It's only competitor is Java applets.	irrelevant	0.295	0.019	selected
9	Note that I've talked only about _client_ evironments.	irrelevant	0.094	0.009	filtered

Table 4.5: Example for the Passage Extent Determination component

Based on these observations, the PASSAGE EXTENT DETERMINATION component is introduced. Its goal is to refine the results of the SENTENCE SELECTOR by considering consecutiveness of relevant or irrelevant passages. He et al. (2004) describes a Passage Retrieval technique based on a Hidden Markov Model that was submitted to the HARD 2004[29] track of the TREC conference. Their method is called *Passage Extent Determination* and will partly be used for this component. A formal definition of the Hidden Markov Model and its parameterization is given in the following:

Definition 15 (Hidden Markov Model) Let $S = \{s_1, \ldots, s_N\}$ be a set of states, q_t the current state at time t and $V = \{v_1, \ldots v_M\}$ a set of symbols. A *Hidden Markov Model* (HMM) $\lambda = (A, B, \pi)$ consists of

- a matrix $A = \{a_{i,j}\}$ of transition probabilities $a_{i,j} = Pr[q_t = s_j \mid q_{t-1} = s_i]$ from state s_i to state s_j,
- a function vector $B = \{b_j\}$ of emission probabilities $b_j(k) = Pr[v_k \mid q_t = s_j]$ for emitting symbol v_k at time t in state s_j and
- a vector $\pi = \{\pi_i\}$ of initial probabilities $\pi_i = Pr[q_1 \mid s_i]$ for the HMM to be in state s_i at time $t = 1$.

The original idea of Hidden Markov Models can be found in Baum and Petrie (1966), Baum and Eagon (1967), Baum et al. (1970) and Baum (1972), while the notation above was taken from Rabiner (1989)—an early application of HMMs in automatic speech recognition.

According to this notation, $O = O_1 O_2 \ldots O_T$ with $O_j \in V$ denotes an observation sequence (i.e. a sequence of observed symbols) and $Q = q_1 q_2 \ldots q_T$, $q_i \in S$ denotes the corresponding sequence of (hidden) states, the HMM is in. The goal of the Hidden Markov Model is to find an optimal state sequence Q to a given observation sequence O. The optimal Q has maximum probability $Pr[Q \mid O, \lambda]$ and can be calculated with Viterbi's algorithm, introduced in Viterbi (1967).

The next section describes the configuration of the HMM to fulfill the passage extent determination task. Then, scaling of the sentence similarity scores is addressed and the HMM is tested on the training data.

4.5.1 Model Configuration

Following the approach of He et al. (2004), the states $S = \{s_1 = \textit{irrelevant}, s_2 = \textit{relevant}\}$ are used. The probability for two consecutive irrelevant states $a_{1,1} = 0.96$ and for two consecutive relevant states $a_{2,2} = 0.87$. This leads to $a_{1,2} = 0.04$ for an irrelevant passage

[29] High Accuracy Retrieval from Documents (HARD) 2004 – http://projects.ldc.upenn.edu/HARD

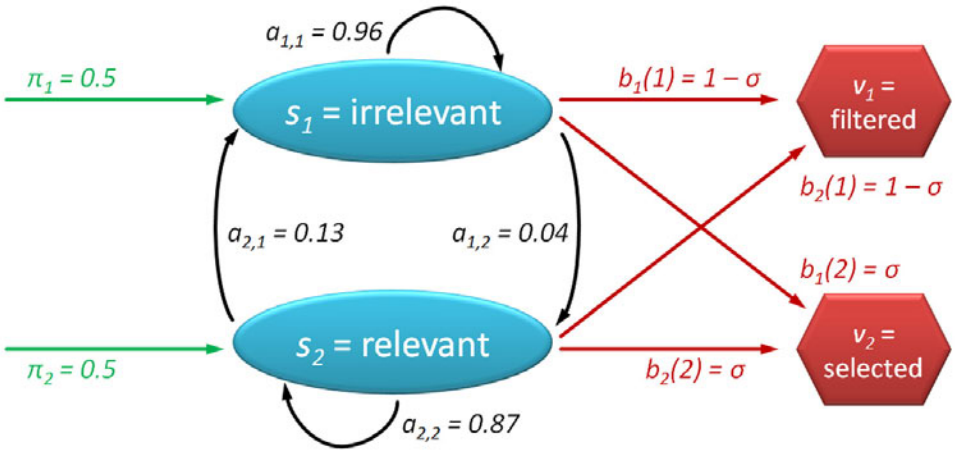


Figure 4.3: Configuration of the Hidden Markov Model used for Passage Extent Determination

following after a relevant one and $a_{2,1} = 0.13$ for a relevant passage after an irrelevant one. The probabilities have been determined in He et al. (2004) by training on a mixture of the HARD 2003 and HARD 2004 data. Although a new training could be useful, the system should benefit from the original heterogeneous document collection rather than applying this thesis' training data once more. Since no particular distribution of relevant and irrelevant passages at the beginning of a document could be found, the initial probabilities are set to $\pi = \{0.5, 0.5\}$.

As symbol set $V = \{v_1 = \textit{filtered}, v_2 = \textit{selected}\}$ is defined, indicating that a passage was considered irrelevant or relevant by the previous components. The similarity scores between passage and query are used as emission probabilities $b_j(k)$, which can however not be used directly (apart from He et al. (2004)'s method), because of different thresholds θ for every measure. To cope with that, a function $\phi : [0,1] \longrightarrow [0,1]$ is defined here to scale the similarity score and thus transform it to a 'real' probability value. The chosen function should fulfill:

- $\phi(0) = 0$ "No emission of v_2 = selected if similarity is 0"
- $\phi(1) = 1$ "No emission of v_1 = filtered if similarity is 1"
- $\phi(\theta) = 0.5$ "Emission of v_1 and v_2 is equiprobable",
- ϕ is monotonous "higher similarity scores lead to a higher emission probability".

The average value of the scaled AltaVista-PMI and the scaled Wikipedia-ESA is then taken as emission probability: $b_2(2) = \sigma$ and $b_2(1) = 1 - \sigma$ if the system is in state $s_2 = \textit{relevant}$; $b_1(1) = 1 - \sigma$ and $b_1(2) = \sigma$ if in state $s_1 = \textit{irrelevant}$ with

$$\sigma = \frac{\phi_{PMI}(sim_{PMI}) + \phi_{ESA}(sim_{ESA})}{2}$$

Figure 4.3 shows the configuration of this Hidden Markov Model graphically.

4.5.2 Different scaling functions

Experiments showed that different scaling functions ϕ can lead to quite different results. Therefore, 6 distinct variants are defined and tested in the following.

A simple interpolation method for the four constraints above is using *quadratic polynomials*. The result is a continuous function around threshold θ. The polynomial is however not necessarily monotonous especially if θ is near the boundaries. The function is therefore set to 0.0 for negative values and 1.0 for values exceeding 1.0:

$$g(x) = \frac{0.5-\theta}{\theta^2-\theta}x^2 + \frac{\theta^2-0.5}{\theta^2-\theta}x, \qquad \phi_{poly2}(x) = \begin{cases} 0 & \text{if } g(x) \leq 0 \\ 1 & \text{if } g(x) \geq 1 \\ g(x) & \text{else} \end{cases}$$

Besides quadratic, also higher dimensional polynomial approximation could be interesting, but has not been considered here. The second group of scaling functions uses spline approximation. A simple non-continuous approach uses *linear splines*:

$$\phi_{spline1}(x) = \begin{cases} \frac{1}{2\theta}x & \text{if } x \leq \theta \\ \frac{2\theta-1}{2\theta-2} - \frac{1}{2\theta-2}x & \text{if } x > \theta \end{cases},$$

while a more elaborate (and continuous) function makes use of *quadratic splines*:

$$\phi_{spline2}(x) = \begin{cases} \frac{1}{\theta}x - \frac{1}{2\theta^2}x^2 & \text{if } x \leq \theta \\ \frac{x^2-2\theta x+2\theta^2-2\theta+1}{2(\theta-1)^2} & \text{if } x > \theta \end{cases}$$

Both the spline functions are monotonous.

Apart from polynomial functions, a *power function* could be used for scaling, which leads to a continuous function, motivated by a growth model:

$$\phi_{power}(x) = x^{-\log 2/\log \theta}$$

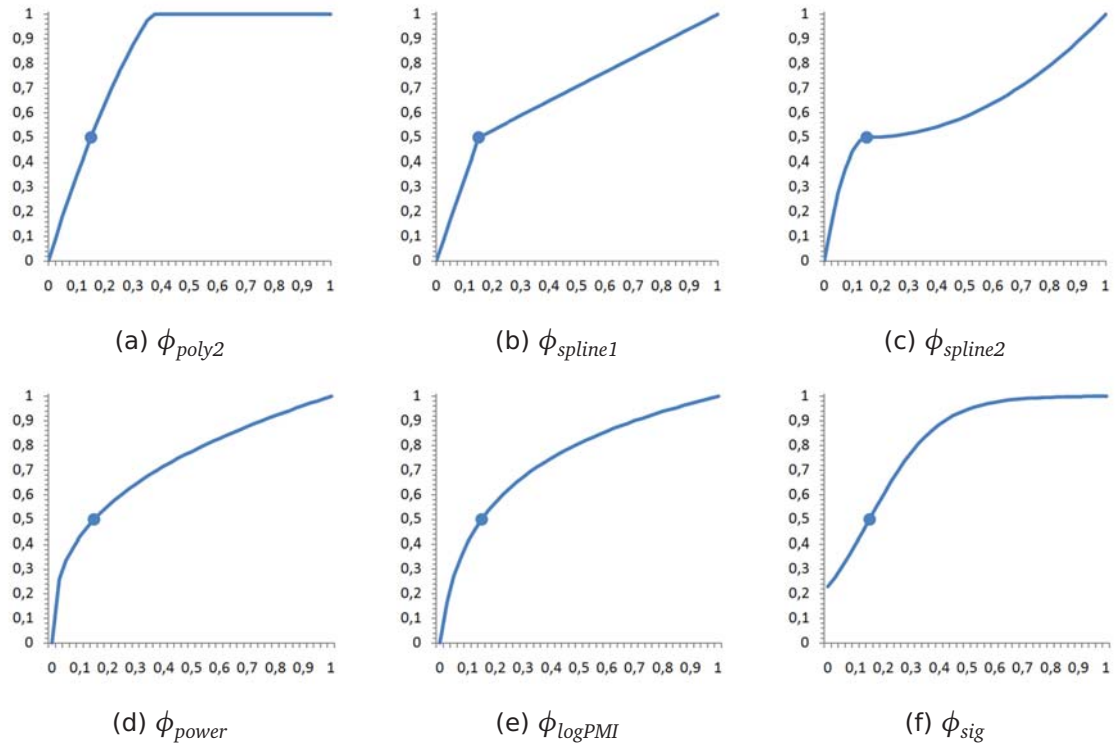
Thinking of the inverse function, motivates the usage of a logarithm function. Unfortunately, no closed form could be found, so a function has been numerically calculated in a Computer Algebra System for both the AltaVista-PMI and Wikipedia-ESA thresholds:

$$\phi_{logPMI}(x) = 0.288 \cdot \log(x \cdot e^{1/0.288} - x + 1)$$

$$\phi_{logESA}(x) = 0.123 \cdot \log(x \cdot e^{1/0.123} - x + 1)$$

Finally, the *sigmoid function* (or s-curve) from logistics will be tested. Note, that $\phi(0) = 0$ and $\phi(1) = 1$ does not necessarily hold for the s-curve.

$$\phi_{sig}(x) = \frac{1}{1 + \exp\left(\frac{x-\theta}{\theta-1}\log 1000\right)}$$



(a) ϕ_{poly2} (b) $\phi_{spline1}$ (c) $\phi_{spline2}$

(d) ϕ_{power} (e) ϕ_{logPMI} (f) ϕ_{sig}

Figure 4.4: Graph of the 6 scaling functions for a threshold θ of 0.15

Figure 4.4 shows the graph of all the functions for $\theta = 0.15$.

Recall the example from table 4.5. If the similarity scores are scaled with the quadratic polynomial interpolation function ϕ_{poly2}, the observation sequence $O = O_1 O_2 \dots O_T$ (and corresponding emission probabilities $b_j(k)$) can be calculated as shown in Table 4.6. If the Hidden Markov Model is then applied to this example, the sequence $Q = q_1 q_2 \dots q_T$ of hidden states is returned.

The model's result differs in only the two sentences $k = 4$ and $k = 8$. Sentence 4 was filtered by the SENTENCE SELECTOR, because of the low ESA score. As the surrounding sentences are considered highly relevant, the HMM however promotes also sentence 4 to be selected. Sentence 8 was selected because of its high ESA score, although it is irrelevant. Again, the context sentences indicate that also sentence 8 is rather irrelevant, which is reflected in the HMM result. The model thus corrects both the classification errors of the example without adding new ones. Sentence 1 for instance is still not selected, although it is followed by 2 highly relevant sentences.

4.6 Evaluation Results

The results of the DUPLICATE DOCUMENT REMOVER component can be found in Figure 4.1. As the component is exact, there is no need for a performance evaluation.

	Symbol	Similarity		Emission probabilities			Hidden State
k	O_k	*PMI*	*ESA*	σ	$b_1(k)$	$b_2(k)$	q_k
1	v_1 *"filtered"*	0.000	0.000	0.000	1.000	0.000	s_1 *"irrelevant"*
2	v_2 *"selected"*	0.630	0.057	1.000	1.000	0.000	s_2 *"relevant"*
3	v_2 *"selected"*	0.572	0.096	1.000	1.000	0.000	s_2 *"relevant"*
4	v_1 *"filtered"*	0.068	0.017	0.380	0.620	0.380	s_2 *"relevant"*
5	v_2 *"selected"*	0.471	0.053	1.000	1.000	0.000	s_2 *"relevant"*
6	v_1 *"filtered"*	0.236	0.001	0.384	0.616	0.384	s_1 *"irrelevant"*
7	v_1 *"filtered"*	0.067	0.002	0.157	0.843	0.157	s_1 *"irrelevant"*
8	v_2 *"selected"*	0.295	0.019	0.723	0.723	0.277	s_1 *"irrelevant"*
9	v_1 *"filtered"*	0.094	0.009	0.306	0.694	0.306	s_1 *"irrelevant"*

Table 4.6: Passage Extent Determination applied to the example data

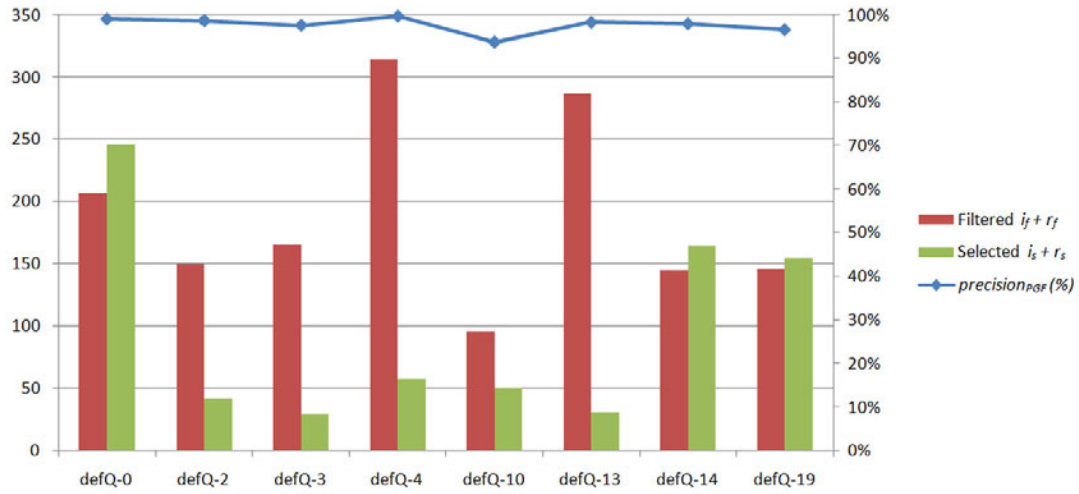
The PARAGRAPH FILTER component was optimized on the training query *"What is flash media?"* (`defQ-0`) and has been evaluated on 7 annotated queries of the evaluation corpus (cf. section 2.5). Figure 4.5 (a) shows the number of filtered and selected paragraphs for each query and the corresponding $precision_{PGF}$, according to the definition in section 4.3. The component performs very well—accomplishing precision values between 93.7% and 99.6%. The worst precision was observed for query `defQ-10`, which can be explained by the relatively small amount of paragraphs (147, compared to an average of 286 paragraphs per document), but high number of relevant passages (31, compared to an average of 21). Although the applied method is very simple, almost half of all paragraphs in each document is filtered out; thus reducing the amount of data by 66% in average.

The second diagram (b) shows the total number of relevant paragraphs $r_s + r_f$ together with those, that have been filtered out. Possibly relevant (center bar, r_{f1}) and definitely relevant (right bar, r_{f2}) are distinguished. With an exception in query `defQ-13`, the number of filtered-definitely relevant paragraphs r_{f2} is always lower than the number of filtered possibly relevants. In three queries, even not a single definitely relevant paragraph is filtered. In `defQ-10` however, 6 relevant paragraphs are filtered (4 possibly relevant/interesting, 2 definitely relevant). Although this is the maximum of the 8 testing queries, only 15% of all relevant paragraphs are lost and the remaining 33 paragraphs can be used for the summary.

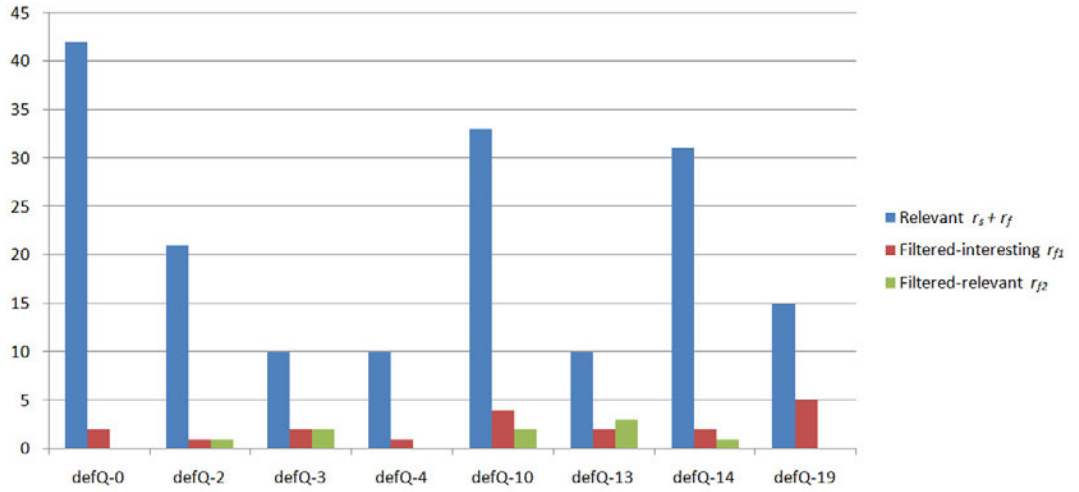
A similar evaluation has been done for the SENTENCE SELECTOR component. Figure 4.6 (a) shows the number of filtered and selected sentences for each query on the left axis. The component does not filter as many passages as the PARAGRAPH FILTER: only 21% of the sentences are filtered. If one considers however, that passages removed in PARAGRAPH FILTER are not regarded anymore, the amount of filtered sentences is totally fine. Quality measures as defined in section 4.4, are aligned to the right axis. The component achieves recall values of 83% to 100% both for $recall_{STS}$ and $recall_{2,STS}$, which indicates that most relevant sentences are selected. Precision ranges from 96% to 100%, indicating that more irrelevant

than relevant sentences are filtered out. In query `defQ-10`, there is however an outlier with precision 52%. The reason is, that only 21 sentences are filtered, of which 10 are relevant. This issue will be observed for improvement during passage extent determination.

Figure 4.6 (b) plots the total number of relevant sentences, together with the number of interesting and definitely relevant sentences that have been filtered out by the component. Note that the diagram was cut off to allow a detailed analysis of the range between 0 and 10. Query `defQ-0` contains 107, `defQ-10` 80 and `defQ-14` 63 relevant sentences. As seen in the precision value, `defQ-10` removes the most sentences: 7 possibly relevant and 3 definitely relevant. In `defQ-2` and `defQ-19` no relevant sentences are filtered, in `defQ-3` and `defQ-13` only 1 sentence is filtered.



