Retrieving and Summarizing Educational Document Collections Using Machine Learning

Konstruktion und Zusammenfassung bildungsbezogener Dokumentensammlungen mittels Machine-Learning-Verfahren

Master-Thesis von Stefan Henß

15. April 2013

TECHNISCHE UNIVERSITÄT DARMSTADT

UBIQUITOUS KNOWLEDGE PROCESSING

Retrieving and Summarizing Educational Document Collections Using Machine Learning
Konstruktion und Zusammenfassung bildungsbezogener Dokumentensammlungen mittels Machine-Learning-Verfahren

vorgelegte Master-Thesis von Stefan Henß

Gutachten: Prof. Dr. Iryna Gurevych
Betreuung: Dr. Saeedeh Momtazi

Tag der Einreichung:

Erklärung zur Master-Thesis

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

Darmstadt, den 15. April 2013

(S. Henß)

Zusammenfassung

Webverzeichnisse sind zumeist hierarchisch strukturierte Internetseiten, welche aus Kategorien und Unterkategorien aufgebaut sind, die jeweils Verweise zu themenbezogenen Webseiten beinhalten. Ein solcher Aufbau erlaubt es den Besuchern ein Thema zu erschließen, indem man ihnen sowohl qualitative, jedoch inhaltlich breit angelegte Einstiegsinformationen bietet, als auch die nähere Erforschung von Spezialgebieten über Unterkategorien ermöglicht. Webverzeichnisse zu erstellen erfordert jedoch umfangreichen Arbeitsaufwand und Fachkenntnisse, um relevante sowie vertrauenswürdige Quellen aufzufinden und organisieren zu können. Weiterhin bieten die meisten Verzeichnisse eigens bereitgestellte Informationen wie Schlagwörter und Zusammenfassungen an, anhand derer die Leser besser für sie relevante Links zu identifizieren vermögen. Aus dem dazu nötigen Aufwand ergibt sich, dass Webverzeichnisse schlecht skalieren und in den letzten Jahren von Suchmaschinen als Primärmittel zur Erschließung des Webs verdrängt wurden. Das Webverzeichnis-Konzept hat jedoch insofern weiterhin Relevanz, als es im Unterschied zu Suchmaschinen, die auf Basis von speziellen Suchanfragen arbeiten, in der Regel keine Vorkenntnisse des Benutzers voraussetzt, um ihm insbesondere auch komplexe Themen in vollem Umfang darstellen zu können. Um Konstruktion sowie Erweiterung von Webverzeichnissen zu unterstützen, widmet sich diese Arbeit verschiedenen Mitteln des maschinellen Lernens um wichtige Arbeitsschritte zu automatisieren. Zunächst wird ein Web-Crawler konzipiert, mit dessen Hilfe themenrelevante Webseiten und PDF-Dokumente automatisch gefunden werden können. Darauf werden die ausgewählten Links automatisch sowohl mit Schlüsselbegriffen, als auch mit aus Beispielsätzen zusammengesetzten Zusammenfassungen beschrieben, um dem Benutzer einen bestmöglichen Überblick über das Informationsangebot zu verschafffen. Für die Umsetzungen dieser Aufgaben werden Methoden des Unsupervised Learning, Supervised Learning, sowie des bisher wenig in Natural-Language-Processing-Anwendungen erforschten Reinforcement Learning kombiniert. Wir vergleichen unsere Ergebnisse mit dem Stand der Technik, schlagen neue Bewertungsmethoden für unerforschte Aufgaben vor, und führen Benutzerstudien durch, die nahelegen, dass Machine-Learning-Methoden von signifikantem Nutzen für die Konstruktion von bildungsbezogenen Webverzeichnissen sein können.

Abstract

Web directories are Internet services that are typically structured as hierarchies of categories and subcategories, each containing links to Web pages offering information related to the respective topics. With such a format, it is possible to guide readers through complex matters by presenting them high-quality introductory resources as well as by allowing them to further explore specific topical aspects through subcategories. To publish Web directories, considerable effort and domain knowledge is required for finding and organizing both relevant and credible Web pages. Furthermore, most Web directories offer link annotations, such as keywords and summaries, that allow the user to better find Web resources matching their specific information needs. Consequentially, manually creating Web directories does not scale well and in the last decade, they have been replaced by search engines as the primary tool for browsing the Internet. Yet, with the typical search engines' drawback of requiring the user to formulate adequate queries, Web directories still are a relevant approach to organizing Web links as a means of introducing readers to complex topics, while not assuming any prior knowledge on their part. To aid the constructing of Web directories, as well as to allow them growing faster, this thesis explores machine learning methods for automating significant design steps. First, a Web crawler is designed for retrieving links to topic-specific Web pages and PDF documents. Second, keyphrases are assigned to each link, helping the reader to better anticipate its contents. Third, as a further insight into content and style, representative sentences are automatically selected and joined to form summaries of single resources. To accomplish those tasks, methods of three major statistical machine learning classes are employed; unsupervised learning, supervised learning, and, relatively new to natural language processing, reinforcement learning. We compare our results to state-of-the-art technology, suggest new quantitative performance setups and metrics where we break new grounds, and perform qualitative evaluation studies, leading to conclusive proof that constructing educational Web directories can significantly be supported by means of machine learning.

Contents

1 Introduction

1.1 Motivation

Since the early days of the World Wide Web in 1991, Web directories like the *Virtual Library*[1] and later, more prominently, the *Yahoo! Directory*[2] and the *Open Directory Project (DMOZ)*[3], have served its users as gateways to the evergrowing amount of sparsely connected Websites by offering organized, topic-specific link collections. In fact, the very first Web page ever to be published soon started to function as a directory for the early Web[4]. Back when search engines for hypertext Websites, as we know them today, were not yet introduced, but content started getting available on more and more public Web servers, apart from directly accessing Web pages through their URLs (*uniform resource locator*) or following direct links from other sites, Web directories were the only possible way to explore the Internet and to discover resources for specific topics that were made available across multiple servers. Typically structured as hierarchies of categories and sub-categories, so-called taxonomies, Web directories allowed their readers to find general, introductory resources, as well as to delve into specific subcategories — even if the user did not yet have a clear vision of what he was looking for and which information would be available. In distinction from even the earliest search indexes, however, Web directories have always been a predominately manual endeavor and therefore difficult to grow in size and to keep pace with the latest trends within the exponentially growing World Wide Web.