(a) Number of filtered and selected paragraphs (left axis) and performance measures (right axis)

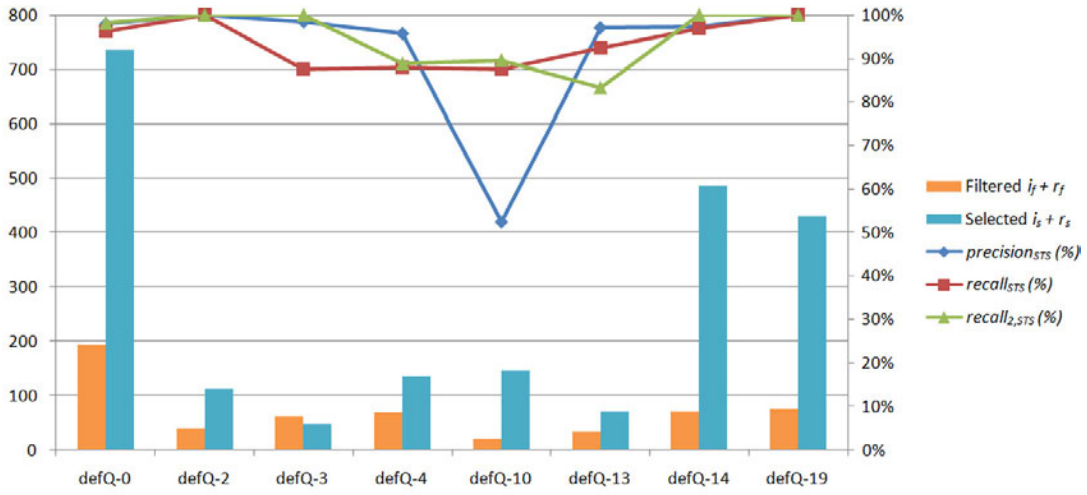


(b) Total number of relevant paragraphs and amount of filtered relevants

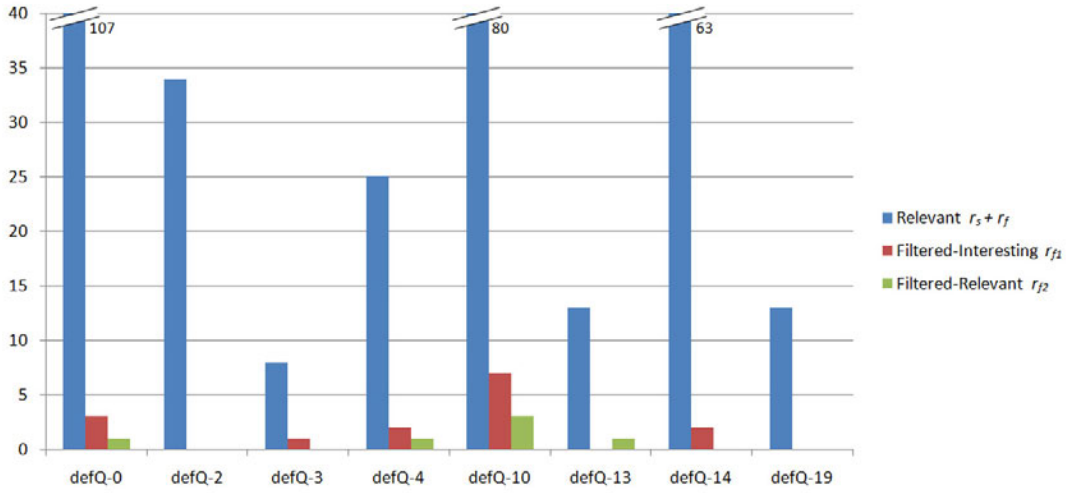
Figure 4.5: Evaluation of the Paragraph Filter component

The results of the SENTENCE SELECTOR are passed to the PASSAGE EXTENT DETERMINATION component. All the six scaling functions of section 4.5.2 are evaluated. In figure 4.7 (a) the change of the recall score $\Delta recall = recall_{PED} - recall_{STS}$ is shown.

A first observation is the large decrease in recall of the two spline approximations. Especially for query `defQ-3`, the recall drops from 87% to 12%. The amount is as high, since only 8 relevant passages exist for this query, of which only 1 is filtered by SENTENCE SELECTOR, but 7 by PASSAGE EXTENT DETERMINATION. With linear spline approximation, recall is reduced by 17% in average; with quadratic splines by even 30%. Both the scaling functions will therefore not be considered in the following. For the other scaling functions, the recall value remains stable after executing the PASSAGE EXTENT DETERMINATION component, for the quadratic polynomials and the sigmoid function, it is increased.



(a) Number of filtered and selected sentences (left axis) and performance measures (right axis)

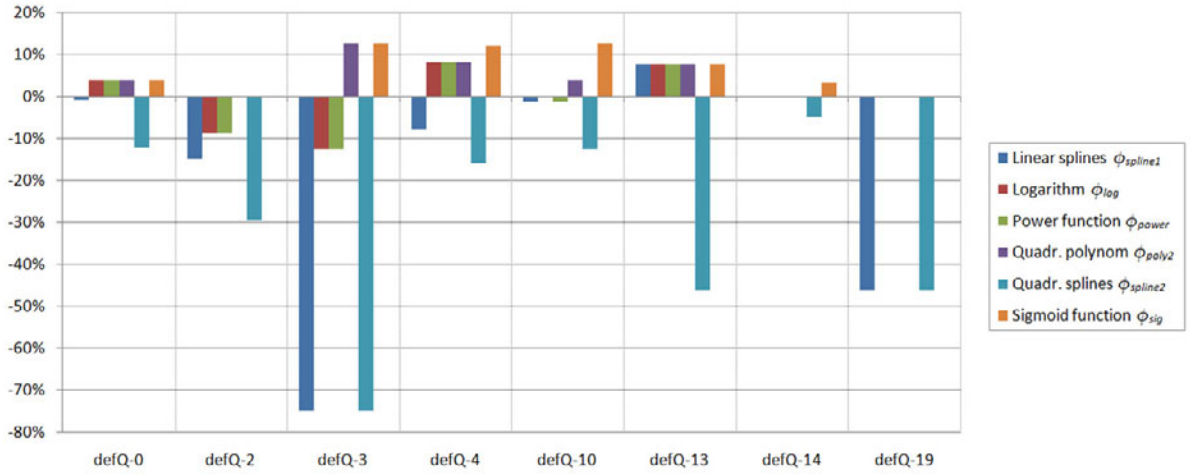


(b) Total number of relevant sentences and amount of filtered relevants

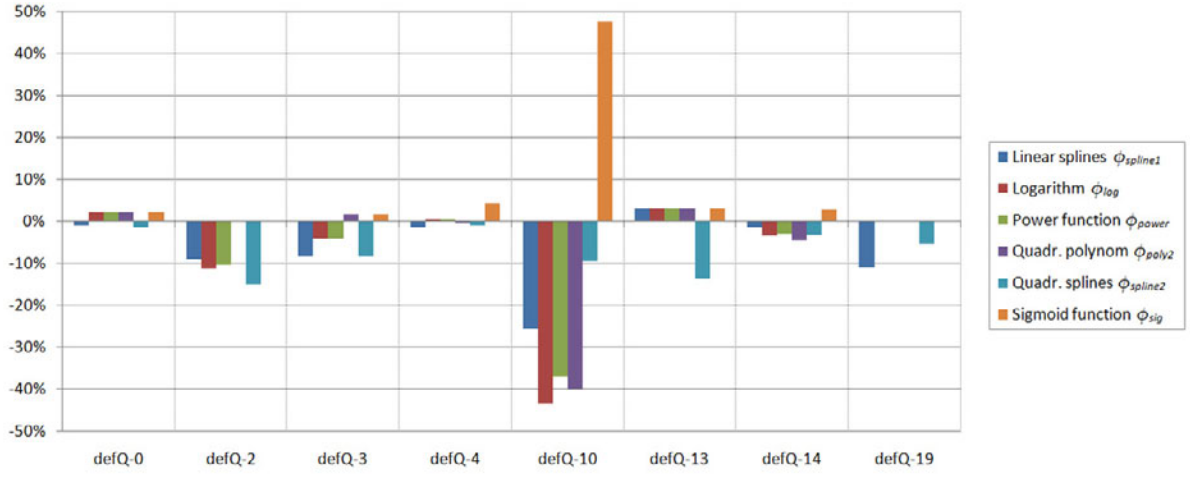
Figure 4.6: Evaluation of the Sentence Selector component

Figure 4.7 (b) shows the corresponding change $\Delta precision = precision_{PED} - precision_{STS}$ in precision score between SENTENCE SELECTOR and PASSAGE EXTENT DETERMINATION. Notable is again `defQ-10` with its high amplitude from −40% to +50% precision. The best result is encountered by using the sigmoid scaling function—precision then increases from 52% to 100% (All of the 10 filtered sentences are now selected). Over all tested queries, the sigmoid function leads to an average increase of 7% precision. As this is the best scaling function both for $\Delta precision$ and $\Delta recall$, the sigmoid function ϕ_{sig} is applied to both the similarity scores during PASSAGE EXTENT DETERMINATION.

Unfortunately, all the other scaling functions do not improve the result of the SENTENCE SELECTOR and are therefore omitted. Possible reasons for this behavior can be found in the



(a) Comparison of $\Delta recall = recall_{PED} - recall_{STS}$ of different scaling functions



(b) Comparison of $\Delta precision = precision_{PED} - precision_{STS}$ of different scaling functions

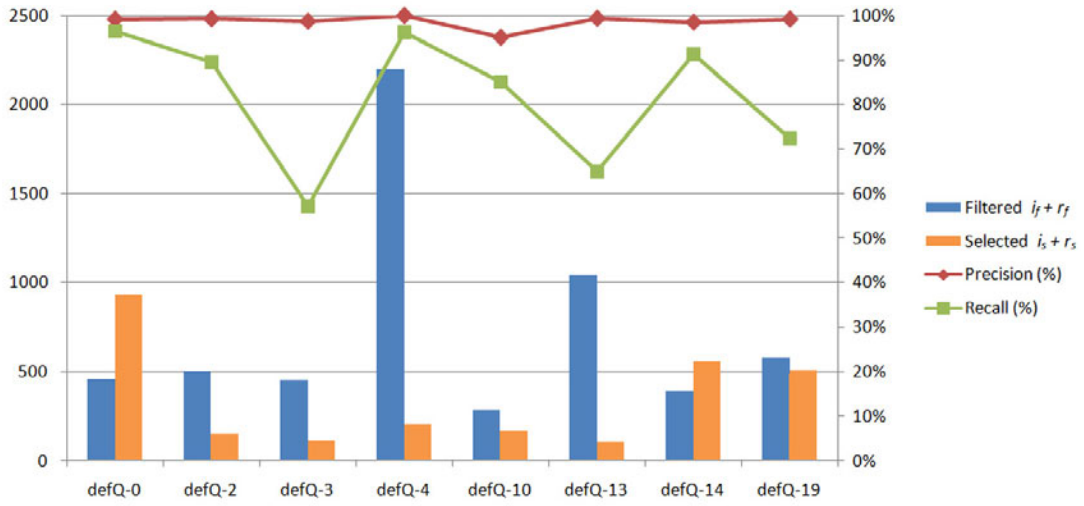
Figure 4.7: Evaluation of the Passage Extent Determination component

relatively small number of relevant sentences for each query. This allows the HMM to build long sequences of irrelevant states and thus filtering previously selected sentences that are surrounded by lot of irrelevant ones. The sigmoid function is special in this context, since similarity scores of (nearly) 0 do not lead to an emission probability of $\phi(0) = 0$. This allows to select sentences, whose similarity score could not be determined well and is therefore (nearly) 0. Also, sentences with a high similarity score are assigned with a high emission probability. The HMM considers a sentence then more likely to be relevant than in most other scaling functions.

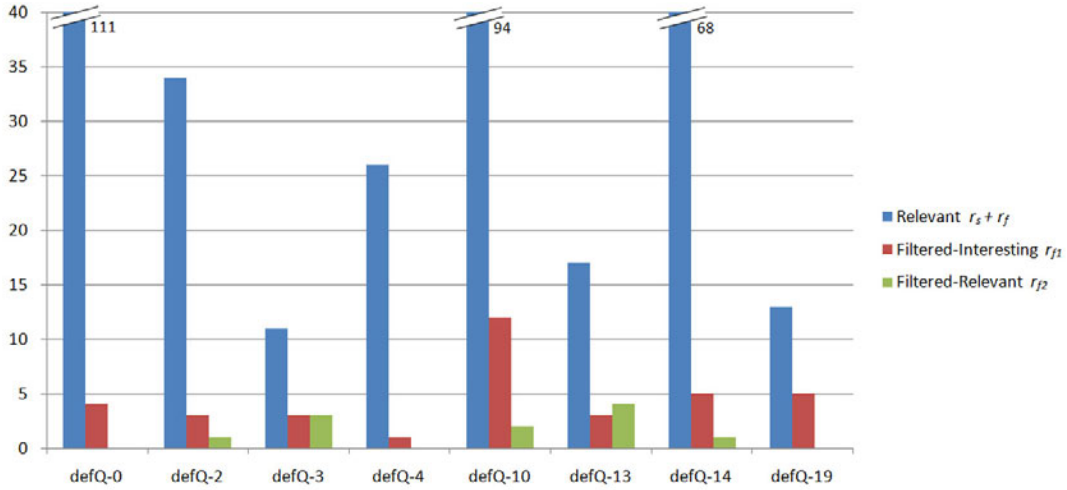
As the idea of the Hidden Markov Model based PASSAGE EXTENT DETERMINATION still makes sense due to the arguments of section 4.5, further improvements of the algorithm have been thought of. Retraining the model with dedicated data from the used data sources would be an evident suggestion. As the evaluation corpus is however very small and specific to the computer science domain, a larger training set would be useful to get stable results. Besides that, both the similarity scores and the scaling function can be varied. Currently, the sentence's AltaVista-PMI and Wikipedia-ESA score is used as input for the scaling function, allowing to do a further evaluation by including the cosine or the WordNet measures. Another idea would be, to choose a different threshold θ for the scaling function—a lower threshold would e.g. consider a medium similar sentence rather relevant than irrelevant, which hopefully leads to a better performance of the component.

Figure 4.8 shows the final result of the whole passage extraction task with the above configuration. The component's precision varies from 95% to 99% with an average of 98%, which is very good. The recall scores range from 57% to 96% with an average of 81%. Although these results could be further improved, they still lead to a high quality result of the passage extraction task that manages to remove an average of 66% of the retrieved sentences, of which only 1% is relevant. In absolute numbers, an average of 739 sentences is filtered for each query; only 5 of them are relevant. The maximum amount of relevant sentences (14) is filtered in query `defQ-10`. As this query however contains the second most relevant sentences (94), there should be enough relevant sentences left.

The large decreases of the recall score for query `defQ-3`, `defQ-13` and `defQ-19` are due to the small amount of relevant sentences: all of the 3 queries contain less than 20 relevant sentences—removing 6 of them (in average) obviously leads to a bad performance. To cope with that, a larger document collection could be used, which (hopefully) comes with a larger amount of relevant passages. Another possible problem occurs if a relevant paragraph is filtered, since there is the possibility that the paragraph consists of a significant amount of relevant sentences that would have been selected by the SENTENCE SELECTOR. If calculation time is not longer an issue, a combined system of PARAGRAPH FILTER and SENTENCE SELECTOR could be helpful to get more stable results.



(a) Number of filtered and selected sentences (left axis) and performance measures (right axis)



(b) Number of relevant sentences and fraction of definitely and possibly filtered relevants.

Query	i_f	r_{f1}	r_{f2}	i_s	r_{s1}	r_{s2}	*Precision*	*Recall*
`defQ-0`	454	4	0	823	52	55	0.991	0.963
`defQ-2`	497	3	1	118	32	2	0.992	0.894
`defQ-3`	448	3	3	103	7	1	0.986	0.571
`defQ-4`	2196	1	0	180	16	9	0.999	0.961
`defQ-10`	272	12	2	87	51	29	0.951	0.851
`defQ-13`	1034	3	4	91	7	6	0.993	0.650
`defQ-14`	387	5	1	496	43	20	0.984	0.913
`defQ-19`	578	5	0	495	6	7	0.991	0.722

(c) Data table

Figure 4.8: Final evaluation of the Passage Extraction task

Chapter 5

Topic Clustering

Having identified relevant and irrelevant parts of the retrieved documents, each answer should now be assigned to a topic group by calculating a clustering on the document set. Three different approaches will be tried and afterwards combined. The results are evaluated by computing the purity score of a clustering.

5.1 Background

If sentences from different topics are combined in a summary, it is obviously hard for the reader to find out, which fact belongs to which topic. Figure 5.1 shows three retrieved documents for the example query *"What is flash media?"* (`defQ-0`). The first document is about Adobe Flash, the second about flash memory (here: Compact Flash cards) and the third document is about the photographic flash. If topics are omitted, a resulting summary might look as follows:

> Compact Flash is widely used today as a convenient and affordable storage medium for digital cameras, handheld PDAs, MP3 players, and other personal electronic devices. In photography, a flash is a device that produces an instantaneous flash of light (typically around 1/1000 of a second) at a Color temperature of about 5500K to help illuminate a scene. Our Compact Flash (128 MB and larger) boasts the latest Ultra Performance 30X technology, making it 2.5 to 3.5 times faster than traditional Compact Flash or the products from our competitors!

Although it is easy to identify that the first and third sentence is about Compact Flash and the second is about photography, this summary looks odd and is rather confusing the reader. It would be better to order the sentences with respect to their topics, maybe divided into separate paragraphs. Displaying a list of topics would also be a helpful tool for the user to further define, which documents are of interest. The summary should then be built using only the selected topics.

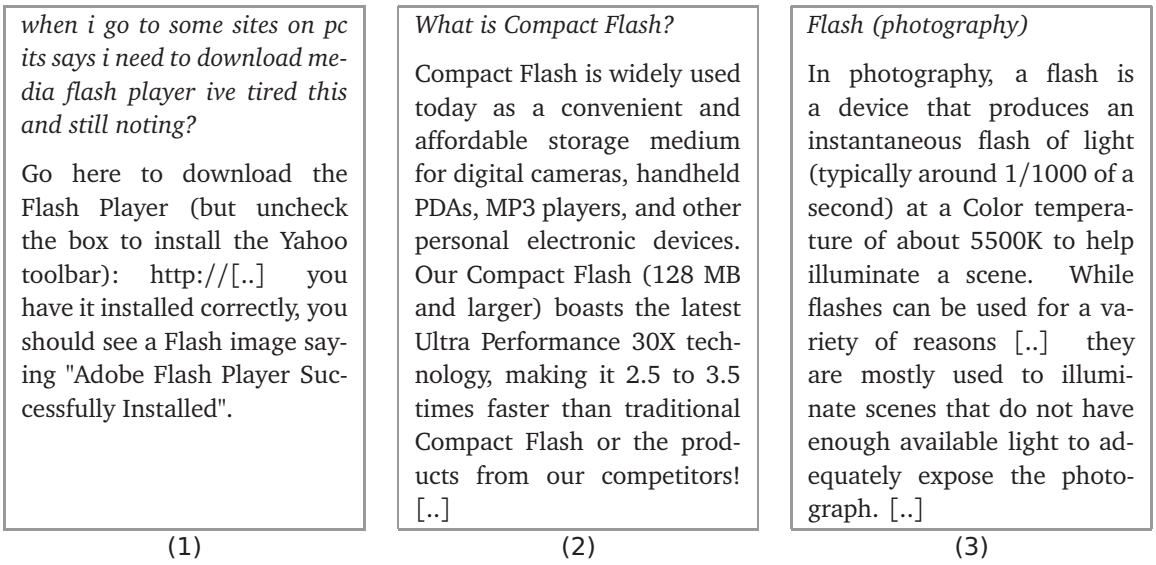


Figure 5.1: Three answer documents for the example query “What is flash media?”

Since neither the number of topics nor the topics themselves are known in advance, an unsupervised learning technique is applied to generate a list of topics and the corresponding clustering of the documents.

Definition 16 (Clustering) Let $D = \{d_1, d_2, \ldots, d_n\}$ be the set of retrieved documents and $C = \{c_1, c_2, \ldots, c_m\}$ a set of clusters with $m \leq n$. A *clustering* is a function $\varphi: D \longrightarrow C$, that assigns a cluster $c = \varphi(d)$ to every document d.

The *size* of a cluster $|c| = |\{d \mid \varphi(d) = c\}|$ determines the number of documents in cluster c. Obviously $\sum_{c \in C} |c| = n$ holds. A cluster c is called an *empty cluster* if $|c| = 0$, a *singleton* if $|c| = 1$ and a *supercluster* if $|c| = n$.

Clustering methods usually rely on the similarity of the documents to each other, in order to decide, whether they are about the same or about different topics. If the similarities of each document pair are required for the clustering algorithm, $\frac{1}{2}n(n-1)$ calculations have to be performed (omitting symmetric and identical pairs). To avoid long calculation times, a fast text similarity scoring should be chosen. In the following, *cosine similarity* (cf. definition 4) will be used, as it is the only text similarity measure in this thesis, that does not use the token pair-based maximization of definition 5 and thus provides a faster calculation for larger documents.

There are several different clustering methods—a good overview can be found in Jain et al. (1999) or Manning et al. (2008). A brief description of hierarchical agglomerative clustering, k-means clustering and Newman’s graph-based method is given in the upcoming sections, before the clustering results are evaluated.

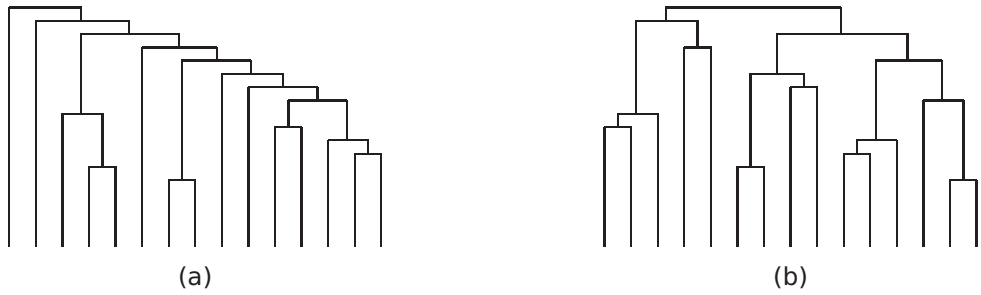


Figure 5.2: Two example dendrograms, (a) using single-link and (b) using complete-link

5.2 Hierarchical Agglomerative Clustering

An intuitive clustering method is to start with a singleton cluster for each document, merge those two clusters that are most similar and repeat until the desired number of clusters is reached or the maximum similarity score is below a certain threshold. The size of the clusters thus grows with the number of iterations, while the number of singleton clusters usually decreases. The result is called a *dendrogram*, which is a binary tree, connecting two nodes to a new one if they have been merged in the corresponding step. Example dendrograms can be found in figure 5.2.

This bottom-up approach is called *Hierarchical Agglomerative Clustering* in literature. The original ideas can be found in King (1967) and Sneath and Sokal (1973), while Jain et al. (1999) contains an excellent review. Hierarchical clustering methods mainly differ in the chosen document similarity method and the definition of similarity between two clusters. For the latter, common approaches are single-link, average-link and complete-link. Comparing each document of the first cluster with each document of the second cluster, a *single-link* approach uses the score of the most similar pair, while *complete-link* considers the least similar pair. *Average-link* however sums up all similarity scores and divides by the number of pairs (resulting in the mean similarity score).

While using the single-link technique, a chaining effect has been observed in Nagy (1968), which means that in (almost) every step, a large cluster is merged with a singleton cluster. The result of this effect usually contains a few very large clusters (or a supercluster) and many very small clusters (maybe only singletons). The dendrogram in figure 5.2 (a) shows the chaining effect. Complete-link clusters are usually more compact and often lead to better results in practice, according to Baeza-Yates (1992). Average-link approaches are a mixture between single-link and complete-link and are sometimes the best choice in practice, as of Manning et al. (2008), p. 395. All three approaches will however be evaluated on the training dataset.

An advantage of hierarchical agglomerative clustering is that the number of desired clusters does not have to be specified in advance. In fact the similarity score of two nodes can be

used as a cut-off threshold for the algorithm, allowing to cluster document collections both with a large and a small amount of clusters.

Besides agglomerative clustering (or bottom-up clustering), a divisive clustering (or top-down clustering) can be calculated. This technique starts with a supercluster of all documents and splits the cluster that contains the best distinguishable contents. As hierarchical divisive clustering is only seldom used in practice, the evaluation will focus on the agglomerative approach.

5.3 Newman Clustering

Another clustering approach addresses finding *community structures* in a graph representation of the documents. A community, according to Newman (2006b), is a group of vertices with a high density of intra-group connections and a low density of inter-group connections. Newman and Girvan (2004) present an algorithm that identifies such communities by iteratively removing edges with a high *betweenness* score (rather than the lowest similarity in divisive hierarchical clustering). Vertices between two communities have a higher betweenness value than vertices inside a community.

Due to high computational complexity, Newman (2004a) improves this algorithm by calculating the *modularity* score of a clustering. Modularity indicates the difference between the number of inter-community edges and the expected number of such edges. While the first follows directly from the current clustering, the latter however depends on choosing an appropriate *null model*, i.e. the expected graph structure. A simple null model is the Bernoulli model, which expects an Erdős-Rényi random graph with equal probabilities for each pair of nodes to be connected by an edge—cf. Erdős and Rényi (1959) for details.

Newman (2006a) however proposes to use a null model that better represents real-world networks. In fact, the expected vertex degree should equal the actual vertex degree, which promotes the probability of an edge between two nodes with high outgoing degrees. The expected graph structure is then scale-free, according to the definition in Barabási and Albert (1999). Newman's modularity score Q of a network therefore calculates to

$$Q = \frac{1}{2m} \sum_{i,j} (A_{ij} - P_{ij}) \delta(g_i, g_j),$$

where A_{ij} is the number edges between vertices i and j; m is the total number of edges; g_i is the corresponding cluster, assigned to vertex i; $\delta(g_i, g_j)$ is 1 if g_i and g_j are equal and 0 otherwise. P_{ij} expresses the expected number of edges between two communities and is set to $P_{ij} = \frac{1}{2m} k_i k_j$, with k_i denoting the degree of vertex i (the scale-free model). This definition follows Newman (2004a) and Newman (2006a).

For clustering the retrieved answers, a graph representation needs to be created first. Each document becomes a node, while the similarity of a document pair forms a weighted edge

between both the document's nodes. Additionally, a threshold can be defined to omit edges with low similarity scores. Changes in the algorithm that allow working with weighted edges, can be found in Newman (2004b).

Exhaustive enumeration of all possible clusterings to find the maximum Q and thus the best community-clustering is usually intractable due to the high computational effort. Possible heuristic optimizations imply simulated annealing or a method based on the spectral values of the graph. For this thesis, the LinLogLayout[30] implementation of Newman's modularity score is used, which solves the optimization problem with a greedy approach.

5.4 k-Means Clustering

An early statistical clustering approach iteratively improves a given partitioning in each step. The original ideas for such a method can be found in Steinhaus (1956). If applied to data clustering, the method is known as *k-Means Clustering*. Since no cluster hierarchy is built, it is also called a flat clustering technique.

k-means clustering starts with k randomly chosen cluster centers that may lie at arbitrary positions in the document space. One possibility is to simply use k of the input documents as initial positions. For each document $d \in D$, the similarity to all k cluster centers is calculated and the most similar one is chosen as new $\varphi(d)$. These newly generated clusters update their cluster center by calculating the mean document. The process repeats with computing the most similar (i.e. nearest) cluster center for every document until there is no more change in the resulting clustering or a maximum number of iterations is reached. For the implementation here, a cluster center is defined as a document vector that consists of all words in the cluster's documents.

The underlying approach is called *Expectation Maximization*, as previously introduced in Dempster et al. (1977). Expectation maximization consists of two steps, which for k-means clustering are: assigning each document to the most similar cluster center (E-Step or expectation step) and calculating the new cluster centers (M-Step or maximization step).

Depending on the choice of the starting point, the results can differ very much. This allows improving the results through multiple runs and needs to be considered during evaluation. One problem is the fixed number of clusters that needs to be known in advance. For the Question Answering task this is not clear from the beginning: a document collection for an ambiguous query can consist of many different topics, while an unambiguous query might lead to only documents about the same topic that would not need any clustering at all. In hierarchical clustering, this problem was overcome by setting a cut-off threshold.

[30] LinLogLayout – http://code.google.com/p/linloglayout

5.5 Cluster Merging

To benefit from both the advantages of two different clustering techniques, a method to combine the results of two arbitrary clustering algorithms has been developed for this thesis. Although various definitions of such a *cluster merging* algorithm could be tried, only one technique will be considered here:

Definition 17 (Cluster Merging) Let $D = \{d_1, d_2, \ldots, d_n\}$ be the set of retrieved documents and $C_1 = \{c_{1,1}, c_{1,2}, \ldots c_{1,m}\}$, $C_2 = \{c_{2,1}, c_{2,2}, \ldots c_{2,k}\}$ be cluster sets. A *merged clustering* of the input clusterings $\varphi_1 : D \longrightarrow C_1$ and $\varphi_2 : D \longrightarrow C_2$ is a function

$$\varphi : D \longrightarrow C, \text{ with } d \mapsto (c_1, c_2) \text{ and } C = C_1 \times C_2.$$

Even though the amount of new clusters is $|C| = |C_1| \cdot |C_2| = mk$, many clusters are usually empty and can be omitted. Table 5.1 shows two clusterings of 7 example documents and the corresponding merged clustering. Clustering φ_1 contains three clusters with sizes $|1| = 3, |2| = 2, |3| = 2$, while φ_2 also has three clusters with sizes $|a| = 4, |b| = 2, |c| = 1$. The merged clustering thus contains 9 clusters, of which only 5 are not empty: $|1a| = 2, |1b| = 1, |2a| = 2, |3a| = 1, |3c| = 1$.

D	d_1	d_2	d_3	d_4	d_5	d_6	d_7
φ_1	1	1	1	2	2	3	3
φ_2	a	a	b	a	a	b	c
φ	$1a$	$1a$	$1b$	$2a$	$2a$	$3b$	$3c$

Table 5.1: Merged clustering for two example input clusterings

5.6 Evaluation Results

Several evaluation tests of the clustering algorithms have been done. After defining the utilized quality measures, some of the main results are presented in the following sections.

5.6.1 Baseline and Quality Measures

To evaluate the performance of the different clustering approaches, the manually annotated document groups of the evaluation dataset (see section 2.5) can be used. As the generated clusters do not have a unique id, it is however not possible to directly map the annotated (expected) clusters to those, found by the algorithms. The mapping needs to be estimated by determining the largest overlap of expected and actual clustering. Strehl (2002) and Manning et al. (2008) define the purity score of a clustering that will be used in the following to quantify the evaluation results.

Definition 18 (Clustering Purity) Let $C_e = \{c_{e,1}, c_{e,2}, \ldots, c_{e,m}\}$ be the set of expected clusters and $C_a = \{c_{a,1}, c_{a,2}, \ldots, c_{a,k}\}$ the set of actual clusters. The corresponding clusterings are $\varphi_e : D \longrightarrow C_e$, representing the manual annotations, and $\varphi_a : D \longrightarrow C_a$, which is the result of the clustering algorithm ($D = \{d_1, d_2, \ldots, d_n\}$ is again the document collection). The *purity* of clustering φ_a with respect to φ_e is defined as:

$$purity(\varphi_a, \varphi_e) = \frac{1}{n} \sum_{i=1}^{k} \max_{j=1,\ldots,m} |cont(c_{a,i}) \cap cont(c_{e,j})|$$

with $cont(c)$ denoting the set of documents in cluster c. Analogously, the *inverse purity* can be computed:

$$invpurity(\varphi_a, \varphi_e) = \frac{1}{n} \sum_{i=1}^{m} \max_{j=1,\ldots,k} |cont(c_{a,j}) \cap cont(c_{e,i})|$$

Consider the document collection $D = \{d_1, d_2, \ldots, d_7\}$ with expected topics $C_e = \{\square, \triangle, \diamondsuit\}$ and found clusters $C_a = \{a, b\}$. The following table shows the cluster assignment (left) and a matrix of cluster overlaps (right):

D	d_1	d_2	d_3	d_4	d_5	d_6	d_7
φ_e	$\square$	$\square$	$\triangle$	$\triangle$	$\triangle$	$\triangle$	$\diamondsuit$
φ_a	a	a	a	b	b	b	b

	$\square$	$\triangle$	$\diamondsuit$	max
a	2	1		2
b		3	1	3
max	2	3	1	

Each cell contains the amount of documents that are both assigned to the row's cluster $\varphi_a(d)$ and the column's cluster $\varphi_e(d)$, i.e. the overlap $|cont(c_a) \cap cont(c_e)|$ with $c_a \in C_a$ and $c_e \in C_e$. Given this matrix, the corresponding purity of φ_a and φ_e is the sum of the maximum values in each row, divided by the total number of documents—calculating to $5/7$. The inverse purity is $6/7$ and can be calculated the same way, using the maximum of each column.

While annotating the evaluation data, the goal was to obtain a good separation of totally different topics, rather than having an in-depth separation of minor details. Documents about *Adobe Flash* are for example in a different cluster than documents about *flash memory technologies*, while documents about the flash memory in *USB sticks* are assigned to the same cluster as documents about *JFFS*,[31] even though the topics differ in detail.

Besides that, there are several "miscellaneous" clusters in the evaluation dataset that contain rare topics with only a few (irrelevant) documents. For the query *"What is flash media?"* (`defQ-0`), there was one document about a flash in a dice game and one document about *flash mobs*[32] retrieved that have been assigned to the same cluster, although there are about

[31] JFFS: a journaling file system for flash memory

[32] Flash mob: a spontaneous gathering of a crowd in a public place for performing an unusual action, usually coordinated by internet or cell phone

different topics. The intention was to have only a few amount of clusters that represent the most important groups of documents and provide an appropriate summary to the user. The expected cluster count is usually between 1 and 5. Appendix A.2 shows the list of annotated training queries and the number of annotated clusters.

The goal of the TOPIC CLUSTERING component is not to obtain a perfect match with the annotated data (which would be a hard task), but to model the expected topic separation as good as possible, maybe by defining a few more clusters than necessary. Evaluation will therefore focus on purity, because it is the best representation of above considerations. Since purity is highest if each document is assigned to its own singleton cluster, there will however be a tradeoff between purity and the amount of clusters. As a baseline system for purity, the clustering is chosen according to the data source of the document.

5.6.2 Evaluation of Hierarchical Agglomerative Clustering

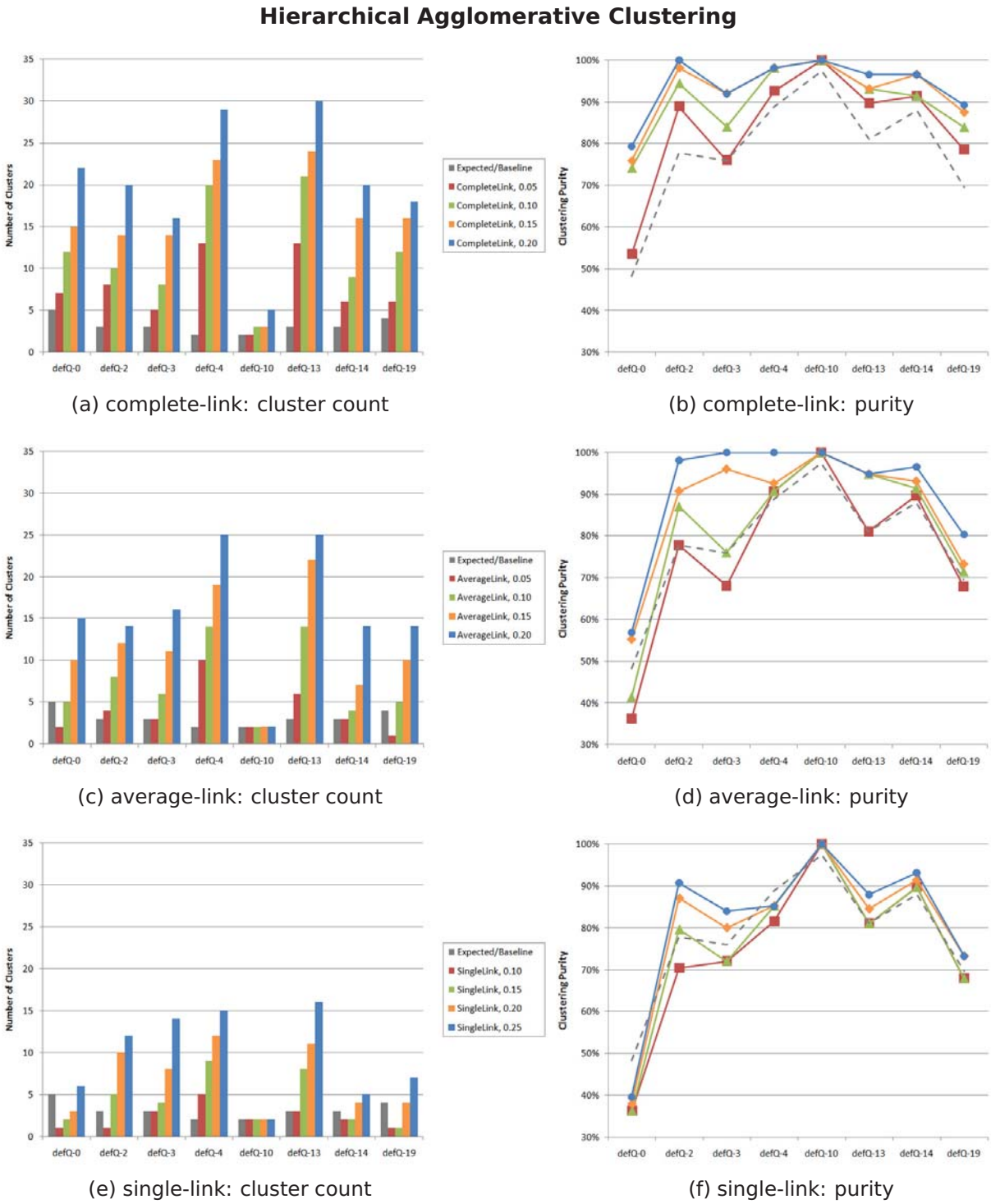
The first algorithm to be evaluated is hierarchical agglomerative clustering (HAC). Parameters for this algorithm are the cut-off threshold and the definition of inter-cluster similarity (i.e. complete-link, average-link and single-link). Figure 5.3 shows the performance with all of the three similarity definitions and several corresponding cut-off thresholds. The left side diagrams show the amount of clusters that have been generated by the algorithm, together with the expected number that is based on the evaluation corpus. On the right side, the corresponding purity scores can be found. The dashed line there indicates the results of the baseline system.

It is not surprising that purity increases with a higher threshold (and thus a higher number of clusters), since purity is maximal when having a singleton cluster for each document. A reasonable threshold should therefore be found that is both near the expected number of clusters and provides a high purity.

The complete-link approach generates many clusters. Even with a cut-off threshold of only 0.05, the number of actual clusters is higher than the expected for every document. For `defQ-4` and `defQ-13`, even 13 clusters are created. The purity scores however outperform the baseline for every query and threshold. As a reasonable value, 0.1 could be chosen, since it leads to an average boost in purity of 6%, compared to a cut-off threshold of 0.05. Further increasing the threshold, results in an average boost of only 2%, while 6 more clusters would be generated for each query in average.

In the average-link approach, the number of generated clusters is lower than in complete-link, while the purity score is also lower. The baseline is only outperformed for a cut-off threshold greater or equal 0.15. Best choice for average-link would thus be a threshold of 0.15, resulting in an average of 11 clusters per query.

The amount of clusters is further decreased in a single-link approach. For a threshold of 0.1, the total number of clusters is even lower than the expected count. Unfortunately,



(a) complete-link: cluster count

(b) complete-link: purity

(c) average-link: cluster count

(d) average-link: purity

(e) single-link: cluster count

(f) single-link: purity

Figure 5.3: Evaluation results for different parameters in Hierarchical Agglomerative Clustering

single-link HAC does not fully outperform the baseline, which can be explained by the chaining effect, described in section 5.2. Especially the queries `defQ-0` and `defQ-4` could not outperform the baseline at any of the evaluated thresholds. For the other queries this is at least possible for a threshold ≥ 0.2. The threshold 0.2 will thus be considered in the following as the best choice for single-link HAC.

5.6.3 Evaluation of Newman Clustering

Newman's clustering method has only one parameter to evaluate: the similarity threshold for inserting edges into the graph. If the similarity score of two documents is below this threshold, no edge will be inserted. Since a higher threshold decreases the connectivity of the graph, the number of generated clusters usually increases with higher thresholds. Figure 5.4 shows the evaluation results for four different thresholds compared to the expected number of clusters resp. the baseline system.

Almost all thresholds outperform the baseline, there are however two exceptions for 0.05 and 0.15. A reasonable trade-off between cluster count and purity seems the choice of 0.1, leading to an average cluster count of 4, while the purity score is in average 3% higher than the baseline system.

An interesting observation of Newman clustering, is the decreased purity for threshold 0.15 and query `defQ-3`. While the purity score was constantly increasing with higher thresholds in hierarchical agglomerative clustering, this is no longer true for Newman clustering. A more detailed evaluation of the similarity thresholds between 0.1 and 0.15 could thus lead to interesting results and can be analyzed in the future.

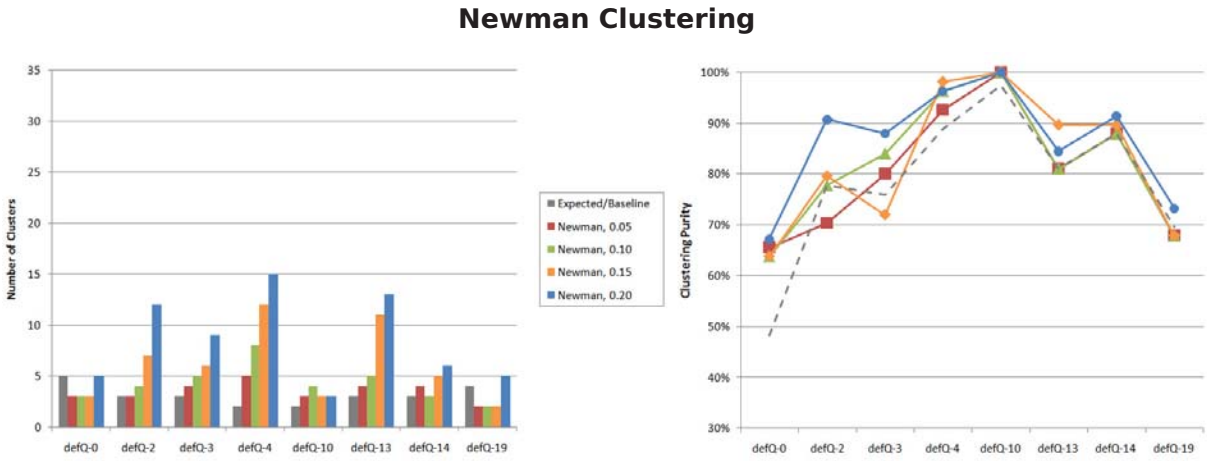
5.6.4 Evaluation of k-Means Clustering

The results of the k-means clustering algorithm differ in the number of created clusters by choosing parameter k. As the number of expected clusters varies from 3 to 5, these values (and additionally $k = 6$) will be used to evaluate the algorithm.