After the first powerful search engines, indexing large amounts of the Web, emerged in the mid and late '90s[5], Web directories began to lose their significance, as search engines covered far more Websites and allowed to retrieve suggestions based on specific keyword-based queries [BP98]; therefore nowadays serving as the general public's primary tool for exploring the Web. Nevertheless, still today, Web directories offer advantages over search engines, stemming from their concept, their structure, and the persistence of their contents. Search engines have to be able to respond to virtually any query, making it impossible to have each answer stored beforehand — especially if they were to be provided manually. As a consequence, search results usually only reflect simple measures of how well links match a query and how prominent they are within the Internet [PBMW99]; but they are merely capable of explicitly explaining whole concepts and putting them into perspective with related topics — a more intellectually challenging endeavor. Directories, on the other hand, are dedicated to specific topics, and have their structure and content fixed beforehand, thus allowing human interaction and topic-specific design decisions. This approach, while, compared to search engines, being limited in its search capacities and scalability, allows human authors to design topical hierarchies and link collections specifically for their target audience [She00]. Furthermore, it allows them to better guarantee the validity and credibility of their linked resources than with results retrieved on-the-fly. Those three aspects, high control over quality standards, catering to specific target audience needs, and the possibility of guiding the reader through complex topics by means of subpages instead of merely offering a list of Websites matching some keywords, make online directories distinct from nowadays search engines and therefore, in our opinion, still a relevant format for supporting the access to the Internet.

[1] `http://vlib.org/`
[2] `http://dir.yahoo.com/`
[3] `http://www.dmoz.org/`
[4] `http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html`
[5] '93's `http://www.aliweb.com/` being the first and full text search available since '94's `http://www.webcrawler.com/`

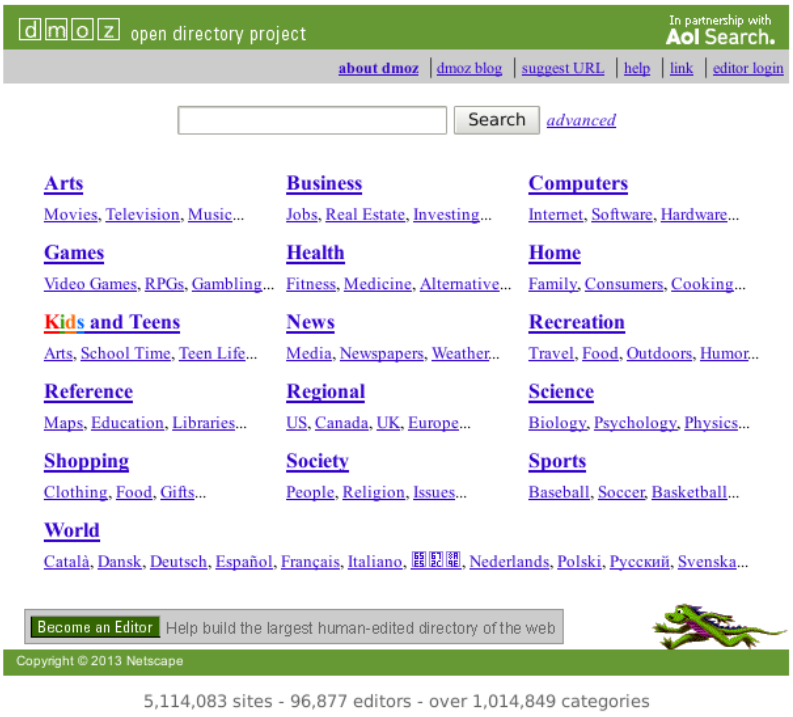

Figure 1.1.: Front Page of dmoz.org, the Largest Open Web Directory.

Constructing a Web directory typically consists of three major steps. First, the general focus of the directory has to be determined, as well as its structure, which, for instance, typically includes a hierarchy of topics and sub-topics — its taxonomy. Second, for each category, a well-arranged selection of the most relevant online resources, i.e., high-quality Websites covering all important aspects of a topic, has to be established. Finally, these topical categories and their selected Web resources have to be presented in a way which allows the user to navigate the contents and to find the information he is looking for as fast as possible. Such a presentation can, for instance, include keywords, quality ratings, images, snippets, or summaries. To achieve high quality standards, which, as mentioned, is key to Web directories' significance, a considerable amount of effort and domain knowledge has to be put into each step. Not only is profound domain knowledge required in order to be able to structure a topic and to identify the best resources for the target audience. But also much time has to be invested into searching content, reading it, and presenting it to the reader, e.g., by writing short summaries about each resource. In a way, designing a Web directory's categories can be compared to the intellectual challenges of structuring books or writing encyclopedic articles, such as for Wikipedia. As a result, this approach lacks scalability; for example, the largest general directory, *Dmoz*, only links to about 5 million distinct URLs, while search engines like Google or Yahoo typically index well over 10 billion Web pages[6].

In order to combine the clear structure and design benefits of Web directories with the scalability of search engines, the need for expensive expert knowledge and considerable time investment for creating directories and keeping them up-to-date has to be overcome. With advanced indexing techniques [BP98] and network analysis algorithms like PageRank [PBMW99], search engines are able to automatically match candidate Websites to queries, to rate their relevance, and to judge their credibilities; for example, through the way other presumably credible Websites link to them. Web directories, in turn, for their construction much more depend on an actual human understanding of semantics for properly searching, selecting, and categorizing Web resources, and, especially, for providing quality meta information such as

[6] `http://www.worldwidewebsize.com/`

keyphrases and summaries. In this thesis, we thus seek to explore ways of approximating such intelligent behavior in constructing Web directories by means of statistical machine learning, as outlined in more detail in Section 1.2. As our primary example for the learning and evaluation process, we will consider the DIPF Eduserver, a large directory of primarily German educational resources, which currently hosts over 27,000 links. Finally, in Section 1.3, we will give a brief overview about how the remainder of this thesis is structured.

1.2 Work Hypothesis

For us to be able to support the construction process of Web directories, or to even automate it as a whole, the general design steps a human author follows have to be modeled and have to be approximated as good as possible. Gathering collections of topic-specific text documents, especially for constructing Web directories, typically involves a process of multiple subsequent and interdependent steps. As already discussed in the previous section, first, the general approach towards structure and content of the collection has to be decided on. Second, for each considered topic or sub-topic, relevant documents have to be retrieved from the available resources, e.g., the World Wide Web. In this process, it should also be made sure that the whole range of a topic is covered, while redundancies are minimized. Finally, having obtained a collection of promising documents, each has to be analyzed in depth, making it possible to assure its quality and to further assign meta information such as keywords and a short summary.

Additional to the high-level process, the single aforementioned steps themselves involve processes of subsequent decision making. For instance, efficiently searching vast resources for relevant content — like the Internet, where visiting every Web page is practically impossible — requires to learn search strategies from past decisions and to make subsequent predictions about whether following a link directly or indirectly leads to the desired information or not. Furthermore, analyzing, annotating, and summarizing the obtained documents involves decisions to be made and their effects on further options to be considered. Concentrating the essence of whole topics into subsets of documents, as well as writing abstracts or assigning keyphrases for resources, does not only require independent estimates of single information's importance. One also has to consider that, for example, pieces of knowledge carrying similar information will become redundant, so that every decision has implications to be considered for optimal results. In the most self-evident example, writing a summary, the content and grammar of each sentence directly narrows down possible choices for the next sentences.