Figure 5.5 shows the purity scores for these four settings. Multiple runs have been calculated, each with a different initial clustering. The diagrams show four randomly chosen representatives to visualize the range of clustering purity scores.

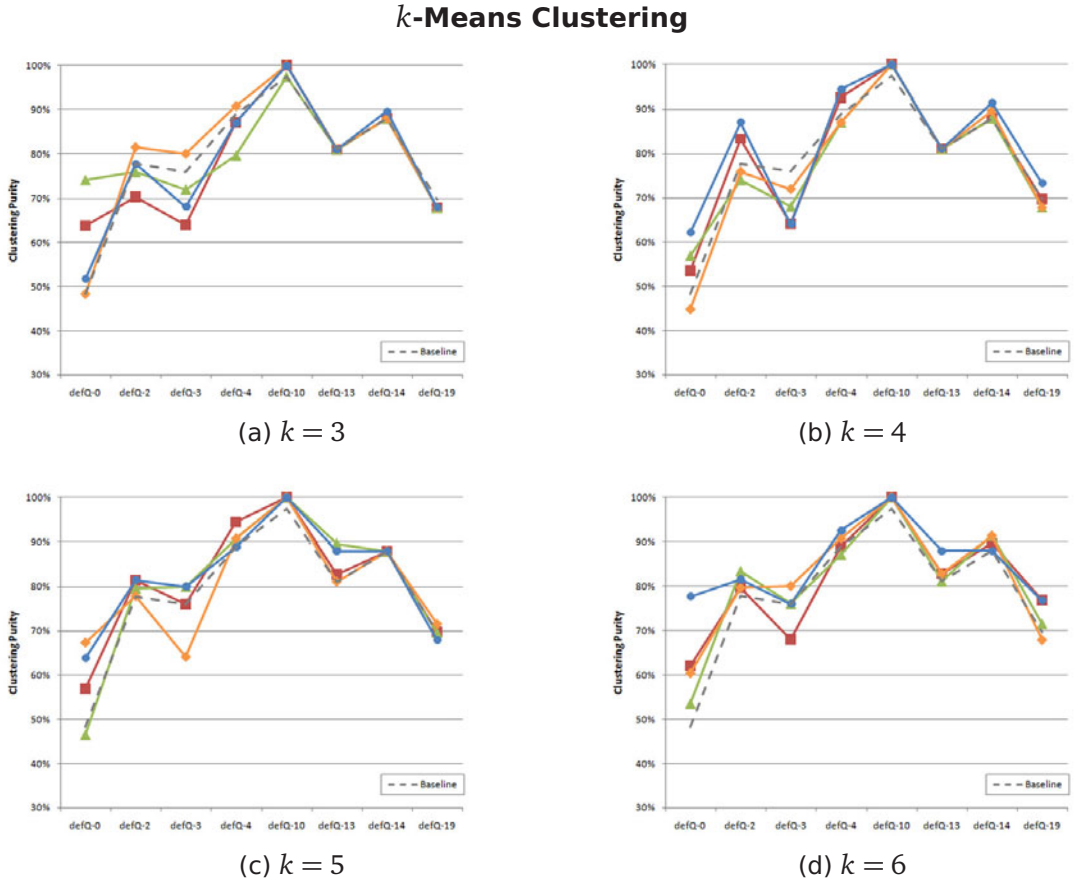
The results are strongly aligned with the baseline system. For $k = 3$ and $k = 4$, the purity score tends to be below the baseline, for $k = 5$ and $k = 6$ above. $k = 5$ therefore seems to be a good trade-off between the number of clusters and the resulting purity.

Notable is the overall bad performance of k-means clustering on `defQ-3`, which is the query *"What is a 'mashup'?"*. This trend can also be seen in the other clustering approaches, with the exception of complete-link HAC, which fully outperforms the baseline. Although, the sentences in the evaluation corpus could be clearly divided into mashup in the sense of a



(a) cluster count (b) clustering purity

Figure 5.4: Evaluation results for different parameters in Newman Clustering



(a) $k = 3$ (b) $k = 4$

(c) $k = 5$ (d) $k = 6$

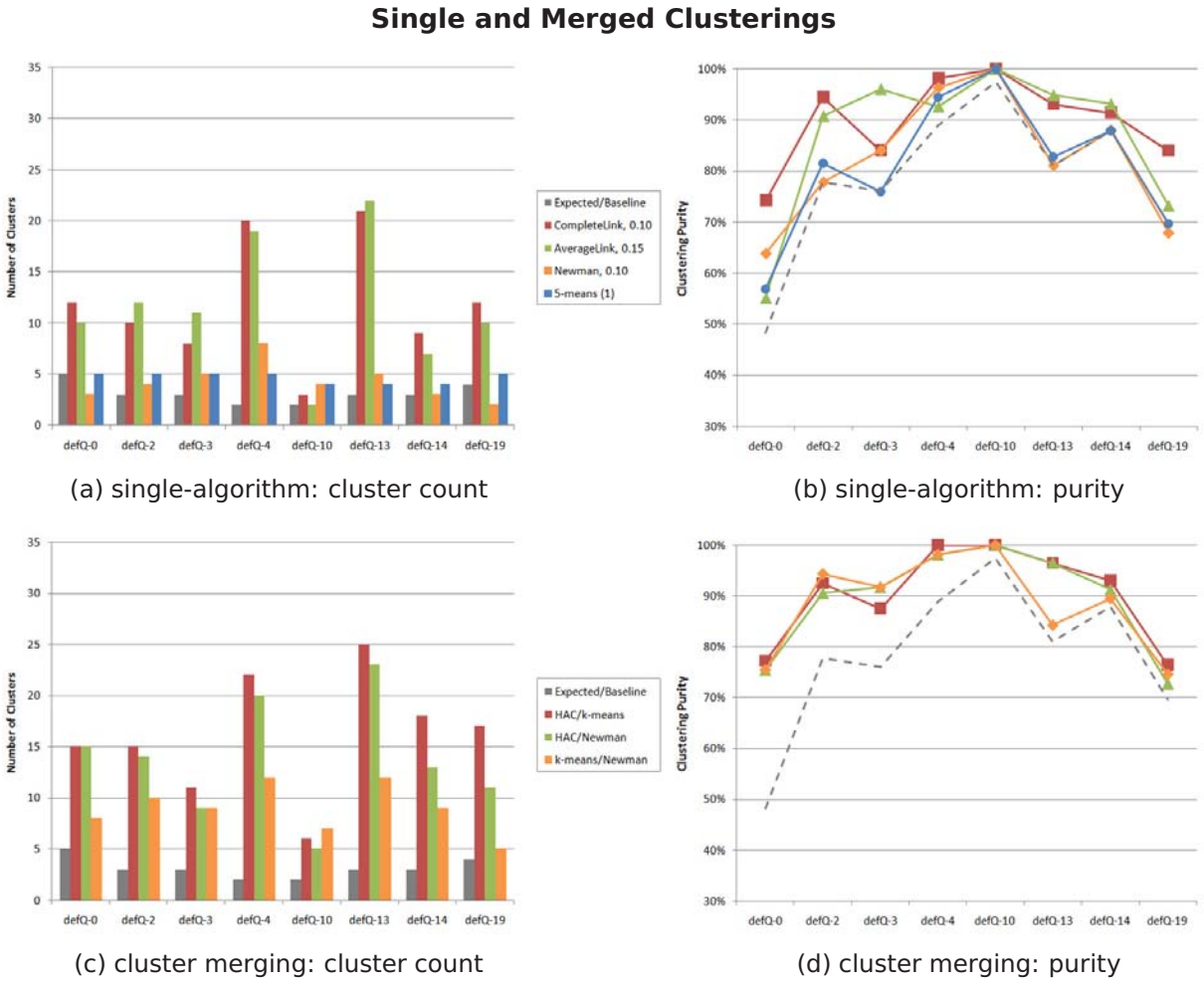
Figure 5.5: Evaluation results for different parameters in k-Means Clustering

musical remix and in the sense of a Web 2.0 application that combines content from several sources (plus a third cluster for miscellaneous topics), the inter-sentence similarities seem not to work well with this kind of data.

5.6.5 Evaluation of Cluster Merging

To overcome problems of individual algorithms like the one described above, a combination of two of the clustering approaches could be helpful to benefit from both the advantages of the techniques. The cluster merging approach as defined in section 5.5 will be used for combining the results.

Figure 5.6 shows a comparison of the four best individual clustering algorithms and three results of the merged clusterings. Following the arguments of the previous sections, as best single clustering methods have been chosen (diagrams a–b): complete-link HAC with cut-



(a) single-algorithm: cluster count

(b) single-algorithm: purity

(c) cluster merging: cluster count

(d) cluster merging: purity

Figure 5.6: Comparison of single-algorithm and merged clusterings

off threshold 0.1 (red squares), average-link HAC with threshold 0.15 (green triangles), Newman with threshold 0.1 (orange diamonds) and k-means with $k = 5$ (blue circles). As combinations (diagrams c–d) have been tried: complete-link HAC with threshold 0.15 merged with Newman 0.1 (red squares), same HAC with k-means at $k = 5$ (green diamonds), same k-means with Newman 0.1 (orange diamonds).

All the single-algorithm clusterings exceed the baseline system, but only the two HAC approaches lead to a noticeable quality boost. The purity scores of the merged clusterings are in average 4% higher than those of the single-algorithms and are in average 11% higher than the baseline. The merged clusterings generally produce a higher number of clusters but as stated before in section 5.5, the number of clusters does not grow quadratically and is only 4.4 clusters per query.

If the number of clusters exceeds 15, the system gets impractical, because the user has to enable/disable lots of clusters to get an appropriate summary. Unfortunately HAC generates up to 25 clusters and will not be considered in following. The remaining algorithms are maximized, resulting in the merged clustering of Newman 0.1 and k-means with $k = 5$. This configuration is used for the TOPIC CLUSTERING component.

Chapter 6

Answer Summarization and Presentation

The final task of this thesis is generating a summary of the relevant information found in the previous components. After removing invalid sentences, a ranking is computed to find out what are the most important sentences to be included in the summary. The chapter concludes with the presentation of the system's graphical user interface and an evaluation of the summary results.

6.1 Invalid and Duplicate Sentence Removal

In section 4.2, duplicate documents have been removed by the DUPLICATE DOCUMENT REMOVER component to avoid processing identical documents, which would not lead to new results. The remaining documents could however still contain duplicate sentences. This is e.g. the case in FAQ articles that have been derived from each other. Although the main text of the two articles is identical, adding only one additional information would cause the DUPLICATE DOCUMENT REMOVER to keep both documents. Duplicate sentences also emerge from Wikipedia's article templates that provide standardized headers (like *"External links"* or *"This section requires expansion"*).

If a sentence appears twice in a summary text, the result looks unnatural and contains obvious redundancy. Therefore, duplicates are removed by the DUPLICATE SENTENCE REMOVER component, which applies the same method used before in the DUPLICATE DOCUMENT REMOVER component: For each sentence an MD5 hash is calculated and saved. If a sentence with the same hash exists already, the sentence is removed.

Looking through the remaining text, there are often invalid or incomplete sentences. Many of them have been used as keywords, headlines or list items, like *"∗ Flash crowd"*, *"Slashdot effect"* or *"Types of green flashes"*. They may however also occur if the text contains orthographical or grammatical errors, e.g. *"yes its possible."* or *"computer into media centre?"*.

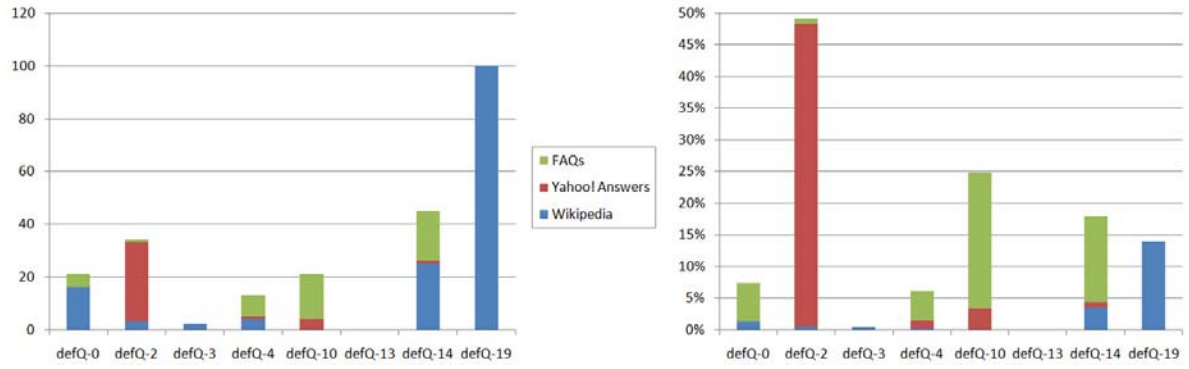


Figure 6.1: Number and percentage of sentences removed by the Duplicate Sentence Remover

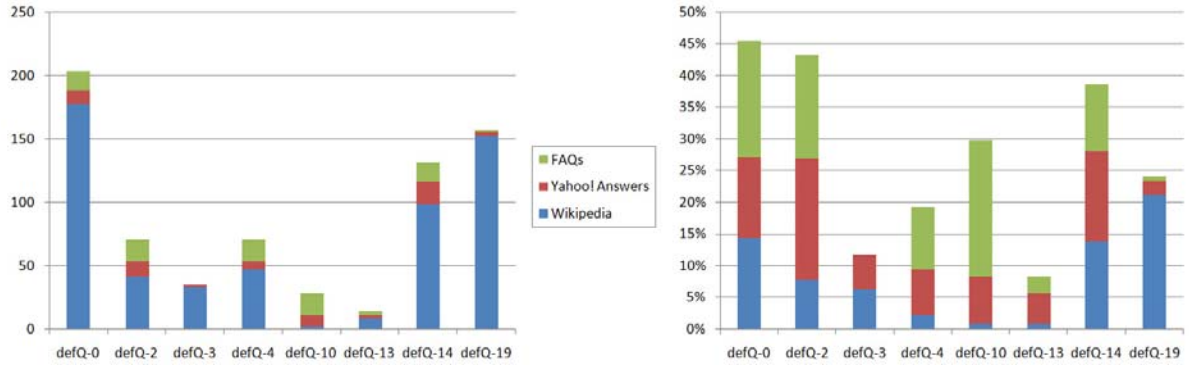


Figure 6.2: Number and percentage of sentences removed by the Invalid Sentence Remover

Invalid sentences are usually confusing the reader if they are found within a summary and should thus be removed. A possible method for identifying these sentences is linguistic parsing. The input sentence is valid if a feasible parse tree can be derived, or invalid otherwise. Parsing is however very time-consuming, which is the reason for choosing a simpler method here: Based on the part-of-speech annotations of the tokens, a sentence is considered valid if it consists of at least one noun and at least one verb. Part-of-speech annotations have been created during the preprocessing steps (cf. section 3.1) and can therefore be reused for the INVALID SENTENCE REMOVER component.

Figure 6.1 shows the amount of duplicate and figure 6.2 the amount of invalid sentences for some example queries of the training dataset. Most sentences are removed from Wikipedia articles, which is due to the high number of repeating headings and keywords. Besides that, the Wikipedia articles simply contain the most sentences (in average 890 compared to 121 in Yahoo! Answer documents and 89 in FAQs). There is no need to evaluate the performance of the duplicate sentence removal, since the method is exact. The removal of invalid sentences could be evaluated with precision and recall on a manually annotated corpus. A quantitative analysis for the training dataset has not yet been done. Manually inspection showed however that the result quality is appropriate for the summarization task.

6.2 Sentence Ranking

Until now, only the Boolean relevance of passages has been considered, without any particular order. Jansen et al. (2000) found out that most users of a search engine tend to only look at the first page of results before reformulating their query. The same may apply to summaries: The user might stop reading if the answer starts with irrelevant or less interesting sentences.

Figure 6.3 shows two example summaries for the training question *"What is flash media?"* (`defQ-0`) that combines the same six sentences in two different orders. Although all of the six sentences are somehow relevant for the query (interpreted as topic *Adobe Flash*), the summary in (a) does not explain what flash really is in the beginning. The reader finds out that flash is something from Macromedia, which can be installed and maybe comes already with the browser. Summary (b) in contrast, provides more important information in the beginning, viz. that flash is a multimedia plug-in for web pages, allowing users to create animations that can be seen in the web browser.

Thus, it is an important task to provide the most informative information at the beginning of the answer to convince the reader of its relevance and usefulness. *Sentence ranking* methods can be used to determine an ordered list of the relevant sentences with the most important ones at the top and the least important at the bottom.

A naïve approach simply orders the sentences according to their occurrence in the input document with the idea that the most informative sentences appear at the beginning of the retrieved documents. This observation holds if one considers Wikipedia articles, which usually provide the most important information in the beginning of the article. The approach will be referred to as *Position Rank* and acts as a baseline in the upcoming evaluation.

What is flash media?

Installing Flash from Macromedia's web site is quick and easy. Flash was known as FutureSplash until 1997, when Macromedia Inc. bought the company that developed it. While most computer systems and browsers come pre-packaged with Flash, your configuration may vary. Flash animations will look the same in all browsers. With Flash, users can draw their own animations or import other vector-based images. Macromedia Flash is a powerful multimedia plug-in that allows you to view rich media content on the internet.

(a)

What is flash media?

Macromedia Flash is a powerful multimedia plug-in that allows you to view rich media content on the internet. With Flash, users can draw their own animations or import other vector-based images. Flash animations will look the same in all browsers. While most computer systems and browsers come pre-packaged with Flash, your configuration may vary. Installing Flash from Macromedia's web site is quick and easy. Flash was known as FutureSplash until 1997, when Macromedia Inc. bought the company that developed it.

(b)

Figure 6.3: Two example summaries with different sentence rankings

Another simple method makes use of the similarity scores, which have been calculated during the passage extraction task (cf. chapter 4). The sentence with the highest similarity score (i.e. the most informative) is annotated with rank 1, followed by the sentence with the next smaller similarity score and so forth. The method will be called *Similarity Rank*. To obtain the ranking, the σ value of each sentence is used; σ has been introduced as emission probability during passage extent determination (cf. section 4.5) and represents a combination of a scaled PMI and ESA similarity score.

In the following, a third, more elaborated approach will be introduced that does not rely on the independent sentences only, but considers the whole document collection by taking into account the similarities between each pair of sentences.

6.2.1 Biased LexRank

Some common ranking techniques are based on graphs. The idea of most algorithms is that the most prominent nodes in the graph should receive the highest ranks. Such prominent nodes are often defined to be at central positions of the network (so called *central nodes* or *hubs*), which can be identified by calculating a *centrality* score for each node and rank the nodes according to this score in descending order.

The centrality score of a node can be calculated during a (Markov) *random-walk* through the graph. A random-walk starts at an arbitrary node and randomly follows one of the incident edges in each step. The basic idea of this technique is that important (i.e. central) nodes are often visited, as they can be reached from many other nodes. Good overviews on random-walk theory and centrality measures can be found in Henzinger et al. (1999) and Freeman (1978/79).

Possibly the most famous random-walk algorithm is *PageRank*, the basic ranking algorithm of the Google[33] search engine. Brin and Page (1998) implement a random-walk on the World Wide Web, based on the hyperlinks between pages. In each step, the algorithm chooses between following an incident edge or jumping to a random node. The latter is necessary to ensure termination of the algorithm.

Erkan and Radev (2004) present an extension of the PageRank algorithm that focuses especially on ranking sentences. The method is called *LexRank* and uses inter-sentence similarity scores as (weighted) graph edges instead of the (unweighted) hyperlinks in PageRank. A further improvement for generating focused summaries has been recently introduced in Otterbacher et al. (2009) and is known as *Biased LexRank*. The similarity score between the sentence and a certain context or topic is included in this approach and allows a ranking with respect to the given context. More precisely, the random jump is weighted according to the node's similarity to the context. For the question answering system, the query will be used as context, resulting in the following definition of the Biased LexRank algorithm:

[33] Google – http://www.google.com

Definition 19 (Biased LexRank) Let $G = (V, E, w)$ be the weighted graph of all sentences V that have been connected by an edge $(u, v) \in E$ between node u and v and weighted according to the cosine similarity score $w(u, v)$ of each sentence pair. The *Biased LexRank* $LR(u)$ of node $u \in V$ is defined as:

$$LR(u) = d \cdot \frac{b(u)}{\sum_{z \in V} b(z)} + (1 - d) \sum_{v \in adj[u]} \frac{w(v, u)}{\sum_{z \in adj[v]} w(v, z)} LR(v),$$

with $b(u)$ denoting the similarity of sentence u with the user's query and $adj[u]$ denoting the set of incident edges of node $u \in V$.

The probability d allows to control the balance between inter-sentence similarity and query-sentence similarity. In Otterbacher et al. (2009), the best results have been produced with d in a range of $[0.65, 0.95]$. Ibid., a similarity threshold for the insertion of edges into the graph has been introduced as a second parameter. The researchers found the best range for this threshold to be between 0.14 and 0.20.

Calculation of the Biased LexRank can be done efficiently by using the power method, which has been previously used to solve PageRank and derived problems. Kamvar et al. (2004) describes the method and its application to the PageRank computation in detail, making it easy to adapt the method for the calculation of Biased LexRank.

6.2.2 Evaluation Results

An evaluation of the three ranking algorithms should focus on the number of relevant sentences that received a high ranking position and thus are treated as important information. To achieve that, the common measure *precision at rank* could be applied, which divides the number of correct (i.e. relevant) sentences by the number of considered items (i.e. the rank of interest). This measure is mainly useful for evaluating the quality of the results on the first page of a search engine query, because the total number of considered items is fixed (a search engine usually presents the same number of items on a page for each query). If there are e.g. 3 relevant sentences within the top 10 ranked sentences, the *precision at 10* is $3/10 = 0.3$.

Unfortunately, there is no fixed number of sentences to be used for the summary. In fact, it depends on the query, how much information is retrieved and makes sense to use in the resulting summary. Another problem is that the number of relevant sentences varies a lot depending on the query. For a query with a total of 100 relevant sentences, the *precision at 10* is naturally higher than for a query with only 5 relevant sentences.

Above problems can be solved by using the *R-precision* score. *R-precision* equates to the *precision at rank* score, evaluated where rank is the total number of relevant sentences in the document collection. It is defined as follows:

Definition 20 (R-precision) Let $S = (s_1, s_2, \ldots, s_n)$ be a ranking, i.e. an ordered sequence of sentences, and $r(s)$ denote the relevance of sentence s with $r(s) = 1$ if s is relevant and $r(s) = 0$ otherwise. The total number of relevant sentences in the document collection calculates to $R = \sum_{i=1}^{n} r(s_i)$. The corresponding *R-precision* is:

$$R\text{-}precision(S) = \frac{\sum_{i=1}^{R} r(s_i)}{R}$$

Consider for instance a collection of 100 sentences, of which 30 have been annotated as relevant. The number of relevant sentences within the top 30 sentences of a ranking are then counted to 20, which results in an *R-precision* of $20/30 = 0.66$ (the *precision at 30*). A nice introduction to *R-precision* and some other evaluation measures for rankings can be found in Manning et al. (2008), chapter 8.

Figure 6.4 shows the *R-precision* scores for the three ranking algorithms. Again 8 queries of the training dataset (cf. section 2.5) have been used for the evaluation. The baseline (Position Rank) is only outperformed for the four queries `defQ-3`, `defQ-4`, `defQ-10` and `defQ-13`. A possible explanation for the relatively good performance of Position Rank could be that the training queries are rather less specific, but ask for general information on a certain topic, which is usually found at the beginning of a document. Another observation is the bad performance of the Biased LexRank approach that outperforms the other approaches only once for `defQ-10`. This is a surprising result as the method worked fine in Otterbacher et al. (2009) and by design includes the Similarity Rank method to some extent (as bias). Focusing on the similarity scores between the sentences thus does not seem to be a good idea with regard to this question answering system.

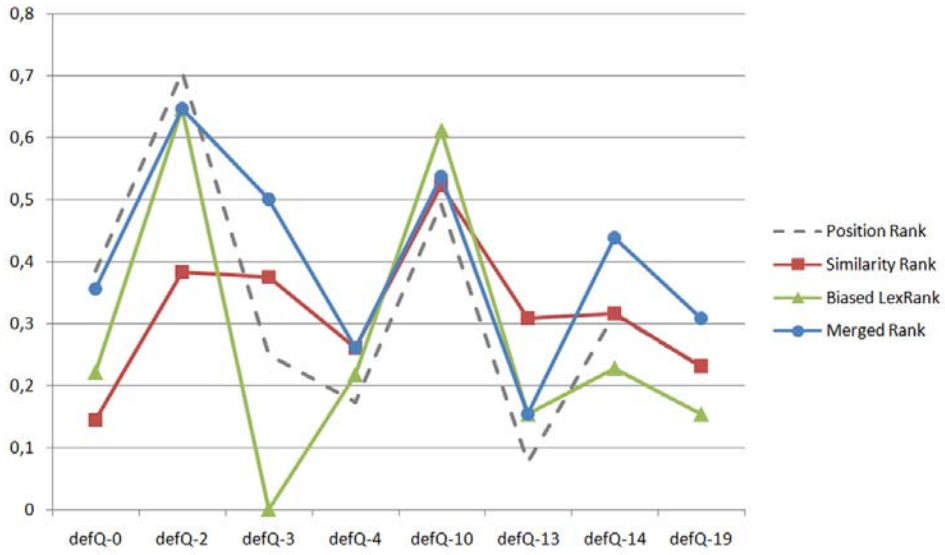


Figure 6.4: Performance of different answer ranking algorithms on 8 training queries

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
Position Rank	2	3	1	5	10	4	7	9	8	6
Similarity Rank	3	1	4	6	2	5	8	7	9	10
Biased LexRank	2	4	6	1	5	8	3	7	9	10
Average rank score	2.33	2.67	3.67	4.00	5.67	5.67	6.00	7.67	8.67	8.67
Merged Rank	1	2	3	4	5	6	7	8	9	10

Table 6.1: Example for the Merged Rank calculation

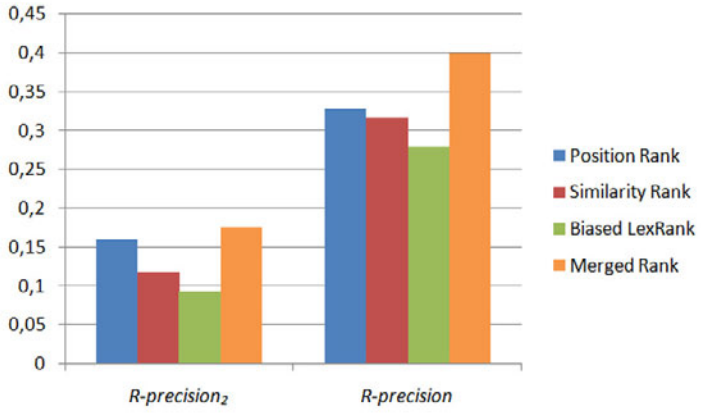


Figure 6.5: Comparison of the average R-precision of 4 answer ranking algorithms

If one looks at the rankings more closely, it turns out that the algorithms lead to overall different rankings. Although there are several similarities, the relevant sentences are highly distributed within the rankings. As relevant sentences are often ranked at top positions in only two of the algorithms, a combination of the approaches could be useful to benefit from the advantages of the individual algorithms. This observation has also been made during the topic clustering algorithms (cf. section 5.5) and seems to be a robust technique when working with heterogeneous data.

The combination method will be called *Merged Rank*. Its performance results are also shown in figure 6.4. The method simply averages the ranking positions of the three input rankings and reranks the sentences according to this average value. Consider e.g. the 10 sentences in table 6.1 and their rankings with Position Rank, Similarity Rank and Biased LexRank. The average rank score for sentence s_2 is e.g. calculated as $(3 + 1 + 4)/3 = 2.67$. The Merged Rank approach outperforms the baseline for 6 of the training queries. Similarity Rank and Biased LexRank are overall outperformed with the exception of one query each (`defQ-10` and `defQ-13`).

Figure 6.5 displays a comparison of the average *R-precision* of the four algorithms over the 8 training queries. Additionally, the *R-precision*$_2$ score has been calculated and averaged. *R-precision*$_2$ considers only definitely relevant sentences and is therefore in general lower

than the *R-precision* for all relevant sentences (possibly and definitely relevant). The modest performance of Similarity Rank and Biased LexRank can also be observed for *R-precision*$_2$. The best results of the four are in both cases those of the Merged Rank approach, which will therefore be used for the system.

6.3 Topic Labeling and Ranking

Clustering algorithms have been used in chapter 5 to identify different topics within the document collection. These topics should be shown together with the summary and allow the user to choose the topics of interest resp. the topic that was initially meant when posting the question. Both a topic label and a ranking of the topics is required for a comfortable use of the topic selection.

There are different methods for automatically assigning a label to a document cluster, like using the n most frequent words, extracting frequent noun phrases or using terms that can be found more likely in one of the clusters. While the first approaches use a token or phrase index, the latter relies on statistical significance. Applications of these methods are e.g. discussed in Chuang and Chien (2004) or Stefanowski and Weiss (2007).

For the system in this thesis a simple bag-of-words index is created for each cluster and the 3 most frequent words are used as cluster labels. The individual terms are separated with commas, like in the label *flash, download, player* that has been generated for the example query *"What is flash media?"* (`defQ-0`). Words with only 1 or 2 characters are omitted as well as numbers, punctuation and symbols.

The most promising topic should be the first one in the list of topics and its contents are to be displayed directly on the response page. The other topics are collapsed in the beginning and can be expanded by the user if necessary. Different criterion are possible to define the most promising topic. They can be mainly divided into the two techniques that consider either the cluster's size or the similarity to the query.

The cluster size can be measured by counting the contained documents, paragraphs, sentences or words. A ranking is then created that sorts the topics with descending order by their cluster size. The idea is that most of the retrieved information belongs to the user's query as the query was well chosen, such that the biggest cluster is also the most promising. There are several drawbacks of this method: If much information for a certain topic has been retrieved, which is not at all related to the query, the wrong topic is on top of the topic list. Another problem occurs if the clustering does not work well on a certain document collection. A usual result in these cases is the chaining effect that has been introduced in section 5.2.

For calculating the similarity score between a cluster and the query, there are several methods. Besides the choice of the definition of similarity between two texts (cf. section 3.4 for

an overview of possible measures) the combination of the similarity scores for each document can be chosen. Particularly, the three different approaches single-link, average-link and complete-link can be used, as introduced during hierarchical agglomerative clustering (section 5.2). Single-link employs the similarity score of the document that is most similar to the query, while average-link considers the cluster as a single document and uses the mean similarity score. Complete-link however uses the score of the least similar document. One drawback of this method can be observed if the cluster is big and maybe still contains several sub topics that could not be well divided. The cluster similarity is then composed of both the relevant and the irrelevant topic such that the cluster is maybe ranked at an average position although it contains the most prominent information. Another problem occurs if the requested information is very general or consists of only stop words and is thus not well classified by a similarity measure.

Although both the approaches are reasonable and could be useful for this thesis' system, only the method that ranks according to the number of contained documents is used here.

6.4 Summary Views

The multi-document summarization can now be achieved by combining the annotations of the previous components. In fact, the sentences will be ordered according to their rank, annotated by the SENTENCE RANKER component (section 6.2) and then concatenated to form a continuous paragraph of text that will be used as resulting summary. This simple summarization technique is known as *extractive summarization* and has successfully been used in question answering tasks before, e.g. in Liu et al. (2008). There are however also more elaborated summarization techniques, which have not been considered for this thesis. An overview on multi-document summarization techniques can be found in Mani (2001) and Torralbo et al. (2005).

In chapter 5, a topic for each document has been automatically identified and then labeled and ranked according to section 6.3. These topics can be used to filter the resulting summary for displaying only information about a certain topic. The same option can be used for the different data sources, the documents are taken from (currently Yahoo! Answers, FAQs and Wikipedia). It should thus be possible for the user to view for instance a summary exclusively about downloading Adobe's flash player, whose sentences have been taken only from the FAQ collection. With this type of filtering, the system gets highly flexible even if the initial result, which is the system's recommendation, does not contain the topic of interest. Filtering data sources is particularly useful for subjective queries as Wikipedia usually does not include subjective comments.

Continuous textual summaries are fine for answering definition questions. For list and factoid questions, there is however a better choice for presenting the answer. Kaisser et al. (2008) evaluated the ideal length and textual form for different question and answer types.

They found out that factoid questions like asking for a certain person, organization, place, time or number, a single phrase would be the ideal answer, while list questions that search for a resource, product or website are best answered by a list of result items or links. Although the main focus of this thesis are definition queries, there will be a *list view* and a *link view* on the result page that intends to cope with above issue. The user then has the possibility of switching from the *text view* for continuous reading to the *list* or *link view*.

While the link view simply contains a list of web links that have been found in the documents, there is no exact definition of what to use as list items in the list view. Like for topic labeling, frequent words or phrases (particularly noun phrases) can be used. Since the most frequent words are already included in the topic label itself, this is not an optimal choice to use in the list view because of the relatively low information gain. Noun phrases are usually a good choice, although they fail to answer questions like *"What are synonyms for (to) walk?"*, which expects a list of verbs as result.

Lin (2007) introduces a redundancy-based method for answering factoid questions. This system overcomes the problem of using frequent noun phrases by considering frequent n-grams, which are defined as follows:

Definition 21 (n-gram) Let $T = t_1 t_2 \dots t_m$ be a text of length $m \in \mathbb{N}$. An *n-gram*

$$g = t_i \dots t_{i+n-1} \qquad \text{with} \qquad i \in \{1, \dots, m-n\}$$

is a sub sequence of T that consists of n consecutive tokens. A 1-gram is usually called a *unigram*, a 2-gram a *bigram* and a 3-gram a *trigram*. The query *"What is flash media?"* consists for instance of three bigrams: *(What, is)*, *(is, flash)* and *(flash, media)*.

Consider the two phrases *"Obama runs for president"* and *"Yes, we can"*. Since the first contains a verb phrase and the second consist of stop words only, both the examples could neither be found within the most frequent words nor the most prominent noun phrases. n-grams however do not rely on part-of-speech tags or stop words, so both the phrases would be added to the list view if they tend to appear more often than other n-grams.

To find out, which n-grams are informative, Lin (2007) computes a TF-IDF-like score for each n-gram that is used for ranking the results. The original scoring model is slightly modified to work with the specificity scores that have already been calculated by the SPECIFICITY ANNOTATOR (introduced in section 3.2). The specificity value of each token in the n-gram is averaged and then multiplied with the number of occurrences of the n-gram within the document collection. Only n-grams with $n = 2, \dots, 5$, an average score of at least 1.0 and a minimum occurrence of 3 are considered as *frequent n-gram*. n-grams that start or end with a stop word are currently also filtered out.

As the scoring model is rather naïve, a dedicated evaluation has not been done yet. Manual review of the high scored n-grams suggests that a more elaborated method could definitely be useful in order to obtain better results. The quality of the actual available n-grams however seemed to be promising.

6.5 Graphical User Interface

The question answering system should provide a graphical user interface that allows the user to post a question and view the results. To achieve that, a web service QASRV has been created from the components described above and deployed on an Apache Tomcat application server. It can be accessed by the search-engine-like JSP application QAWEB. See section 2.4 for details on the system architecture.

It is not part of the project to implement or connect a document retrieval engine. But although there is no such component yet, the system has been designed to be able to add an engine later on. Until then, only the 60 questions from the evaluation corpus can be used. A full list of these questions can be found in appendix A.1.

The system consists of three JSP pages. The first comes with a form field to type in the question. A short list of clickable training questions is shown for convenience. The question is posted by using the *"Ask"* button that submits the form and thus invokes the web service. On the following JSP page, the user receives a permanent status feedback from the web service, while the components are executed and the summary is composed. A typical run takes about 90–120 seconds,[34] which strengthens the need of such a status page to inform the user about the progress and that the system is still alive and working. Figure 6.6 shows these first two pages in a web browser screenshot.

As soon as the web service has finished its computation, the user is redirected to the answer page, where the resulting summary is shown. The result consists of three different views:

For the *text view*, a short paragraph of text is composed from the most informative sentences that have been extracted from the retrieved documents. The sentences, which have been classified as relevant within the PASSAGE EXTRACTION task are sorted by their rank in descending order. The merged ranks, calculated in section 6.2, are used. Since the number of relevant sentences is highly different for each topic, not every sentence can be considered in order to provide a "readable" amount of text for each topic. In fact, only the top 10 sentences have been used for the system described here. Clicking on a sentence, brings up a detail pane with the context of the sentence, its determined topic and the original source it has been taken from. This is mainly necessary if the meaning of a sentence highly depends on the previous or following sentence, which is maybe not included in the summary.

The *list view* can be directly built from the results of the previous component, which annotates each n-gram of size 2–5 that occurs at least 3 times and has a score of at least 1.0. Only the top 40 frequent n-grams are used for this view. Each item is again clickable to reveal a detail pane with the n-gram's different contexts to get a better impression of what is meant by the n-gram or which fact it belongs to.

The *link view* consists of the top 200 links that have been identified by the URLANNOTATOR. The view is built exactly like the list view but provides also a "Follow" button to open the

[34] Calculation done on Intel Core2 Quad CPU, 2.4 GHz, 3 GB RAM, Windows Vista, Java 1.6.0.2

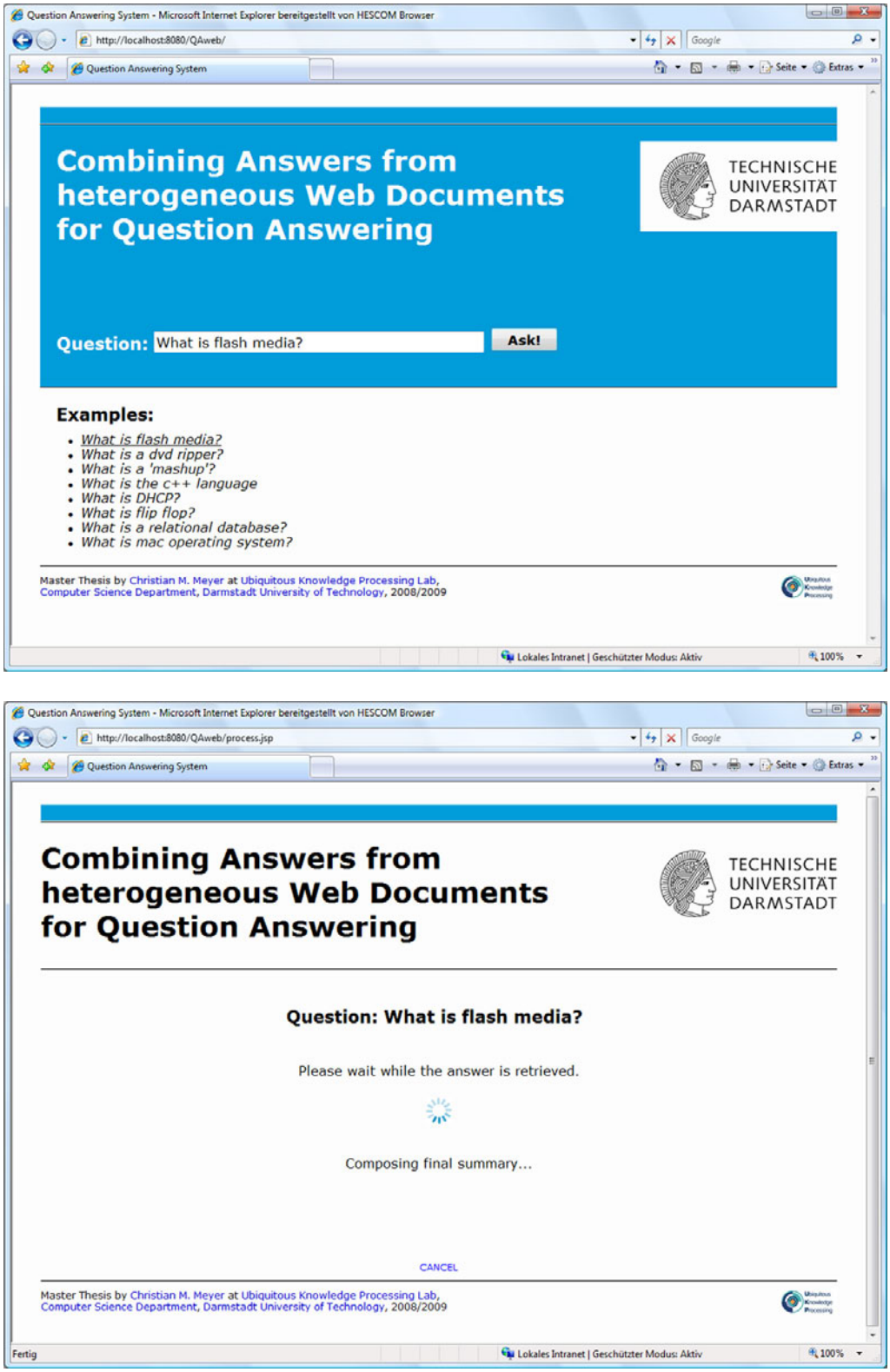


Figure 6.6: The system's web interface: start page (top) and status indicator (bottom)