In the past, different approaches to automating Web directory construction have been undertaken. Research specifically motivated by Web directories so far typically focused on means of organizing new Web pages into a given topic hierarchy / taxonomy. For populating and growing Web directories using examples, various machine learning approaches have been used for building models of categories which can then be utilized to estimate the appropriate categories for new Web pages [DC00] [CD00] [GTL+02]. In another, more recent approach, algorithms are developed which do not solely rely on extensive training examples, but leverage more abstract information of a given taxonomy than example entries for classifying documents [AAS03] [AAS05] [HCC04] [SKK+05] [SNK+06]. Furthermore, disconnected from an explicit attempt of constructing Web directories, the single tasks of focused crawling, link clustering and classification, keyphrase assignment, and summarization, have also been extensively studied in isolation or coupled with similar problems, as we will discuss in the respective chapters.

Problem formulations specific to or applicable for the creation of Web directories typically have been solved using the machine learning techniques of supervised and unsupervised learning. Supervised learning considers a set of training examples, each having at least one "label", which is often manually assigned to the data and shall be learned to be predicted in the future, when only data of a similar format is given, but not the target labels. Applications in the field of text organization and summarization typ-

ically consider training data consisting of either keywords, keyphrases, sentences, or entire documents; and then construct predictors for a relevance indicator, which is assigned to the training examples. Unsupervised learning, or cluster analysis, does not have a desired solution at hand during training, but tries to uncover the structures beneath a data set, such as clusters of related data or latent features for dimension reduction, based on predefined models which are fit to the data. Different from the supervised learning approach, organizing texts or extracting keywords and summaries using unsupervised learning is typically achieved through the identification of intrinsic structures. Those intrinsic structures typically are patterns such as groups or networks, and, resulting from such a representation, those elements which appear to be characteristic or "central" are selected. The problem with both approaches is that within their typical frameworks and most general solutions (their models), neither time, nor decision processes are explicitly considered, which we, however, identified as crucial to the Web directory design process.

Parallel to the aforementioned techniques, another major approach to statistical machine learning has evolved during the past two decades. In *reinforcement learning*, decisions are not observed in isolation, but considered as part of a process, for which the ultimate goal is to maximize the reward over time [SB98]. In this framework, the performance of an algorithm typically does not just refer to the accumulated error of independent decisions, as standard supervised learning applications typically do, or to a model likelihood as in unsupervised learning. Instead, it refers to the quality of the actions an algorithm takes and which implications on the environment and its future they have. When crawling the Internet, for instance, following the right links is crucial when facing constraints, for example, in time. Therefore, for ensuring scalability of Web crawling, it is important to not only optimize the relevance of selected resources, but also the rate with which relevant findings are made. An optimal search strategy hereby also explicitly involves the trade-off between an early, or "greedy", detection of relevant content, which also may lead to dead ends, and the long-term expectations, or, in other terms, approaching the "global optimum" in the space of possible paths through the World Wide Web. In another example, the automated summarization of texts, there may not exist immediate rewards at all, so that each decision has to be made with respect to the estimated utility of the final product, which may, for example, be the reader's satisfaction with the summary as a whole.

Utilizing reinforcement learning techniques, as well as recent advancements in supervised and unsupervised statistical machine learning, in this thesis we will seek to give an up-to-date answer for the question of how far the creation process of Web directories can be automated. Specifically, the hypothesis we seek to follow is that by taking manually created Web directories as training examples, statistical machine learning algorithms can be used to approximate typical design processes, leading to comparable results — both quantitatively and qualitatively. The specific aspects of the entire design process we will research in this thesis are topical Web crawling, keyphrase extraction and assignment, and automated summarization, which, with the exception of automatically finding a topical hierarchy, comprises all important steps of construction Web directories.

As our primary reference for trying to prove this hypothesis, we will consider the DIPF Eduserver, a large and active German Web directory of educational resources. Started in 1996, the Eduserver (*Deutscher Bildungsserver*)[7] offers a large link collection of high-quality content regarding manifold aspects of education and pedagogy. Different from open directories such as the DMOZ, the Eduserver is solely curated by scientifically educated experts of their respective fields, working at the *Deutsche Institut für Internationale Pädagogische Forschung* (DIPF)[8] [Kü12]. Additionally to databases for institutions, events, and jobs, the Eduserver also maintains a manually assembled collection of over 27,000 links to

[7] http://www.bildungsserver.de/

[8] http://www.dipf.de/

Websites and PDF documents, organized in a taxonomy of more than 700 categories, with a hierarchical depth of up to 9 levels — the data we will use for this thesis.

Using the Eduserver as our main reference, the contributions of this thesis will not only be design proposals and quantitative evaluations on generic datasets for each of the three stipulated tasks. Furthermore, we will also present the results on this specific dataset for all tasks — along with a discussion of potentially problematic characteristics which more artificial datasets often lack. Finally, we will also conduct qualitative user studies for our results, in which authors of the DIPF Eduserver will provide their feedback regarding the quality of our findings, from which we can then derive a holistic answer to our work hypothesis. To our knowledge, such dedicated research specifically targeted at constructing educational Web directories has not yet been published.

1.3 Thesis Outline

For the remainder of this thesis, we will follow the structure below.

First, in Chapter 2, a general introduction to statistical machine learning methods in natural language processing is given as the basis for the reader to understand all technical concepts used in this thesis. In Section 2.2, different ways of preparing and representing textual documents in formats suitable for statistical analysis are presented. In sections 2.3 and 2.4, general introductions into the concepts of unsupervised and supervised learning, respectively, are given. Finally, in Section 2.5, reinforcement learning is introduced — a machine learning approach which so far has not found broad application in natural language processing, but appears promising for most of the tasks involved in creating Web directories.

Second, in Chapter 3, general linguistic measures are introduced, as well as modeling techniques, which will be relevant for more than one of our specific tasks. Those include measures for term significance in Section 3.2, the introduction of lexical-semantic resources in Section 3.3, and semantic similarity measures in Section 3.4.

Third, in Chapter 4, our first specific task, topical Web crawling will be introduced. Specifically, we will design a system which, given reference data, will crawl the Web for resources concerning similar topics. Our results will be evaluated both quantitatively, by trying to rebuild and to extend categories from the Eduserver, and qualitatively, through a user study.

Then, in Chapters 5 and 6, we will describe our approaches to keyphrase extraction and single-document summarization. Again, our systems will be evaluated qualitatively and quantitatively — also in comparison with competitive baselines approaches on additional datasets.

Finally, in Chapter 7, we will conclude this thesis with a reflection on our findings and an outlook on possible extensions of our work.