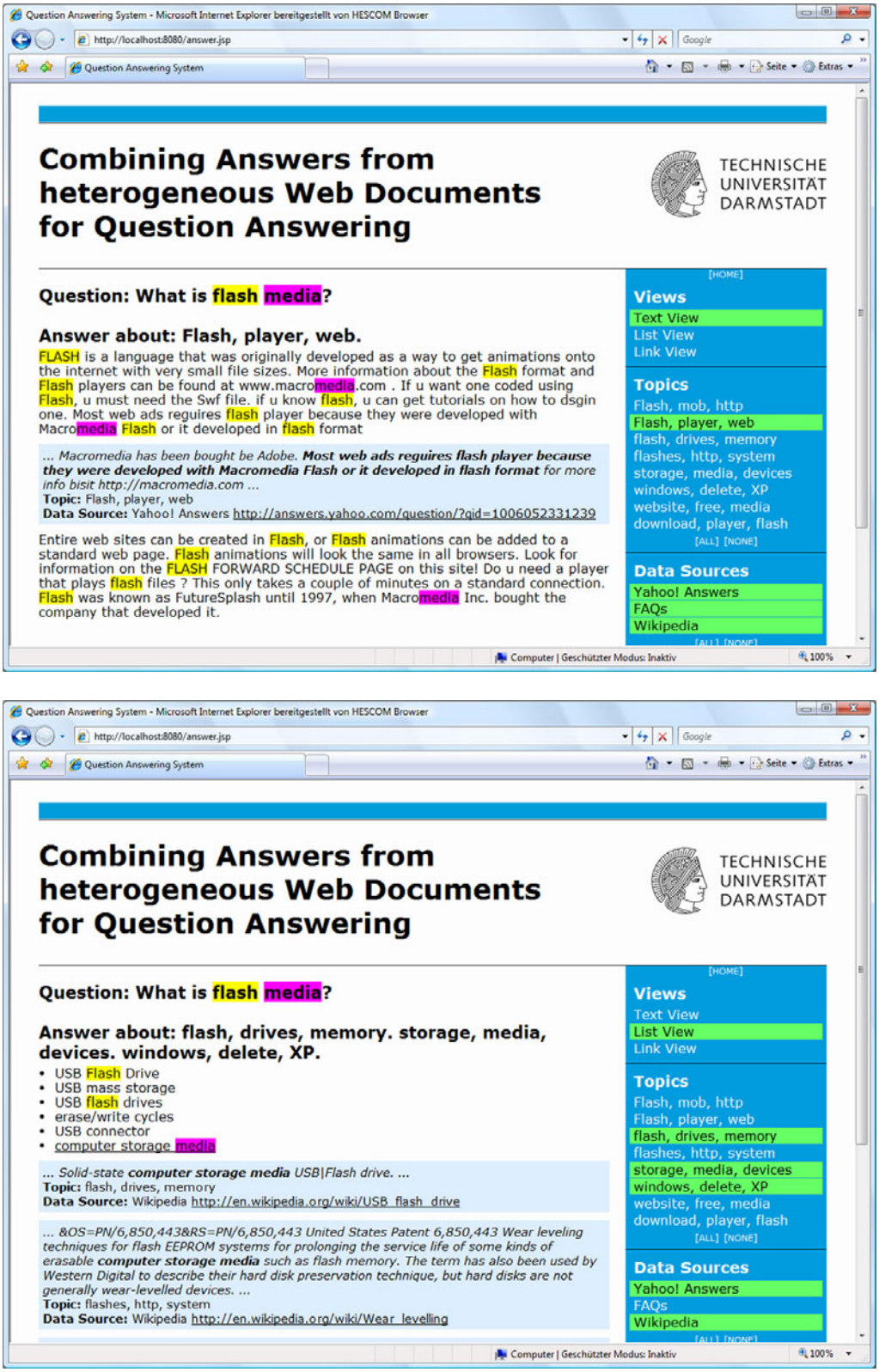


Figure 6.7: The system's web interface: text view (top) and list view (bottom) of the result page

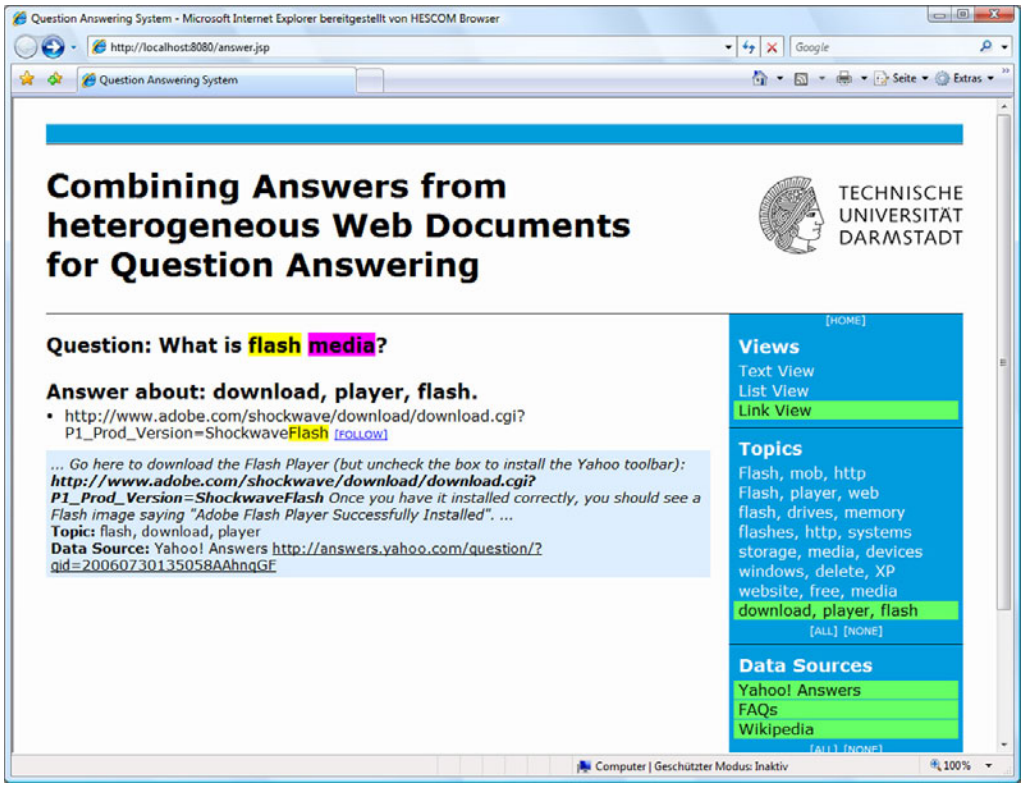


Figure 6.8: The system's web interface: link view of the result page

linked URL in a new browser window. Note, that URLs have been normalized to generate valid HTML code; if no protocol was given, `http` has been assumed.

The control bar on the right allows the user to switch between the three views. Certain topics or data sources can be excluded from the view by simply clicking on the corresponding toggle button—the green bar indicates that information from this topic resp. data source is currently shown. Within the control bar, there are also convenience links for expanding or collapsing all detail panes and for enabling/disabling the keyword highlighting. If the keyword highlighting is activated, each occurrence of a query token is highlighted in a certain neon color like the highlighting function of e.g. the Google Toolbar.[35] Only non-stop words are considered for highlighting to preserve the overview and clarity of the system's answer.

Figure 6.7 shows the text and list view for the example query *"What is flash media?"* (`defQ-0`). For the top image—the text view—the second topic *"Flash, player, web"* has been selected to display information about the Adobe Flash player. One of the sentences has been clicked on to open the detail pane. The second picture shows the list view for the same query. However, the topic selection has been changed to get information about flash memory. In total, three topics have been selected, whose frequent terms are combined to a single list. Frequent terms from the FAQ data source have been excluded from the view

[35] Google Search Toolbar – `http://toolbar.google.com`

for demonstration purpose. The detail pane for the term *"computer storage medium"* has been opened, which shows 2 occurrences of this *n*-gram within the original context of the document (2 more occurrences can be found after scrolling down a bit). Finally, the link view is shown in figure 6.8. The last topic *"download, player, flash"* was chosen for the link view, which contains only a single URL (in fact, it is a valid download link for Adobe's Flash player). The follow button can be used to open the link in a new window, while the detail pane shows the context of the link, which was initially found in a Yahoo! Answer.

6.6 Evaluation Results

Evaluating the results of an automatic summarization algorithm is hard, since the definition of a good summary is mainly subjective. A good summary of course needs to contain all of the most important information and be shorter in length than the input documents. The expected length depends both on the amount of data that is to be summarized and on the purpose of the summary. For the question answering system here, the size of the summary was chosen to approximately fit on a web page such that scrolling is not necessary. The presented information should be basically free of redundancy to keep the summary as short as possible. Besides these content-related aspects, a good summary also needs to be of consistent style and fluent readability.

Manual evaluations are only possible for small-scale applications due to the time-consuming annotation and revision work. According to Lin (2004a), a manual evaluation of the results of the Document Understanding Conference[36] would take over 3,000 hours of human effort. A bunch of automatic evaluation methods has therefore been proposed to ease the extensive manual process. Possibly the most common toolkit for the evaluation of automatic summarization is ROUGE,[37] which has been introduced in Lin (2004b). ROUGE allows to compare two summaries, mostly based on the overlap of *n*-grams. If a reference summary (i.e. a *model* in ROUGE terminology) is given, the evaluation toolkit is able to determine the quality of the corresponding automatically computed summary (resp. *peer*) by comparing it to the reference summary and check for the overlap percentage. However, to capture the absolute quality of a summarization system, the results of at least one other system has to be compared against the reference summary, too.

For the evaluation here, the 20 definition queries have been used. A full list of these queries can be found in appendix A.1. The reference summaries are taken from *Ask.com* and *Answerbag*.[38] *Ask.com* offers a specialized Q&A search mode that implements a question answering system and is able to retrieve an answer for most of the evaluation questions. Using *Answerbag*, a collaborative question answering platform, allows to find exact answers for the evaluation questions, since they originally have been taken from *Answerbag*. Both the

[36] Document Understanding Conference – http://duc.nist.gov
[37] Recall-Oriented Understudy for Gisting Evaluation (ROUGE) – http://berouge.com
[38] Ask.com – http://www.ask.com; Answerbag – http://www.answerbag.com

systems have not been used as a data source, which was another criteria for choosing a reference corpus (as *Ask.com* produces links to arbitrary web pages, links to Wikipedia, Yahoo! Answers or dedicated FAQ pages have been omitted). If multiple appropriate answers could be found, up to four different versions per platform have been used.

As comparable systems, *START* and *MEAD* have been chosen. *START*[39] is an open-domain question answering system that extracts data from several databases. It has been introduced in Katz (1997) and Katz et al. (2002), while its relevance for this thesis has been addressed in section 2.6. Reference summaries for the training questions could be generated by simply posting the answer to the web interface and copying the result to a plain text file. START was able to generate an answer for 16 of the 20 training questions. *MEAD*[40] is a general purpose multi-document summarization system that has been introduced in Radev et al. (2004). For each training question, *MEAD* was supplied with all of the retrieved documents (converted to *MEAD*'s desired input format) and parameterized to produce a summary of about 10 sentences. This should allow best comparability with the other systems. Each of the 20 training questions could be processed and led to an appropriate answer.

Table 6.9 shows the recall, precision and F-measure (with $F = 1$) scores of the ROUGE-1, ROUGE-2, ROUGE-L and the ROUGE-W measure. The scores have been averaged over the 20 training queries. While ROUGE-1 and ROUGE-2 are instances of ROUGE-N with parameter $N = 1$ and $N = 2$ that measures the co-occurrence of N-grams, ROUGE-L determines the longest common subsequence of the two input summaries. ROUGE-W is a weighted version of ROUGE-L with weighting parameter 1.2; cf. Lin (2004b) for the exact definition of the four measures. Besides the performance of *START* and *MEAD*, three different configurations for the system of this thesis have been tried. *QA* includes the text view of each topic individually. For the query *"What is flash media?"* (`defQ-0`) there are 5 summaries with topic labels *"Flash, mob, http"*, *"flash, drives, memory"*, *"flash, systems, file"*, *"Flash, download, website"* and *"PlayStation, camcorder, Portable"*. As the topics have been ranked according to their importance, the first topic will most likely contain the best answer. The *QA-first* dataset therefore uses only the first topic's summary. Since the user is able to manually select the topic of interest, also the performance of the best answer is compared to the other systems. The best answer in this case is the one with the highest recall score of the *QA* run. This dataset will be called *QA-best* in the following.

The highest ($1^{\bullet}$) and lowest ($0_{\diamond}$) scores of each row have been highlighted in the data table. Generally, *MEAD* obtains the highest recall, *START* the highest precision and *QA-best* the highest F-measure. The summarization system *MEAD* focuses on composing the most important information and therefore achieves an excellent recall. Manual inspection however showed that there was some redundancy in the summaries and the sentences were often of different topics, which reduces the readability and understandability of the summary. *START* uses redacted texts from the World Wide Web, like excerpts of *Wikipedia*,

[39] START, Natural Language Question Answering System – `http://start.csail.mit.edu`
[40] MEAD – `http://www.summarization.com/mead`

Measure	START	MEAD	QA	QA-first	QA-best
ROUGE-1 – Recall	0.26270$_\diamond$	0.41681$^\bullet$	0.30568	0.35244	0.39607
ROUGE-2 – Recall	0.07031	0.07762$^\bullet$	0.04517$_\diamond$	0.06401	0.07464
ROUGE-L – Recall	0.24432$_\diamond$	0.38586$^\bullet$	0.27930	0.32378	0.36131
ROUGE-W – Recall	0.08793$_\diamond$	0.12384$^\bullet$	0.09049	0.10446	0.11791
ROUGE-1 – Precision	0.32325$^\bullet$	0.11284$_\diamond$	0.19482	0.18621	0.20377
ROUGE-2 – Precision	0.09266$^\bullet$	0.02097$_\diamond$	0.03354	0.03314	0.03864
ROUGE-L – Precision	0.29742$^\bullet$	0.10480$_\diamond$	0.17970	0.17169	0.18631
ROUGE-W – Precision	0.20239$^\bullet$	0.06279$_\diamond$	0.11126	0.10263	0.11276
ROUGE-1 – F-measure	0.25150	0.17022$_\diamond$	0.21900	0.23431	0.25682$^\bullet$
ROUGE-2 – F-measure	0.07164$^\bullet$	0.03168$_\diamond$	0.03274	0.04209	0.04872
ROUGE-L – F-measure	0.23270	0.15794$_\diamond$	0.20091	0.21567	0.23452$^\bullet$
ROUGE-W – F-measure	0.10708	0.07865$_\diamond$	0.09107	0.09882	0.10904$^\bullet$

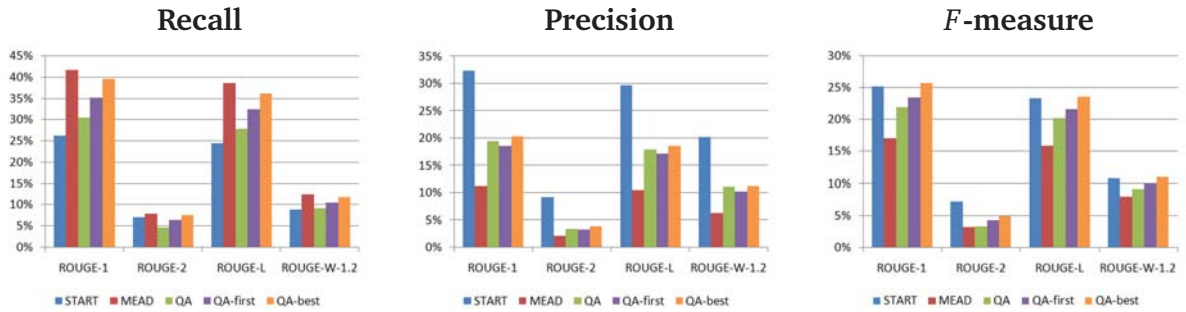


Figure 6.9: Evaluation of the different summarization approaches

Webopedia, *WhatIs.com* or the *Merriam-Webster Dictionary*.[41] Summaries are therefore understandable and usually correct. Most of the time, the answers are however quite short like only 1 or 2 sentences and do thus not fully reflect all the information of the reference corpus. In fact, *START's* recall is almost fully outperformed by all *QA* variants. The same applies to *MEAD's* precision, which is fully outperformed by *START* and all *QA* variants. The overall good F-measure performance of *QA-best* shows that the system, developed in this thesis, combines advantages of the two other systems and provides acceptable results in both precision and recall. The difference between *QA-best* and *QA-first* is only 1.7% in average and thus leads to a good topic ranking. Particularly, the best topic is also the first for 50% of the queries and the second for 25%. Only in one query the best topic is ranked to a position greater than 3.

The differences in precision, recall and F-measure between *QA-best* and *START* as well as *QA-best* and *MEAD* have then been tested for statistical significance. Table 6.2 shows these results. The significance has been determined by a paired t-test. Values with at least 99.0%

[41] Webopedia – http://www.webopedia.com; WhatIs.com – http://whatis.techtarget.com; Merriam-Webster Dictionary – http://www.m-w.com

Measure	QA-best − START	QA-best − MEAD
ROUGE-1 – Recall	0.13337*	−0.02074
ROUGE-2 – Recall	0.00433	−0.00298
ROUGE-L – Recall	0.11699*	−0.02455
ROUGE-W – Recall	0.02998*	−0.00593
ROUGE-1 – Precision	−0.11948	0.09093*
ROUGE-2 – Precision	−0.05402	0.01767*
ROUGE-L – Precision	−0.11111	0.08151*
ROUGE-W – Precision	−0.08963	0.04997*
ROUGE-1 – F-measure	0.00532	0.08660*
ROUGE-2 – F-measure	−0.02292	0.01704*
ROUGE-L – F-measure	0.00182	0.07658*
ROUGE-W – F-measure	0.00196	0.03039*

Table 6.2: Recall/Precision/F-measure improvement of QA-best and its statistical significance

confidence have been highlighted in the table and marked with an asterisk (*). The test reveals that *QA-best* has a significantly better precision and F-measure than *MEAD*, while its recall is also better than *START*'s. The higher recall scores of *MEAD* and the higher precision of *START* turned however out to be not statistical significant (the confidence is between 77.1% and 98.9%), which strengthens the usefulness of the system. Note that although the difference of the averaged precision values of *QA-best* and *START* is quite high, *START* was not able to retrieve an answer for 4 of the 20 queries and thus leads to a confidence of only 98.4%.

A representative of each system type for the training query *"What is DHCP?"* (`defQ-10`) can be found in figure 6.10. Note that some of the texts have been shortened (indicated by [..]) for convenience. Besides the 4 system results in the figure, there are 2 answers from *Ask.com* and 4 from the *QA* dataset, which have been used for the evaluation. *START* shows the definition of DHCP from the corresponding Wikipedia article. The text is informative and redundancy-free, but some information are missing due to the small length, like who is generally using DHCP or what is needed to use DHCP. The summaries of *MEAD* and *QA-first* are longer and thus are able to provide more information. They both contain some redundancy, e.g. that DHCP is short for Dynamic Host Configuration Protocol. All the three systems are however definitely providing comparable or even better results than the user-generated reference answer of *Answerbag*.

Figure 6.11 shows 4 answers for the training query *"What is a 'mashup'?"* (`defQ-3`). In the evaluation dataset there are beyond that 2 answers from *Answerbag*, 1 from *Ask.com* and 2 from *QA*. *START* again shows the corresponding Wikipedia article, which is a disambiguation page in this case. Although this offers a good overview, what the meaning of the term could be, the provided definitions are again quite short such that a new query would

DHCP (Dynamic Host Configuration Protocol) is a communications protocol that lets network administrators centrally manage and automate the assignment of Internet Protocol (IP) addresses in an organization's network. DHCP is short form of dynamic host configuration protocol.This protocol is used by ISP (internet service provider) or companies whose network is very vast its function is to provide ip address to the client automatically.If you are using DHCP the you dont need to give ip address to the clients it will automatically give the ip address to its clients with in the particular range and this makes the administrators job more easy as he need not go to each and every system in the network and configure the ipaddress. To implement DHCP, you must have a DHCP server. [..]

(MEAD)

Through the DHCP server, you can control the assignment of addresses and the configuration of other TCP/IP protocol parameters in whatever way is appropriate for your network and your organization. The DHCP server can be setup to hand out or "lease" an IP number to a client for a specific amount of time. DHCP (Dynamic Host Configuration Protocol) is a way to automatically configure TCP/IP on client computers on a network. DHCP just determines how your devices will receive an IP address. Thus, the DHCP server acts as the network administrator's agent for managing the configurations of DHCP clients. DHCP is short form of dynamic host configuration protocol. [..]

(QA-first)

Dynamic Host Configuration Protocol (DHCP) is a protocol used by networked devices (clients) to obtain various parameters necessary for the clients to operate in an Internet Protocol (IP) network. By using this protocol, system administration workload greatly decreases, and devices can be added to the network with minimal or no manual configurations.

(START)

DHCP stands for Dynamic Host Control Protocol. In order to configure a computer to access the Internet, you need about five numbers, including its IP address. You can set these all up by hand, which is a real pain unless you are a serious Geek. DHCP is a simple way for the computer to send out a call asking any machine listening to send it those numbers so that it can configure itself without bothering the user (who probably wouldn't know what to do anyway). It is the way almost all "end user" machines are configured on the Internet.

(Answerbag)

Figure 6.10: Different summaries for query "What is DHCP?" (defQ-10)

"Red Red Wine" is a song originally written by Neil Diamond that was then covered by Tony Tribe and more famously by UB40 in later years. Inside Out was a Hardcore and/or Metalcore band from Orange County, California - notable for being the first band fronted by Zack de la Rocha, later of Rage Against the Machine. :Mother Nature's Son is also the title of an episode of Only Fools and Horses "Mother Nature's Son" is a Lennon/McCartney song, released by The Beatles on the White Album. The Anti-Hit List is a weekly music column by Canadian music critic John Sakamoto. Electronic civil disobedience, also known as ECD or cyber civil disobedience, can refer to any type of civil disobedience in which the participants use information technology to carry out their actions. [..]

(MEAD)

This is also known as a mashup. In practice, since Web 2.0 is a title given by web designers it is more to do with the underlying technology. Without getting too technical, web services can 'talk' to each other. AJAX is just a fancy way of saying that information is passed to and from the web browser using Javascript rather than HTML. Finally, Web 2.0 is considered by some to be the way in which these individual services and data stores can now intract with each other. The theory here is that each individual web service becomes much more useful when combined with others. This would be a job for a Google maps mashup, which combines Google maps and other data. * http://www.themolu.com Molu - The search spider Mashup based search engine. [..]

(QA-best)

Mashup may refer to:
• Mashup (digital), a digital media file containing any or all of text, graphics, audio, video, and animation, which recombines and modifies existing digital works to create a derivative work.
• Mashup (music), the musical genre encompassing songs which consist entirely of parts of other songs
• Mashup (video), a video that is edited from more than one source to appear as one
• Mashup (web application hybrid), a web application that combines data and/or functionality from more than one source

(START)

The Wikipedia defines a mashup as 'a website or application that combines content from more than one source into an integrated experience'. This is a useful basic definition but a more complete enterprise-relevant description would be: A mashup is a user-driven micro-integration of Web-accessible data. While short, this definition contains a number of important points worth considering: "User-driven" - Mashups are executed for the user, not the by black-box back-end integration systems such as ESB, BPM, BPEL, etc. In this sense, a mashup must be completed by the users themselves. Without this guiding principle, we are merely sending the users back to IT for more development. [..]

(Ask.com)

Figure 6.11: Different summaries for query "What is a 'mashup'?" (defQ-3)

be necessary for retrieving more information. *MEAD*'s summary is mainly about musical mashups, except the last sentence, which is about civil disobedience and thus irrelevant for the query (regardless of its meaning). The remaining summary could be useful but does not explain what a mashup is, in fact the term "mashup" is not even included in the summary. As 16 of the 25 retrieved documents are about musical mashups (compared to 7 about mashups in the World Wide Web), a summary about this topic is definitely a good guess. In fact, also the *QA* dataset's first topic is about musical mashups. The original query was however taken from the computer category of *Answerbag*, so the intended meaning is the definition of a web mashup. Within the *QA* result, there is a topic that contains an answer about web mashups and that can be selected by the user. Although the summary is not very well structured, the user can find out that it is possible to combine different web services in a so called mashup, which is basically the information provided also in *START* and in the reference summary of *Ask.com*. The *QA* result however names the Google Maps mashup and the Molu search engine[42] as two concrete mashup examples. The user also learns that web mashups must have something to do with HTML, AJAX and Javascript. This technical aspect can neither be found in the *START* result nor in any of the reference summaries.

The query *"What does the abbreviation IDK stand for?"* (`defQ-1`) could not be answered by *START*. The *MEAD* summary contains several sentences about abbreviations, but none of them explains what IDK could stand for. A similar text could be observed in *QA-first*, which is a general text about different abbreviations—none of them about IDK. The second topic however consists of only the single sentence *"IDK means, I Don't Know"* that perfectly answers the question. Table 6.12 shows the corresponding answers from *MEAD*, *QA* and the reference corpus.

As regards list and factoid questions, the results have been manually inspected. Figure 6.13 shows the list and link view for the list question *"Where do you search for downloading your books of interest?"* (`listQ-0`) and the text and list view for the factoid question *"Where do I find my Computer Processing Unit(CPU) on my computer?"* (`factQ-18`). The link view turns out to be helpful for the list question as there are some good links like to the American Mathematical Society (AMS) or the Scientific American (SciAm), which offer downloadable books. There are however also some links that are not related to the query, e.g. the link to BMW, a German car manufacturer. Most documents addressed search engines and how to find books in them, so the list view contains especially these terms.

The list and link view is not very helpful for the factoid question `factQ-18`, because they both contain only general information about the CPU. The text view however contains the answer, i.e. that the 'CPU is located on the motherboard'. Considering that, it seems that there is no exact type of view for a specific question type. The correct or best answer can rather be included in any of the views and thus emphasizes the usefulness and flexibility of the provided GUI.

[42] Molu, a meta search engine for different types of data – `http://www.themolu.com`

There are two legends you can use with the maps: the original key and a black and white key , which was developed specifically for use with black and white copies of Sanborn maps. Below are some of the most commonly used abbreviations for personal ads and what they mean: TLA is a three-lettered abbreviation for Three-Letter Abbreviation or Three-Letter Acronym. MOS is a standard motion picture jargon abbreviation, used in production reports to indicate an associated film segment has no synchronous audio track. * MOS may stand for "Minus Optical Stripe," a note from a production sound mixer, notifying recipients that he or she did not expose an optical sound track for a particular scene or take [..]

(MEAD)

Many words and abbreviations have been in general use, but are not commonly used as of 2006: These abbreviations have no relationship to letters in postal codes, which are assigned by Canada Post on a different basis than these abbreviations. These abbreviations allow automated sorting. The use of X as an abbreviation for "cross" in modern abbreviated writing (e.g. This abbreviation is most commonly used in notations for mathematics or qualifications. ".." is an abbreviation for ", et cetera" or ", etc." when it is used at the end of a sentence, as in: The most common Latin words and abbreviations still in use are: [..]

(QA-first)

IDK means, I Don't Know

(QA)

Acronym	Definition
IDK	I Don't Know
IDK	I Didn't Know
IDK	Interface Development Kit
IDK	I Dont Kare
IDK	Internal Derangement of the Knee
IDK	In Da Kitchen
IDK	Individual Decontamination Kit

(Ask.com)

Figure 6.12: Different summaries for query "What does the abbreviation IDK stand for?" (defQ-1)

- Web search engine
- World Wide Web
- Web crawling
- major search-engines
- Site Search
- download pages
- Web crawler
- fraction of the Web
- Search Engine
- web crawler
- Book Search

(listQ-0, list view)

- http://www.ams.org/featurecolumn/archive/pagerank.html
- http://news.bbc.co.uk/1/hi/technology/6198244.stm
- http://www.sciam.com/article.cfm?chanID=sa006&articleID=0006304A-37 [..]
- http://www.sitepoint.com/article/indexing-limits-where-bots-stop
- http://slashdot.org/article.pl?sid=06/02/05/235218
- http://bmw.de
- http://www.seroundtable.com/archives/003133.html
- http://click4thecause.live.com
- http://www.charitycafe.com
- http://www.goodsearch.com
- http://Yahoo.com

(listQ-0, link view)

- central processing unit (CPU
- processing unit (CPU
- Central Processing Unit
- Intel Pentium
- modern CPUs
- CPU design
- Central processing unit
- central processing unit
- unit (CPU
- Processing Unit
- Central Processing

(factQ-18, list view)

The computer CPU is responsible for handling all instructions and calculation it receives from other hardware components in the computer and software programs running on the computer. (See also CPU design and computer architecture.) The system unit is simply the whole computer, meaning the computer case with all components installed. In computer engineering, an execution unit is a part of a CPU that performs the operations and calculations called for by the program. A control unit is the part of a CPU or other device that directs its operation. The control unit is a part of the Central Processing Unit (The Processor). Also known as a processor or microprocessor the CPU was first developed by Intel in 1974 and is short for Central Processing Unit. At one time control units for CPUs were ad-hoc logic, and they were difficult to design. The CPU is located on the motherboard and will have a cooling unit mounted on top of it. All types of control units generate electronic control signals that control other parts of a CPU.

(factQ-18, text view)

Figure 6.13: Example results for the list and factoid queries listQ-0 and factQ-18

Chapter 7

Conclusion

The goal of this thesis was the implementation of a question answering system that combines answers from the three heterogeneous data sources Wikipedia, Yahoo! Answers and a collection of Frequently Asked Questions (FAQ) in a multi-document summary. The summary should initially provide an answer to definition questions and then be extended for factoid and list questions.

Based on a training set of 20 definition, 20 factoid and 20 list questions and the corresponding result of a document retrieval engine for each data source, I identified several challenges resulting from the heterogeneity of the data, like differences in format, length, style, focus, type and number of errors. I defined three main tasks to solve these issues: *passage extraction*, *topic clustering* and *summarization*.

For evaluating the system tasks, I have annotated eight of the definition questions by assigning a relevance score between 0 (irrelevant) and 2 (definitely relevant) to each sentence as well as a topic to each document. I then built a component to read the retrieved documents for a single query from an XML file. UIMA was chosen as a framework for annotating and processing large amounts of data. The data is segmented into paragraphs, sentences and tokens. For each token, the part-of-speech, the lemma and a specificity score is determined. A combination of DKPro and my own components are used for these preprocessing steps. Part-of-speech and lemma annotations are created by parsing the results of TreeTagger, while the specificity score calculation is based on the token's frequency in the Wortschatz corpus. Finally, I disambiguated the sense of each token by comparing the surrounding document text with the sense glosses in WordNet using Lesk's algorithm.

After preprocessing, I applied a passage extraction technique on the document collection that started with removing identical duplicates from the collection to reduce the calculation time of the upcoming components. Irrelevant paragraphs have then been filtered by calculating the semantic similarity score between the paragraph's text and the query. I found a high correlation between WordNet-based measures and decided to use only the measures defined by Leacock/Chodorow and Resnik together with the cosine similarity score. The objective of the paragraph filter was removing a high number of irrelevant paragraphs while

keeping most of the relevant ones. I have optimized the component on one of the training queries and found the best results for removing a paragraph if its cosine similarity score is below 0.065, its Leacock/Chodorow score is below 0.513 or its Resnik score is below 0.06. Evaluating this configuration on the eight queries of the evaluation corpus, led to an average precision of 97% while removing 185.3 irrelevant and only 3.5 relevant paragraphs per document collection in average (out of 286.1 available paragraphs).

I then thought about a more elaborated approach for filtering the remaining data. I have chosen to use AltaVista-based Pointwise Mutual Information (PMI) and Wikipedia-based Explicit Semantic Analysis (ESA) on the sentence level. Optimization led to a threshold of 0.15 for PMI and 0.017 for ESA, while the evaluation resulted in an average precision of 92%. The component thus removed 68.5 irrelevant and 2.6 relevant sentences per document collection in average (out of 342.2 available sentences). To improve that, I focused on finding natural boundaries between relevant and irrelevant passages by applying a Hidden Markov Model (HMM) that has been trained by He et al. (2004). I tried 6 different scaling functions to convert the PMI and ESA similarity scores to an emission probability for the HMM. An evaluation showed that the results can be improved by using a Sigmoid function (i.e. an s-curve) as scaling function. The average improvement in precision turned out to be 7%, such that the passage extraction task could be concluded with an average precision of 98% and a recall of 81%.

The topic clustering task aims to automatically determine the topic of each document and thus allow the user to select the topic of interest if the query was ambiguous. I have implemented the two well-known clustering techniques hierarchical agglomerative clustering (HAC) and k-means clustering and included an existing implementation of Newman's community-based clustering approach. The evaluation on the eight training queries showed good purity scores for complete-link and average-link HAC and for Newman's approach, while the number of generated clusters was too high in the HAC algorithm. I then experimented with a combination of the clustering results of the three algorithms and found out that especially the combination of k-means and Newman improved the results of the individual algorithms and outperformed the baseline's purity scores in average by 10%. The mean clustering purity was 88.4% for this merged clustering algorithm. As baseline, the document's data source has been used to form three data source clusters.

To increase the quality of the resulting summary, duplicate and invalid sentences have been removed. For removing duplicates, I used a simple hashing function, while invalid sentences have been found by checking each sentence to have at least one noun and one verb. The extractive summary is then prepared by ranking the sentences according to their importance and relevance to the query. I have used the position of a sentence within the document as a baseline system (the earlier a sentence appears in the document, the more important and relevant it is). After that, I compared the baseline with a ranking based on the similarity scores of the passage extraction task (Similarity Rank) and the result of the Biased LexRank algorithm, which I implemented as suggested by Otterbacher et al. (2009).

I found out that none of the two approaches outperformed the baseline; in particular, the elaborated Biased LexRank method produced the worst results in average. Each approach however has shown its advantages in certain situations, so I tried a combination of the rankings by calculating the mean of the three individual ranks and reordering the sentences according to this new merged rank. The new ranking outperformed the three input algorithms in almost every query and led to a 7–12% higher R-precision score on the 8 training queries.

The identified topics have then been ranked according to the number of documents that they contain. I also determined topic labels by using the three most salient words of the contained documents. While the ranked sentences directly form an extractive summary that works fine for answering definition questions, I thought about a different method for list and factoid questions: I identified the most frequent n-grams within the documents and presented them as a list of important terms with respect to the given query. The n-grams are ranked according to their occurrence and their average specificity. The configuration of using n-grams of size 2–5 with a minimum average specificity of 1.0 and at least 3 occurrences has been chosen manually. Looking at the available n-grams, the method promised good results, but the actual results seemed to be well improvable by choosing a better scoring model. A quantitative evaluation has therefore been postponed to allow a comparison with a better model at a later time.

Finally, I have implemented a graphical user interface to display the summarization results. The main application is packaged in a web service QA$_{\text{SRV}}$, while the front end is deployed as the JAVA web application QA$_{\text{WEB}}$ that provides three JSP pages. On the first page, the user posts the query, which is submitted to the second JSP page that invokes the web service and displays status messages to the user. After completing the calculation, the user is redirected to the third page and can choose between three views for the answer. The text view contains the continuous textual summary based on concatenation of relevant sentences with high ranking scores. The list view displays the list of frequent n-grams, which is useful for factoid and list questions and the link view shows a list of all URLs that have been found in the documents. I focused on a flexible and fast interface that is easy to use. In fact, the user can exclude sentences or list items from one of the data sources or show only information on a certain topic. Each sentence or item additionally provides a detail pane that can be opened by simply clicking on the element. It shows the context, the topic and the data source of the element as well as a link to the original document, it has been taken from.

The final text summaries have been compared with the results of the *START* question answering system and the *MEAD* multi-document summarization system. I have chosen to calculate the ROUGE-N, ROUGE-L and ROUGE-W scores between each system and a reference corpus, which has been created from user-contributed answers on *Answerbag* or that have been found in the *Ask.com* Q&A engine. The evaluation showed that my system achieved the highest F-measure scores and thus provides complete, reliable and under-

standable information. For summaries of the *MEAD* system, precision was lower than in both the other systems, while *START* had the lowest recall scores in average. A paired t-test showed that the improvements in precision and F-measure of my system over *MEAD* were statistically significant with a confidence of over 99.0%. The same applied to the comparison of my system's recall scores with the *START* results. Better precision of *START* or better recall scores of *MEAD* achieved however only a confidence between 77.1% and 98.9% and thus were not very significant. Manual inspection also showed that the summaries of my system usually provided the necessary information for the question and succeeded to answer questions that could not be answered at all by *START* or not answered appropriately by the *MEAD* summary.

Further work and extensions of the system can be done in many components. Besides attaching the document retrieval engine (which was not part of the thesis) for responding to arbitrary questions, calculation time is an issue. Currently, the system takes 90–120 seconds, but implies some query-specific caching mechanisms that cannot be used for arbitrary questions. I have identified the bottleneck during the calculation of the PMI and ESA score, during k-means clustering and during the calculation of the Biased LexRank scores. As each task uses a combination of different measures or algorithms, it is easily possible to remove some of the components in favor of a faster calculation. While working with heterogeneous data, I found however out that a combination of different approaches is crucial to get stable results.

Improvements of the data quality could be obtained by including more data sources and by using a better sentence splitter, tokenizer and part-of-speech tagger. For the passage extraction task, the HMM can be trained on the evaluation data to get better results. For calculating the document clusterings an alternative similarity measure can be tried. The topic labeling algorithm also offers alternatives like using noun phrases or terms that appear statistically significant in a certain document. The method to find frequent n-grams can be improved by developing a better scoring model and quantitatively evaluating the results compared to a baseline. The support of list and factoid questions can also be further improved and evaluated.

Appendix A

Evaluation Data

A.1 Example Queries

The following list of example queries has been used for testing, training and evaluating the question answering system. The queries have been taken from the "Computer" category of the *Answerbag* platform by Kateryna Ignatova. The original URL was:
http://www.answerbag.com/c_view/20_new

— 20 Definition Queries —

ID	*Query*
defQ-0	What is flash media?
defQ-1	What does the abbreviation IDK stand for?
defQ-2	What is a dvd ripper?
defQ-3	What is a 'mashup'?
defQ-4	What is the c++ language
defQ-5	What is asp?
defQ-6	What is RAM?
defQ-7	What is DNS zone?
defQ-8	What is relay agent?
defQ-9	Whats meant by full duplex in net working
defQ-10	What is DHCP?
defQ-11	What is promiscuous mode?
defQ-12	What is the definition of an exclusive relation?
defQ-13	What is flip flop?
defQ-14	What is a relational database?
defQ-15	What is a dropper?
defQ-16	What is an AT slot?
defQ-17	What are dead pixels?
defQ-18	What is Blue-ray Disc Technology?
defQ-19	What is mac operating system?

— 20 LIST QUERIES —

ID	*Query*
listQ-0	Where do you search for downloading your books of interest?
listQ-1	Where can I find and download a free text-recognition image-reader that somehow transposes text in images onto Microsoft Word (and any other word-processing document?)
listQ-2	Where can I download a .flv to video(eg. wmv) converter?
listQ-3	What website can give me a free antivirus program?
listQ-4	Where can I download kidpix 4 for free?
listQ-5	Where can i find free flash video tutorials for download
listQ-6	Are there any free programs that will convert .3gp files into .wmv?
listQ-7	Where online can I sell my used stuff?
listQ-8	What are three programming languages that you think every programmer should know?
listQ-9	What are the different medias used in networking
listQ-10	4 main operational statements in SQL?
listQ-11	What are some good free spyware removal programs?
listQ-12	What substrate materials used in computer chip production?
listQ-13	WHAT ARE THE DIFFERENT TYPES OF MEMORY
listQ-14	What are the characteristics of postsript printer language
listQ-15	What are the main types of RAM?
listQ-16	Which laptops have the best battery life?
listQ-17	WHICH WEBSITE CAN I LEARN TO TYPE FASTER
listQ-18	What are the symptoms of computer being infected by boot sector virus?
listQ-19	What are the things that I must consider for DVD handling and storage?

— 20 FACTOID QUERIES —

ID	*Query*
factQ-0	On average, how many dollars per gigabyte (or gigabytes per dollar) does RAM cost today?
factQ-1	How much does the typical 1 Terabyte hard drive cost today?
factQ-2	How many different tags are there on Notepad Web Page Designing (i.e. <TEXT>)?
factQ-3	What is the Range of WiMAX?
factQ-4	How many commands we can use in Run..(like ping,msconfig etc.)?
factQ-5	Does anyone know how many channels the new latest macbook's sound card has?
factQ-6	What's the average lifespan of a laptop computer before it breaks down?
factQ-7	How many cents per gigabyte is hard drive space now?
factQ-8	How many cents per gigabyte is RAM now?
factQ-9	What type of connector is used with CAT-5 cable?
factQ-10	What is the purpose of cladding in optical fibre?
factQ-11	What's the latest invented computer?
factQ-12	What is the maximum amount of data a Blu-Ray Disc can have?
factQ-13	What is good external hard drive brand? not too expensive
factQ-14	What's the average time it takes to load a simple blog on the internet?
factQ-15	What processor is coming after the Intel 2 Extreme?
factQ-16	What is the purpose of Service Pack 2 and 3 for?
factQ-17	How long does hp deskjet ink last for?
factQ-18	Where do I find my Computer Processing Unit(CPU) on my computer?
factQ-19	How close can speakers (magnets) get to a hard-drive without risking the loss of data?

A.2 Annotated Evaluation Data

I have manually annotated some of the example queries to evaluate the system's performance. The following table shows the number of possibly and definitely relevant sentences and the number of topics that appeared in a document collection. Additionally, the total number of documents, paragraphs and sentences within the collection is given.

Query ID	*Documents*	*Paragraphs*	*Sentences*	*definitely relevant*	*possibly relevant*	*Topics*
`defQ-0`	60	458	1401	58	57	5
`defQ-2`	60	198	694	5	37	3
`defQ-3`	25	196	565	4	10	3
`defQ-4`	60	384	2430	20	19	2
`defQ-10`	42	150	457	32	63	2
`defQ-13`	60	320	1152	10	10	3
`defQ-14`	60	312	977	22	49	3
`defQ-19`	60	313	1134	8	13	4

Appendix B

UIMA Components

The following is a list of UIMA components that have been implemented for the question answering system. Besides the package and class name of the component, the input and output annotations are given to allow further usage of the individual components.