2 Machine Learning in Natural Language Processing

2.1 Introduction

Machine learning refers to a class of algorithms targeted at automatically analyzing data, recognizing patterns, and learning to make predictions or to mimic behavior from it. Essential to all machine learning frameworks is the ability to derive complex information or behavior which was not hard-coded into the algorithm or given in a database, but directly learned from data. In the most frequent scenario, this means identifying "hidden" relations between variables, and, for instance in the case of supervised learning, trying to map observations to expected outcomes. For example, in credit risk rating, often concatenations of functions are used, so-called neural networks, which feature two kinds of variables: those which represent features of the debtor, such as his age, and coefficients which then are automatically tuned so that for training examples, the resulting predictions of the network (the resulting function values) are as close to the real outcomes (credit default or not) as possible [HCH+04]. This specific framework of analyzing quantitative aspects of data, such as the correlation of one factor with a response variable, is also called statistical machine learning, as opposed to other branches of artificial intelligence where not statistics, but encoded relations and constraints are used to deduce new knowledge by logic.

In natural language processing, machine learning is often used to organize and categorize text, to obtain sparse representations of text, and to learn making analytical decisions for rating or annotating text. For example, in word-sense disambiguation, all words within training texts typically are annotated by a human and then an algorithm derives rules which allow to estimate correct word sense assignments for future texts, solely based on features of the words themselves and their context [Nav09]. In the example of quality assessment, much like in credit risk rating, quality ratings for training data are used to identify global features of text, such as its length, as indicators for its quality [SBHZ10].

To be able to computationally analyze texts, a suitable representation of texts and their elements, such as single words or sentences, is required. Section 2.2 will introduce various measures which will be used in this thesis to obtain and preprocess text from Websites and PDF documents. In Sections 2.3 and 2.4, we will introduce general approaches to supervised learning (i.e., using examples of expected outcomes for training) and unsupervised learning (used to identify intrinsic structures based on predefined models, without examples of "correct" results), respectively. Finally, in Section 2.5, we will introduce reinforcement learning, which considers chains of decision making, but so far have not been explored much for natural language processing.

2.2 Text Representations

For analyzing textual data, it has to be represented in a machine-readable format that is suitable for conveying all essential information, and that matches the input format of the later applied algorithms. In natural language processing techniques dealing with content and semantics, it is often sufficient to provide an explicit separation of sentences, words, and numbers. The latter two will also be called "tokens" in this thesis to express the atomic role of words and numbers in most analysis; however, it should not be confused with a similar use of the term in other publications, where it also subsumes punctuation and other syntactical features, which, due to our focus on content, generally is not of interest

to us. Still, there are applications, like sentence-structure parsing, where also punctuation is necessary to be represented, since it carries necessary syntactical and grammatical information.

In the vast majority of the algorithms presented in this work, it is only necessary to distinguish between sentences and tokens, and to know their original order, which means that many characters from the original texts can be omitted. For the summarization task, however, the original text should still be stored on the sentence-level, typically as UTF-8 strings, to be able to present the results in their most suitable form — including punctuation and special characters that are neither part of words, nor numbers. Since the main objective of this work is to analyze and utilize HTML Websites and documents in other formats that are typically used on the Internet, such as PDFs, several steps are required even before segmenting text, since it usually does not come in a clean, plain string form.

In sections 2.2.1 and 2.2.2, we will describe the preprocessing techniques applied to the typical input formats of this work — Web pages and PDF documents. In Section 2.2.3, further text-processing techniques are described that work on plain standard text representations and therefore are applicable to most natural language processing tasks. Finally, in Section 2.2.4, we will present various steps of further filtering and manipulating texts in order to reduce linguistic variance in favor of terms primarily and non-ambiguously representing concepts and ideas.

2.2.1 Web Pages

Typical modern websites use a variety of stylistics means to present their content. Such means include images, videos, scripts — which may load content only on request —, and the distribution of text across blocks/divisions, embedded websites, and multiple pages. Furthermore, most text that is specific for a single Web page does not stand alone, but is surrounded by re-occurring elements such as the navigation, and social media functions of the so called “Web 2.0”, such as commenting sections. While most of such devices enhance the user experience, it becomes increasingly difficult to automatically distill plain text which represents the content of a web page.

The tasks of crawling Websites and assigning keyphrases to them generally only require term frequency information to analyze the contents, which means that is generally possible to utilize any textual content; be it the text itself, navigation items, or HTML meta information such as the document title. Only in some cases it is necessary to distinguish between the main text and the so-called boilerplate contents of a Web page. For instance, in measuring content similarity, large amounts of unspecific boilerplate text may lead to overestimates of similarity, since most of the common text may not be part of the actual content. For summarization, however, it is important that only proper sentences of the actual content are extracted, since mis-interpreting listings as sentences or proposing boilerplate sentences as parts of the summaries would seriously affect the usability of the produced results.

As a consequence, our preprocessing system has to provide multiple representations of a single Web page: meta information such as title and keywords, all visible text as one textual unit, and all visible text as another unit, after having removed boilerplate parts. To obtain a Web page’s content, we first have to download its HTML code, for which we use *HtmlUnit*[1], a Java library which is primarily developed as a testing framework for Web content, but in it offers a feature-rich web browser API, which can also handle most JavaScript. HtmlUnit is also used to download any other Web content, including PDF documents, and the downloaded material is stored to disk so that during development, the Websites do not have to be accessed again each time a change to the preprocessing algorithms occur. Additional to downloading the plain HTML markup, which is presented to any Web browser accessing a website, HtmlUnit also parses the HTML to allow iterating its elements. We, however, only use the textual/XML representation

[1] `http://htmlunit.sourceforge.net/`

of the HTML code and parse it using *jsoup*[2], which offers a little more convenient information extraction methods to access meta data such as keywords. Finally, for obtaining the text of a page, we apply *boilerpipe*[3]. Boilerpipe is originally designed to use HTML features in conjunction with supervised learning methods to automatically distinguish between relevant and irrelevant text blocks — specifically for extracting news articles from Web pages [KFN10]. We, however, found out that for many types of Web content which are relevant for this thesis, the models provided with boilerpipe do not fit well and are too restrictive. As boilerpipe's models try to identify large blocks of consistent text, it typically ignores information that is only expressed through a few words but yet potentially relevant for content analysis. Still, boilerpipe is useful to us for efficiently segmenting text in Web pages that is separated by the various layout options available to HTML and CSS — without filtering any content. From this segmentation, we can then define "sentences" as logical units which will keep the separated text segments apart for further steps. In contrast, HtmlUnit and jsoup only return simple strings of visible text, which is concatenated so that, for example, items from a navigation menu appear to form a proper common sentence, which, especially with respect to summarization, could be a dangerous threat to the quality of our final results.