— PREPROCESSOR —

annotator.UKPSentenceSplitter: Creates an annotation for each sentence in the document. The JAVA `BreakIterator` is used for the segmentation. (DKPro component.)	**Input:** — **Output:** Sentence
annotator.UKPSentenceTokenizer: Creates an annotation for each word in the document. The JAVA `BreakIterator` is used for the segmentation. (DKPro component.)	**Input:** Sentence **Output:** Token
annotator.POSLemmaAnnotator: Annotator for part-of-speech tags and word lemmas. The component uses an adapter to Helmut Schmid's TreeTagger, that determines the POS and the lemma.	**Input:** Token **Output:** Lemma, POS
annotator.StopWordAnnotator: Checks if a token or lemma is contained in the stop word lists and creates a corresponding StopWord annotation. (DKPro component.)	**Input:** Token, Lemma **Output:** StopWord
annotator.XTokenAnnotator: Combines token, part-of-speech and lemma annotations to a new XToken annotation that allows a faster access to these features. XTokens are only created for non-stop words.)	**Input:** Token, Lemma, POS, StopWord **Output:** XToken

annotator.TextSegmentationAnnotator: Adds annotations for paragraphs and splits sentence annotations that exceed paragraph boundaries or contain a large amount of consecutive white spaces. Sentences of length 1 are also removed.	**Input:** Sentence **Output:** Paragraph, Sentence
annotator.SpecificityAnnotator: Annotates an IDF-like score to each XToken. The word frequencies $freq(w, C)$ in the Leipzig Wortschatz corpus are used to calculate $spec(w) = 1 - \log freq(w, C)/\log\|C\|$.	**Input:** XToken **Output:** XToken
annotator.WordNetSenseAnnotator: Implementation of Lesk's word sense disambiguation algorithm that compares the sense gloss of WordNet with every token in the document and returns the appropriate word sense (i.e. the most probable) for every token. The sense id, the gloss and the number of available senses will be stored as annotation features.	**Input:** XToken **Output:** WordNetSense, XToken
annotator.QueryViewGenerator: Creates a view for the query document and duplicates each annotation that has been added so far.	**Input:** Query **Output:** View::Query

— PASSAGE EXTRACTION —

annotator.DuplicateDocumentRemover: Calculates an MD5 hash for each document and adds a CandidateDocument annotation with `relevant = false` if the same hash has already been seen for another document.	**Input:** RetrievedAnswer **Output:** CandidateDocument
annotator.ParagraphFilterAnnotator: Computes the semantic similarity between each paragraph and the query. The component uses cosine similarity and the methods defined by Leacock/Chodorow and Resnik. A CandidateParagraph annotation is created for each processed paragraph; the feature `relevant` is set to `false` if the cosine similarity is below 0.065, the Leacock/Chodorow similarity is below 0.513 or the Resnik similarity is below 0.06.	**Input:** Query, Answer, Paragraph, XToken **Output:** CandidateParagraph

annotator.SentenceSelectorAnnotator: Computes the semantic similarity between each sentence and the query. The component uses AltaVista-based PMI and Wikipedia-based ESA. Only sentences within a relevant paragraph is processed (an annotation of type CandidateParagraph with feature `relevant` needs to be present). A CandidateSentence annotation is created for each processed sentence; the feature `relevant` is set to `true` if the PMI score is above 0.15 and the ESA score is above 0.017.	**Input:** Query, Answer, Sentence, XToken, CandidateParagraph **Output:** CandidateSentence
annotator.PassageExtentDeterminationAnnotator: Finds natural borders between relevant and irrelevant passages by computing an optimal relevance distribution for each paragraph using a pre-trained Hidden Markov Model. PMI and ESA score as calculated in the previous component are used as emission probability.	**Input:** Answer, CandidateSentence **Output:** CandidateSentence

— TOPIC CLUSTERING —

annotator.KMeansTopicClusteringAnnotator: Calculates a k-means clustering for the given input documents. The number of clusters k can be specified. An annotation is created for each document. Empty clusters will not be annotated.	**Input:** CandidateDocument, XToken **Output:** KMeansTopicCluster
annotator.NewmanTopicClusteringAnnotator: Calculates a community clustering for the given input documents. To achieve that, a graph representation is created from the documents. The clustering is then optimized for a good modularity score, according to Newman's definition. The component uses the LinLogLayout implementation, which performs a greedy optimization. An annotation is created for each cluster. The cut-off threshold (i.e. the minimal similarity for adding an edge to the graph) can be specified; cosine similarity is used to weigh the edges.	**Input:** CandidateDocument, XToken **Output:** NewmanTopicCluster
annotator.MergedTopicClusteringAnnotator: Merges both the input cluster annotations to a new merged cluster by calculating the cross product of the two clusterings. Although the possible number of clusters is the product of both the input clustering sizes, most of the clusters will be empty. The new cluster id is a concatenated string of the original cluster ids.	**Input:** KMeansTopicCluster, NewmanTopicCluster **Output:** MergedTopicCluster

annotator.TopicLabelAnnotator: Iterates through all topic clusters of the given type and determines a label for the cluster. A bag-of-words index is first built from all documents within the cluster. The tokens are weighted by their specificity value to find rare words that appear often within the cluster. The three words with the highest score are finally used as topic label, separated by commas.	**Input:** MergedTopicCluster, XToken **Output:** MergedTopicCluster

— SUMMARIZATION —

annotator.DuplicateSentenceRemover: Calculates an MD5 hash for each sentence and changes the CandidateSentence annotation to `relevant = false` if the same hash has already been seen for another sentence.	**Input:** RetrievedAnswer, CandidateSentence **Output:** CandidateSentence
annotator.InvalidSentenceRemover: Identifies sentences with either no noun or no verb and changes their CandidateSentence annotation to `relevant = false`.	**Input:** RetrievedAnswer, Token, POS, CandidateSentence **Output:** CandidateSentence
annotator.DocumentPositionRankAnnotator: Ranks the sentences according to their occurrence position in the document. Sentences in the beginning of a document receive higher ranks than sentences in the middle or end.	**Input:** Answer, Sentence, CandidateSentence **Output:** DocumentPosition-RankedPassage
annotator.SimilarityRankAnnotator: Ranks the sentences according to their HMM emission probability that has been computed during passage extent determination. This similarity score consists of a scaled PMI and ESA value. High similarity scores obtain a high rank.	**Input:** Answer, Sentence, CandidateSentence **Output:** SimilarityRankedPassage
annotator.BiasedLexRankAnnotator: Calculates the Biased LexRank score for each sentence in the document collection and ranks them in descending order.	**Input:** Answer, Sentence, CandidateSentence **Output:** BiasedLexRank-RankedPassage

annotator.MergedRankAnnotator: Calculates the mean of the previous three rankings and reranks the sentences according to this score. The result is thus a combined ranking of *Position Rank*, *Similarity Rank* and *Biased LexRank*.	**Input:** DocumentPosition-RankedPassage, Similarity-RankedPassage, BiasedLexRank-RankedPassage **Output:** MergedRankedPassage
annotator.URLAnnotator: Uses a regular expression to find uniform resource locators (URLs) within the documents and annotates each occurrence. The underlying token, lemma and part-of-speech are removed and replaced by a new Token/XToken that spans the whole URL.	**Input:** Token, POS, Lemma, XToken **Output:** URL, Token, XToken
annotator.FrequentNGramAnnotator: Identifies n-grams in the document collection and adds an annotation for those with high scores. The n-grams are scored with their number of occurrences and their mean specificity value. Only n-grams of size 2–5 with a mean specificity of at least 1.0 and a minimum occurrence of 3 are considered.	**Input:** Paragraph, Token, XToken **Output:** FrequentNGram
consumer.SummaryHtmlFormatter: Generates the HTML excerpt that is displayed in the graphical user interface. A text, list and link view is created from the ranked sentences, the frequent n-grams and the URL annotations. A control bar is also formatted that allows a selection of topics or data sources.	**Input:** RetrievedAnswer, Sentence, XToken, MergedTopicCluster, CandidateSentence, RankedPassage, FrequentNGram, URL **Output:** —

Bibliography

R. A. Baeza-Yates. Introduction to data structures and algorithms related to information retrieval. In W. B. Frakes and R. A. Baeza-Yates (Eds.), *Information Retrieval: Data Structures and Algorithms*, pages 13–27. Upper Saddle River NJ, Prentice-Hall, Inc., 1992. ISBN 0-13-463837-9. 43

A.-L. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286 (5439):509–512, October 1999. ISSN 0036-8075. 44

L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In O. Shisha (Ed.), *Inequalities III: Proceedings of the Third Symposium on Inequalities*, pages 1–8. New York NY, Academic Press, September 1972. 31

L. E. Baum and J. A. Eagon. An Inequality with Applications to Statistical Estimation for Probabilistic Functions of a Markov Process and to a Model for Ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363, May 1967. 31

L. E. Baum and T. Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, December 1966. 31

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, 41(1):164–171, February 1970. 31

K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public Web search engines. *Computer Networks and ISDN Systems*, 30(1–7):379–388, April 1998. ISSN 0169-7552. 1

S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, April 1998. ISSN 0169-7552. 58

K.-U. Carstensen, C. Ebert, C. Endriss, S. Jekat, R. Klabunde, and H. Langer (Eds.). *Computerlinguistik und Sprachtechnologie. Eine Einführung*. Heidelberg, Spektrum-Verlag, 2nd edition, March 2004. ISBN 3-8274-1407-5. 15

T. Chklovski and P. Pantel. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In D. Lin and D. Wu (Eds.), *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '04)*, pages 33–40. Morristown NJ, Association for Computational Linguistics, July 2004. 22

S.-L. Chuang and L.-F. Chien. A practical web-based approach to generating topic hierarchy for text segments. In *Proceedings of the thirteenth ACM conference on Information and knowledge management (CIKM '04)*, pages 127–136. New York NY, ACM Press, November 2004. ISBN 1-58113-874-1. 62

K. W. Church and P. Hanks. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics*, 16(1):22–29, March 1990. 21

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 0035-9246. 45

P. Erdős and A. Rényi. On Random Graphs. I. *Publicationes Mathematicae Debrecen*, 6: 290–297, 1959. 44

G. Erkan and D. R. Radev. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research (JAIR)*, 22:457–479, December 2004. 58

C. Fellbaum (Ed.). *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. Cambridge MA, MIT Press, May 1998. ISBN 0-262-06197-X. 19

L. C. Freeman. Centrality in Social Networks: Conceptual Clarification. *Social Networks*, 1 (3):215–239, 1978/79. 58

E. Gabrilovich and S. Markovitch. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of The 20th International Joint Conference on Artificial Intelligence (IJCAI)*, January 2007. 22, 28

O. Gospodnetić and E. Hatcher. *Lucene in Action*. Greenwich CT, Manning Publications Co., 2004. ISBN 1-932394-28-1. 6

A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web (WWW '05)*, pages 902–903. New York NY, ACM Press, May 2005. ISBN 1-59593-051-5. 1

D. He, D. Demner-Fushman, D. W. Oard, D. Karakos, and S. Khudanpur. Improving Passage Retrieval Using Interactive Elicition and Statistical Modeling. In E. M. Voorhees and L. P. Buckland (Eds.), *Proceedings of the 13th Text Retrieval Conference (TREC)*. Gaithersburg MD, National Institute of Standards and Technology (NIST), November 2004. 31, 32, 78

M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. Measuring index quality using random walks on the Web. *Computer Networks*, 31(11–16):1291–1303, May 1999. ISSN 1389-1286. 58

E. J. Horvitz, D. R. Azari, S. T. Dumais, and E. D. Brill. Cost-benefit approach to automatically composing answers to questions by extracting information from large unstructured corpora, United States Patent 7,454,393, November 18, 2008 (filed August 6, 2003). 14

A. Islam and D. Inkpen. Semantic Similarity of Short Texts. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP '07)*, September 2007. 18

A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys*, 31(3):264–323, September 1999. ISSN 0360-0300. 42, 43

B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing & Management*, 36(2): 207–227, March 2000. 1, 57

J. J. Jiang and D. W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics (ROCLING X)*, August 1997. 21

V. Jijkoun and M. de Rijke. Retrieving answers from frequently asked questions pages on the web. In O. Herzog, H.-J. Schek, N. Fuhr, A. Chowdhury, and W. Teiken (Eds.), *Proceedings of the International Conference on Information and Knowledge Management (CIKM '05)*, pages 76–83. New York NY, ACM Press, November 2005. ISBN 1-59593-140-6. 6

M. Kaisser, M. A. Hearst, and J. B. Lowe. Improving Search Results Quality by Customizing Summary Lengths. In *Proceedings of ACL-08: Human Language Technology Conference (HLT)*, pages 701–709. Morristown NJ, Association for Computational Linguistics, June 2008. 1, 63

S. Kamvar, T. Haveliwaland, and G. Golub. Adaptive methods for the computation of PageRank. *Linear Algebra and its Applications*, 386(Special Issue on the Conference on the Numerical Solution of Markov Chains 2003):51–65, July 2004. 59

B. Katz. Annotating the World Wide Web using natural language. In *Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet (RIAO '97)*, June 1997. 12, 70

B. Katz, S. Felshin, D. Yuret, A. Ibrahim, J. Lin, G. Marton, A. J. McFarland, and B. Temelkuran. Omnibase: Uniform Access to Heterogeneous Data for Question

Answering. In *Natural Language Processing and Information Systems*, volume 2553 of *Lecture Notes in Computer Science*, pages 230–234. Berlin/Heidelberg, Springer, 2002. 12, 70

B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 62(317):86–101, March 1967. 43

R. Kohavi and F. Provost. Glossary of Terms. Editorial for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process. *Machine Learning*, 30(2–3): 271–274, February/March 1998. ISSN 0885-6125. 24

C. Leacock and M. Chodorow. Combining Local Context and WordNet Similarity for Word Sense Identification. In C. Fellbaum (Ed.), *WordNet: An Electronic Lexical Database*, pages 265–283. Cambridge MA, MIT Press, May 1998. ISBN 0-262-06197-X. 20

M. Lesk. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In V. DeBuys (Ed.), *Proceedings of the 5th annual international conference on Systems Documentation (SIGDOC '86)*, pages 24–26. New York NY, ACM Press, June 1986. ISBN 0-89791-224-1. 17

C.-Y. Lin. Looking for a Few Good Metrics: Automatic Summarization Evaluation — How Many Samples Are Enough? In *Proceedings of NTCIR Workshop 4*, June 2004a. 69

C.-Y. Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In M.-F. Moens and S. Szpakowicz (Eds.), *Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL '04*. Morristown NJ, Association for Computational Linguistics, July 2004b. 69, 70

D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning (ICML)*, pages 296–304. San Francisco CA, Morgan Kaufmann, July 1998. 21

J. Lin. An Exploration of the Principles Underlying Redundancy-Based Factoid Question Answering. *ACM Transactions on Information Systems (TOIS)*, 25(6):6, 1–55, April 2007. ISSN 1046-8188. 64

Y. Liu, S. Li, Y. Cao, C.-Y. Lin, D. Han, and Y. Yu. Understanding and Summarizing Answers in Community-Based Question Answering Services. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING '08)*, pages 497–504, August 2008. 7, 10, 63

I. Mani. *Automatic summarization*. Natural language processing, Vol. 3. Amsterdam/Philadelphia, John Benjamins Pub. Co., 2001. ISBN 1-58811-059-1. 63

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. New York NY, Cambridge University Press, 2008. ISBN 978-0-521-86571-5. 18, 42, 43, 46, 60

H. Masuichi, D. Sugihara, T. Ohkuma, and H. Yoshimura. Question answering system, data search method, and computer program, United States Patent 7,461,047, December 2, 2008 (filed September 22, 2005). 13

R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and Knowledge-based Measures of Text Semantic Similarity. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI '06)*, pages 775–780. Menlo Park CA, AAAI Press, July 2006. 16, 18, 19, 20, 21, 23, 28

C. Müller, T. Zesch, M.-C. Müller, D. Bernhard, K. Ignatova, I. Gurevych, and M. Mühlhäuser. Flexible UIMA Components for Information Retrieval Research. In *Proceedings of the LREC 2008 Workshop 'Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP'*, pages 24–27, May 2008. 16, 22

E. Möller. *Die heimliche Medienrevolution – Wie Weblogs, Wikis und freie Software die Welt verändern*. Hannover, Heise Zeitschriften Verlag GmbH & Co KG, November 2004. ISBN 3-936931-16-X. 1

M. Murata. Question answering system and question answering processing method, United States Patent 7,444,279, October 28, 2008 (filed June 3, 2004). 13

G. Nagy. State of the art in pattern recognition. *Proceedings of the IEEE*, 56(5):836–863, May 1968. 43

M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, June 2004a. 44

M. E. J. Newman. Analysis of weighted networks. *Physical Review E*, 70(5):056131, November 2004b. 45

M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, September 2006a. 44

M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences (PNAS)*, 103(23):8577–8582, June 2006b. 44

M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, February 2004. 44

J. Otterbacher, G. Erkan, and D. R. Radev. Biased LexRank: Passage Retrieval using Random Walks with Question-Based Priors. *Information Processing & Management*, 45 (1):42–54, January 2009. ISSN 0306-4573. 58, 59, 60, 78

U. Quasthoff, M. Richter, and C. Biemann. Corpus Portal for Search in Monolingual Corpora. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC '06)*, pages 1799–1802, May 2006. 16, 20

M. R. Quillian. Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities. *Behavioral Science*, 12(5):410–430, September 1967. ISSN 0005-7940. 20

L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989. ISSN 0018-9219. 31

D. R. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Çelebi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, J. Otterbacher, H. Qi, H. Saggion, S. Teufel, M. Topper, A. Winkel, and Z. Zhang. MEAD – a platform for multidocument multilingual text summarization. In *Proceedings of the 4th International Conference On Language Resources And Evaluation (LREC '04)*, May 2004. 70

P. Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 448–453, 1995. 20, 21

R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321 (Informational), April 1992. 24

S. M. Ross. *A First Course in Probability*. New York NY, Macmillan, 1976. ISBN 0-02-403880-6. 20

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988. ISSN 0306-4573. 19

G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, November 1975. ISSN 0001-0782. 18

G. Salton, E. A. Fox, and H. Wu. Extended Boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, November 1983. ISSN 0001-0782. 16

H. Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of International Conference on New Methods in Language Processing*, September 1994. 16

M. Seki, K. Marukawa, and M. Takatsu. Question-answering method and question-answering apparatus, United States Patent 7,236,968, June 26, 2007 (filed January 29, 2004). 14

P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy. The Principles and Practice of Numerical Classification*. A Series of books in biology. San Francisco CA, W. H. Freeman & Co., June 1973. ISBN 0-7167-0697-0. 43

K. Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):164–165, 1972. ISSN 0022-0418. 16

J. Stefanowski and D. Weiss. Comprehensible and Accurate Cluster Labels in Text Clustering. In *Proceedings of the 8th Large-Scale Semantic Access to Content Conference (RIAO '07)*, June 2007. 62

H. Steinhaus. Sur la division des corp materiels en parties. *Bulletin L'Acadmie Polonaise des Science C1*, III, IV:801–804, 1956. 45

A. Strehl. *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. PhD thesis, The University of Texas at Austin, May 2002. 46

R. Torralbo, E. Alfonseca, A. Moreno-Sandoval, , and J. M. Guirao. Automatic generation of term definitions using multidocument summarisation from the Web. In *Proceedings of the RANLP '05 Workshop on Crossing Barriers in Text Summarization Research*, 2005. 63

P. D. Turney. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning (EMCL '01)*, pages 491–502. Berlin/Heidelberg, Springer, September 2001. ISBN 3-540-42536-5. 21

C. J. van Rijsbergen. Foundations of Evaluation. *Journal of Documentation*, 30(4): 365–373, December 1974. ISSN 0022-0418. 24

L. H. Vanderwende, A. A. Menezes, and M. L. Banko. Method and system for ranking words and concepts in a text using graph-based ranking, United States Patent 7,430,504, September 30, 2008 (filed April 15, 2004). 14

G. Vickery and S. Wunsch-Vincent. *Participative Web And User-Created Content: Web 2.0 Wikis and Social Networking*. Paris, OECD Publishing, October 2007. ISBN 978-92-64-03746-5. 1

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967. ISSN 0018-9448. 31

Z. Wu and M. Palmer. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, June 1994. 20

D. Yang and D. M. W. Powers. Verb Similarity on the Taxonomy of WordNet. In P. Sojka, K.-S. Choi, C. Fellbaum, and P. Vossen (Eds.), *Proceedings of the Third International WordNet Conference (GWC)*, pages 121–128, January 2006. ISBN 80-210-3915-9. 18, 19, 20

S. Ye, J.-R. Wen, and W.-Y. Ma. A Systematic Study of Parameter Correlations in Large Scale Duplicate Document Detection. In *Advances in Knowledge Discovery and Data Mining*, volume 3918 of *Lecture Notes in Computer Science*, pages 275–284. Berlin/Heidelberg, Springer, 2006. ISBN 978-3-540-33206-0. 24

H. Yoshimura, H. Masuichi, T. Ohkuma, and D. Sugihara. Question answering system, data search method, and computer program, United States Patent 7,418,443, August 26, 2008 (filed December 14, 2005). 13

Z. Zheng. AnswerBus question answering system. In *Proceedings of the second international conference on Human Language Technology Research (HLT '02)*, pages 399–404. San Francisco CA, Morgan Kaufmann Publishers Inc., March 2002. 12