Due to the lack of efficient openly available boilerplate-removal tools, we had to implement one on our own. To do so, we followed our intuition that relevant content could best be extracted from a Web page by separating it from elements that are typically found as well on other pages of the same site. Therefore, for each Website, we derive a sentence filter by collecting text statistics from all subpages of this Website that are linked from its front page. In particular, we store any sentence — recall that as a result of boilerpipe's segmentation, even a single word from a navigation menu might constitute a "sentence" — that can be found in at least 20 more pages of the Website. Those sentences are then removed from any subpage of this site for which we want to only obtain the specific, characteristic textual contents.

2.2.2 PDF Documents

As mentioned in the last section, downloading PDF documents can be achieved by utilizing the same libraries as for obtaining HTML code. PDF documents, however, have their contents described in a different format, which is more oriented towards laying out graphics than just describing textual content. While HTML, without the use of stylesheets, primarily provides markup to structure the text into headings, paragraphs, etc., which are then displayed in order, text in PDF documents is only described as a one kind of graphical element with absolute coordinates and further attributes such as its font. Having manually "programmed" documents, i.e. PDF documents directly described in a markup language, this would not mean much of a problem, since, due to a presumably straight-forwards structure, most of the text could directly be extracted from the PDF file, such as with HTML. PDF documents, however, are generally designed in a visual editor, which then produces the encryption of content and layout in the Portable Document Format. In this process, the original text blocks are often split as the editor software's primary concern is to achieve the same output with the least file size. For example, each single line may be stored separately and only when viewing the document in a PDF reader, where the lines are adjoined through their absolute positions within the document page, a human can recognize the text flow with certainty. Sometimes, with special characters that are not supported well by the font, or with special formatting such as in mathematical formulas, it is not even possible to store a single paragraph or even a single word as a simple string of characters. Instead, multiple parts of text are laid over each other, meaning that without geometric information, it would be very difficult to understand the source code of a PDF document, even if it contains bits of plain text [DM06].

[2] `http://jsoup.org/`

[3] `https://code.google.com/p/boilerpipe/`

With the described issues and other problems, e.g. resulting from encoding, it is not straight-forward to extract error-free text from a PDF file. This is especially true when a semantic segmentation of the text is required, i.e., when headings, listings, paragraphs and sentence boundaries have to be recognized. What is usually evident to the reader from the layout and textual context, may only be recognizable for a machine by means of heuristics. One example is the imprint, which may be interpreted as a single multi-line sentence, when paragraphs, words, and lines are only separated by punctuation and the spaces between them. The latter heuristic, expected text margins, may also become problematic when sentences or even words are split as described above and there are slight differences between expected and observed differences in positions and margins. Therefore it is not unusual for some documents to have words joined to common sentences where they should not be, and characters separated even though they belong to the same word.

Based on observations on the produced text extraction, we selected the *iText* PDF library for Java[4] as the most suitable open-source PDF parser. To address some of the aforementioned problems, we extended the parser with additional heuristics to achieve better text segmentation and handling of special characters. In the *iText* framework, one part of the system extracts single bits of text from the original file, which is typically single, sometimes multiple, characters and their geometric positions. Another component, which we provided on our own, then takes this information to assemble their own text representation. For us, the key challenges are to correctly identify word and paragraph boundaries, since with other PDF libaries, all paragraphs are typically merged, making it more difficult to detect sentence boundaries in a next step. Therefore, we consider the x and y-coordinates and introduce line breaks or paragraph separators when there is a minimum distance between the current and last character. Furthermore, PDF documents often use different kinds of control characters which we map to plain spaces or remove, where applicable; also, we try to reduce hyphenation stemming from line breaks due to layout boundaries.

2.2.3 Parsing and Segmentation

As a final text extraction step required for us to start analyzing texts, we have to explicitly segment it into sentences and words, so that we are able to analyze those information units in isolation as well as in context. This task, however, comes with its difficulties as often the role of punctuation is ambiguous, such as “2 000” being a single number despite the space, or the dot in “Dr.” not being a sentence delimiter. Typically, a segmentation into sentences and tokens is achieved through hand-coded rules, but as our datasets will be comprised of many special documents, such as scientific publications, there will be specific notation and punctuation that may be difficult to capture with standard rules. Therefore, we use a probabilistic, maximum entropy model from the *Apache OpenNLP* library[5], which was trained on a variety of texts to detect sentence and token boundaries using text features.

OpenNLP’s sentence segmenter and tokenizer just split any textual input by reasonable margins, meaning that there is no specific error handling and noise reduction. Especially for scientific PDF documents, however, it is likely that many special characters and notation exist that is irrelevant for our purposes and either captured as single tokens or mistakenly attached to proper words. To remove such noise, we filter all tokens by removing special characters from words’ beginnings and ends and omitting tokens entirely if there are no characters left after filtering. Such characters include any kind of punctuation and, with a few exceptions, non-Latin characters; specifically, we filter on Unicode types and directionality.

[4] http://itextpdf.com/

[5] http://opennlp.apache.org/

2.2.4 Normalization

Once a clean, plain-text representation and segmentation is obtained using such techniques as described in the previous sections, machine learning algorithms can be applied to make worth out of the texts. Be it parsing of sentence structure or text classification, most techniques analyze the text on the level of sentences and words for meaningful patterns related to the text's structure or content. With natural language, however, comes variance in expressions and terminology, meaning that simple statements may be expressed using complex, possibly ambiguous grammatical constructs, or that the same information may be conveyed using distinct sets of words. On the other hand, words that are spelled the same might have different meaning, depending on their context. As a consequence, task-specific text preprocessing is usually applied to aid further analytical procedures. For the task of searching and describing textual documents, we will predominately look at content words appearing in the texts as indicators for their content and quality. Therefore, the following four preprocessing steps will frequently occur, or be relied on, in the remainder of this thesis.

Stop Words

In natural language processing, the term "stop words" typically relates to terms which do not carry any meaning for a given task and therefore can be omitted for computational performance gains. Furthermore, terms may even negatively influence the quality of results; for example, the simple approach of selecting the most frequently occurring terms for a document's keyphrases typically results in the most frequent terms of a language being chosen, such as "a" or "the". Even words that convey a meaning on themselves, such as "education", can take the role of stop words if within a collection of texts they do not help semantically distinguishing a text from others. As the most frequently applied and probably most important preprocessing step in this thesis, we therefore filter out stop words based on manually constructed word lists. Those lists were assembled throughout the development phase and typically reflect term frequency statistics and inspecting the outputs of our algorithms. Words of less than 3 characters are always regarded as stop words, since they are far less likely to rather provide meaning than harm to our approaches. For example, in scientific literature, there is often annotation such as a_i, which may be unique to a subset of documents, but still hardly indicates a topic and certainly is not suitable as a keyword. All manually defined stop words used for our implementation are listed in Appendix A.

Stemming and Lemmatization

Following grammatical rules, most terms undergo transformations — usually at their end — for expressing singular/plural, temporal information, dependencies with other words, and alike. For example, *fish*, *fish's*, *fishing*, and *fished* all represent content related to fish and in order to statistically measure the text's focus on fish without having to know the semantical relations/similarities between those terms, they should be reduced to a common form so that when counting term frequencies, the weight of the concept "fish" becomes evident in one term. The loss of fine-grained information, such as the tense or whether a term is used as a noun or a verb is usually neglectable in classification tasks, where the primary focus lies on correctly identified the main topics of a text and distinguishing it from texts of unrelated content.

To revert such modifications ("inflections"), there are two types of heuristic approaches used frequently in natural language processing: stemming and lemmatization. In lemmatization, algorithms try to restore the original lemma or "canonical form" of a word, which is comparable to a lexicon entry. For

instance, the lemma of *mice* is *mouse*. In the implementation of the approaches proposed in this thesis, for English texts, we use the *Extended Java WordNet Library* (*ExtJWNL*)[6], which offers a Java interface to *WordNet*, an electronic lexicon for the English language [M+95]. WordNet is queried with the word to be lemmatized and the lemma of the first matching lexical entry, which generally is the most relevant one, is returned. The accuracy of this approach could be increased by additionally providing the part of speech of a word, which may result in different lemmas; however, in our applications, the overall quality would only gain small improvements with the cost of unfeasibly much computational effort of automated part-of-speech tagging for large quantities of text. For German texts, we use the lemmatizer that is part of the *anna* library for natural language analysis[7].

An older, less accurate approach to reducing inflections is stemming — a rule-based approach to removing parts of words and simple modifications to obtain the stems of words. Stemmed words predominately are not proper words anymore, and especially for words with few characters often difficult to distinguish. For example, both the noun *management* as well as the verb *manage* will typically be reduced to *manag* by a standard stemmer. Following earlier argumentation, however, for our applications even merging less related terms to a common form can increase content identification capabilities and even outperform analysis on text that was only processed through lemmatization. For stemming, we use the *Snowball* library[8], which offers stemming rule sets for various languages, including English and German.

N-Grams

When segmenting a text into sentences and words, as described in Section 2.2.3, we will obtain single words as our elementary units, which, for most languages, will be separated along punctuation and spaces. In Germanic languages this generally suits our intend of having single words representing content, since compounds are written as one word, e.g., “Betriebseröffnung” or “Finanzdienstleister”. In English, however, compound words are usually separated by space, resulting in words which actually belong together due to their meaning being split by a tokenizer, e.g., “financial service provider”. Therefore, they tend to be less value to content analysis algorithms which are based on lists of words or even only unordered sets of words (also called “bag-of-words model”). Additionally, for all languages, terms often have ambiguities regarding their meaning, and to make most use of a word, it is important to consider its context within a text.

To address the problem of English compound words and to add more context, we will therefore merge words appearing together to a single word that is treated the same way as an atomic word is. Such sequences of consecutive words are typically referred to as n-grams, where n indicates the amount of single words that are joined. In all our applications, n-grams are computed after stop word removal, so that it is ensured that also with small n’s, content words are joined together. Furthermore, when referring to n-gram statistics, the n will generally indicate an upper bound for n-gram’s and that also sequences with less words will be considered. For example, collecting 3-grams for “United States of America” will result in the terms “united”, “states”, “america”, “united states”, “states america”, and “united states america”. A further variation which might help capturing context is to leave out single words to aid capturing context across longer distances. In the previous example, leave-one-out 3-grams would result in the additional term “united america”.

[6] http://extjwnl.sourceforge.net/

[7] https://code.google.com/p/mate-tools/

[8] http://snowball.tartarus.org/

Although generally unfeasible when working with large quantities of text, we will occasionally apply spelling correction when identical spelling of the same word is essential; for instance in evaluating keyphrase extraction/assignment. Since when dealing with specialized content, such as in computer science publications, many terms will occur that are not part of standard dictionaries. Therefore, the following algorithm was devised for spelling correction on the data sets used for this thesis.

Data: Term frequency statistics for the data set.
Input: The word to be checked for spelling.
Output: The correction, or the original word if it was found correct.

if *word is shorter than 3 characters or at least one dicitionary has it listed* **then**
  return *the original word.*
else
  Obtain suggestions from a spelling correction library, including the original word;
  foreach *i-th suggestion with Levenshtein edit distance* ≤ 2 **do**
    assign $score := termFrequency(suggestion) * 2^{-i} * 2^{-2*levenshtein}$
  end
  return *the suggestion with the highest score.*
end

Algorithm 1: Spelling Correction Algorithm

In this algorithm, we reflect observations made from applying spelling correction to both English and German texts. To prefer the original spelling over corrections that are less probable given their difference to the original word and their probability of being used, we try to estimate those two factors and express them numerically. The difference between two words is calculated using their Levenshtein distance, which is the minimum amount of character changes, insertions, and deletions required to transform one word into the other [Lcv66]. The general probability of a word being used is taken into account by considering the total amount of occurrences within the data set from which the word to be corrected originates. For obtaining correction suggestions, in this work we used *Hunspell*[9], which works on *MySpell* dictionaries[10].

[9] http://hunspell.sourceforge.net/
[10] http://extensions.services.openoffice.org/dictionary

2.3 Unsupervised Learning

Unsupervised learning can be regarded as the class of statistical machine learning algorithms that typically imposes the least requirements on its input data. Specifically, unsupervised learning refers to analyzing "hidden" structures and patterns in any given data, without the need for specific annotations, such as a target variable, or feedback regarding its performance. Hereby, such algorithms are distinct from other classes of algorithms, such as in supervised learning and reinforcement learning, where a more traditional understanding of learning is followed by requiring training data from which the expected outcomes can explicitly be learned. Typical applications for unsupervised learning are clustering — the grouping of data points by common characteristics —, dimensionality reduction, where data points are mapped into lower-dimensional spaces, density estimation, and special cases of graphical models.

In clustering, the idea is to map high-dimensional data points to a set of groups or so-called clusters, which can be regarded as previously unknown classes or labels representing similar data points. To represent data points and to measure similarities for grouping them, they first have to be represented as vectors of same dimension; for example, in natural language processing, a vector representing one text could hold the term frequencies of the same fixed set of terms that is used for each text's vector. Clustering as described can also be seen as a strong form of dimensionality reduction; data points that may have a dimensionality of well over thousand could be sufficiently represented by a single cluster identifier. One other of the most frequently used algorithms for dimensionality reduction is *principal component analysis* (PCA) [Jol05], in which vectors in a lower-dimensional space are derived such that they form an optimal basis for representing the higher-dimensional data points through linear combinations between the base vectors and their lower-dimensional feature vectors without much loss. Following a similar principle, so-called auto-encoders consider a network of input and output equations, so that when a vector is passed through the network, the network's functions first map the vector to a lower-dimensional space, and then use the low-dimensional result as input to an output layer which tries to map it back to the original space [DSY+10] [LR10a]. The correspondence between original input and resulting output is an indicator of how well the dimensionality reductions works, i.e, of how representative the lower-dimensional vectors are. It used as a feedback for the network's quality and in an iterative training, the network is tuned to optimize the feedback for a set of training data. Again, no manual examples of good compression are required, but still, if successfully learned, the intermediate representation, i.e. the result of the first function layer, can be used as a lower-dimensional representation, much as with clustering and PCA.

Combining those ideas, many models exist that use an intrinsic quality measure that acts as a replacement for annotated training examples or performance feedback from "outside". In image denoising, for example, models exist that evaluate the quality of an image resulting from applying various transformations. Such a quality estimate typically considers how "smooth" an image is, i.e., how many pixels have similar values as their neighborhood, and how close it is to the original image. This feedback can then be used to adjust the transformation steps in a way that the training images have their denoising quality estimates maximized. Similarly, in natural language processing, models exist that estimate and optimize plausibility measures. For example, in *latent Dirichlet allocation*, a model is described that considers topics, which are previously unknown and will be learned as distributions over words, and documents which themselves will be described as distributions over topics by assigning one topic to each word inside the text [BNJ03] [HMM12]. The structure of the hidden topics is then learned by defining them in a way which makes the texts' distributions over topics plausible — a small amount of topics should be sufficient to explain each text document.

Another unsupervised learning application that is important for this thesis is network analysis. In network analysis, a graph of nodes and connections is constructed and then various algorithms can be applied for calculating various measures for the network. In natural language processing, for example, those nodes could represent words, sentences, or even entire texts, with the graph edges holding information about their similarity. Then, among others, algorithms for graph centrality, such as PageRank can be applied to estimate which nodes are most important in the sense of representing most of the elements in the graph [PBMW99]. This idea, for example, in this thesis will be used for calculating the importance of a sentence for a text's summary [ER04].

2.4 Supervised Learning

2.4.1 Introduction

The difference between supervised and unsupervised learning is that for supervised learning the task is explicitly given, along with training data illustrating the expected behavior. In the general supervised learning framework, this is done by providing a set of training points $D = \{(x_1, y_1), ..., (x_n, y_n)\}$, where x_i is a multi-dimensional vector of a data point's intrinsic features, i.e., those that can always be observed, and y_i is typically a single value indicating a hidden feature whose characteristics shall be learned. Specifically, the goal of supervised learning is to find a function $f(x; w)$ which, given a parameter vector w, produces the best possible estimates of y:

$$f(\mathbf{x}_i; w) \approx y_i \tag{2.1}$$

Such a function, or model equation, is usually selected or specifically designed by hand and offers variable aspects which affect its outcomes — represented by w. In the learning phase, the training data is then used for tuning the model parameters in a way which reduces the expected error. This is typically done by choosing both, the specific function, as well as its parameters, in a way that the error on the training data, which is expressed through a loss function, is minimized:

$$\arg\min_{f,w} L(f(\mathbf{x}; w), \mathbf{y}) \tag{2.2}$$

Using such a formulation, however, imposes the risk of overfitting a model to the training data, having the error on new data diverging from the error expected from the training data. In an extreme example, a loss of 0 for the training data may be achieved by mapping the unique identifiers of training points to the respective values. Obviously, new data could not even be processed due to lacking mappings for their identifiers, or would obtain poor, random estimates. But even for sound model formulations, the risk of overfitting exists where training data is sparse but located in a high-dimensional feature space, and therefore tends to misjudge random occurrences as patterns. To prevent such behavior, most common supervised learning models use regularization techniques, such as penalties on overly complex rules.

Supervised learning algorithms are typically categorized by the type of their target variables y. In the regression problem, target variables are real-valued and therefore often estimated through a linear regression, where f takes the form of a linear combination of features $\mathbf{x}$ and w are its coefficients:

$$y_i = w_0 + \sum_j w_j x_{ij} + \epsilon_i \tag{2.3}$$

Here, ϵ_i is an error term which shall be minimized across all data points when estimating w_j from the training data. To prevent overfitting, the loss function, which will guide parameter estimation, could include a penalty on using too many features. The following example is the mean squared error using l_2 regularization:

$$L = \frac{1}{N} \sum_i^N (w_0 + \sum_j w_j x_{ij} - y_i)^2 + \sum_i w_i^2 \tag{2.4}$$

Regression will be performed throughout this thesis, since most tasks can be formulated as predicting the probability of assignments or the scores assigned to items or actions. However, we will rather use *Pace Regression* [WW99] for quick and simple linear regression, which generally outperforms standard linear regression algorithms through the use of dimension reduction techniques.

Another common class of supervised learning algorithms refers to the classification task. In it, the class labels y are not real-valued, but taken from a finite set of values, which are not required to be ordinal. For example, in the classic example of whether an email is spam or not, the target values take the binary values of either true or false. Such two-class problems, also referred to as binary classification, can be solved using linear regression algorithms by mapping their outcomes to a logistic curve [NP73]. As a result, function values only take values between 0 and 1, therefore placing it on a scale which can be interpreted as expressing the likelihood of being either false or true. The linear regression model becomes reformulated as follows, which also means that different means of inferring its coefficients are required to obtain optima on the training data.

$$y_i = \frac{1}{1 + exp(-w_0 - \sum_j w_j x_{ij})} + \epsilon_i \tag{2.5}$$

However, there are many other approaches to binomial and multinomial classification, such as Naive Bayes models [MN+98], Bayesian networks [FGG97], Support Vector Machines [CV95], and neural networks [Hay94]. Neural networks, comparable to many of our examples on unsupervised learning, are graphical models in which multiple regression nodes are connected, having the results of a set of regression nodes as the inputs to others, with a final "layer" of nodes producing the model's output. Using such combinations of functions, more complex relations between features can be captured, therefore typically achieving higher performance than the aforementioned linear models. Another important, widely used type of regression and classification algorithms is those concerned with learning decision trees as a mean of capturing complex patterns found in data. Since decision trees can be derived and combined in a multitude of ways, typically leading to high performances across various types of data, including the tasks dealt with in this thesis, we will go into more details on them in Section 2.4.2.

A third class of supervised learning algorithms, which is important in the context of this thesis, is multi-label classification. Instead of estimating a categorical value for only one target variable, multi-label classification algorithms assign multiple classes to a data point, where the amount of labels might not even be fixed. Keyphrase assignment is an obvious natural language processing task, which can be formulated as a multi-label classification problem. But also any other information retrieval task which deals with making selections from a given set of candidates can be tackled with such algorithms. Therefore, in Section 2.4.3, we will discuss a framework for multi-label classification in more details.

2.4.2 Gradient Boosted Regression Trees

Decision trees follow a simple principle: Starting from the root of the tree, in each node, we consider a rule such as "text contains word [...]" and then follow one or the other branch, depending on whether the rule applies to our data point or not. Having followed the tree through multiple decision nodes, we finally arrive at one of its leaves, which then contains information about what value should be assigned to the target variable for this particular data point, i.e., its prediction. For example, in spam classification, the tree may "ask" us for the frequencies of various characteristic terms and then finally tells us the spam probability according to the email's term frequency characteristics. It may appear that such a tree works best for classification tasks, where one path through the tree corresponds with exactly one class which is assumed to be most likely. However, many data sets with continuous response variables exist that also work well enough with decision trees and often even better than with linear regression algorithms; for example, in car prices prediction for car auctions, "classes" of prices can be observed, where similar cars have comparable prices for which the smaller differences of a few dollars are neglectable. In Figure 2.4.2, we see a 3-level decision tree (or rather: regression tree) grown for a dataset that has expert quality predictions for white wines.

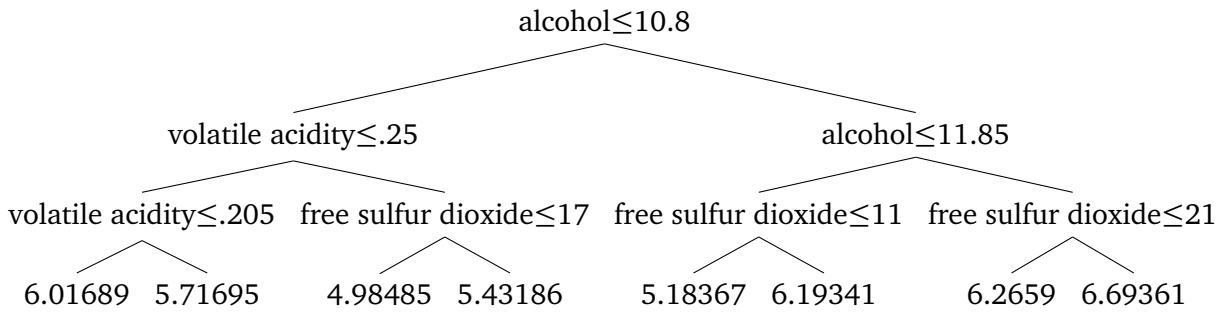

Figure 2.1.: Regression Tree Grown on Wine Quality Prediction Dataset

Once possible values are limited to a finite set of values and possibly even discretized, in practice our regression rather becomes a classification; however, with a numeric regression approach, it is easier to model means, as we will see later. Due to the technical similarity between classification and regression, on the other hand, it is also possible to map classification tasks to regression problems; for example, by translating boolean variables (true/false) into numeric "1" and "0". This allows us to design a general, combined approach to almost any regression and classification problem in the course of this thesis. As we will see, despite not being optimized for this domain, our regression tree approach will still outperform most "native" classification algorithms.

For constructing regression trees, we will follow an early idea of splitting the training data in a way that it becomes straightforward, yet precise enough to assign constants as predictions for each group, i.e., each leaf of its tree and its associated training data [BFSO84]. In particular, a node splitting criteria is to be found that when splitting the training data relevant for one node and taking the average of each resulting group's target variable values, the mean squared difference of variable average and individual training examples' values is minimized. Then, until convergence, each new branch is considered for further splittings of its associated data. To find the best subsets of data, in each iteration, all possible splitting rules are executed, the quality of the resulting subsets is measured, and then the best performing split introduced to the final regression tree.

In more recent approaches to training decision trees, and regression trees in particular, the search for and evaluation of splitting criteria, as well as the final design of the tree is typically subject to further strategies and algorithms. For example, trees are typically grown to a fixed depth and then branches are removed ("trimmed") if they do not contain enough significant information and thus rather tend to overfitting [Qui93]. We, however, will consider an extension of the decision tree idea which involve building and combining large quantities of slightly different decision trees and thus more complicated growing and pruning algorithms would mean that the execution time would rise too much.

The probably most popular idea of combining decision trees today is the one typically referred to as *RandomForests* or *bagged decision trees* [Bre01]. When growing a single decision trees, the layout may strongly reflect the specific training data and yet still not result in the optimal tree for this particular data. The growing procedure is greedy in the way that a node is split such that it optimizes the resulting two leaves, regardless of the implications on further splits on those leaves. Consequently, better splits in deeper branches of the trees might be prevented by bad early decisions. To overcome this issue, the strategy of *bagged decision trees* is to compute many trees based on slight modifications of the training data, hopefully leading to a variety of different "perspectives" on the data, and then to average over all single trees' predictions. Specifically, for each of the fixed amount of tree to be build, a random subset of training examples and data point features is selected.

The idea of increasing model variance by training on data subsets is not restricted to decision trees; such algorithms are also called "meta algorithms" and are typically applicable for a variety of "base models". Another meta learning strategy is *boosting*, of which a particular framework deals with iteratively improving the performance on the training data by applying the same algorithm on the differences between predictions and actual values of the training examples. In [Fri01] and [Fri02], a specific application of this principle ("gradient boosting") was proposed, which was very successfully adopted to decision trees. The idea is to use a differentiable loss function and to then learn to minimize its gradient in each iteration. In the first step, a constant α_0 is found so that the loss function is minimized when α_0 is used as the prediction for each training example:

$$\alpha_0 = \arg\min_{\alpha} loss(\alpha; \mathbf{y}) \tag{2.6}$$

Then, in an iterative process, the gradient of the loss function is calculated and a new model learned to fit the particular gradient value for each training example. The reason for this becomes evident when we consider the mean squared error loss function between prediction $\hat{y}$ and expected y, i.e., $(\hat{y} - y)^2$. It's gradient is a linear difference between prediction and expectation and as a result, when we learn to predict this difference, we can just add the predicted difference to the former prediction and hopefully obtain a better fit. In the gradient boosting framework, we will use an additional factor α_i to optimize the sum of former prediction $\hat{y}$ and predicted difference $f_i(\mathbf{x})$ over all training examples:

$$\alpha_i = \arg\min_{\alpha} loss(\hat{\mathbf{y}} + \alpha \mathbf{f_i}(\mathbf{x}); \mathbf{y}) \tag{2.7}$$

As a result, in each iteration, we obtain a fit for the differences of the previous iteration, as well as its coefficient α_i, which we then can use to update our predictions on which we will perform the next iteration.

As an addition to the standard framework, we observed that for the last iteration, it can be useful to perform an additional linear regression step to fit the final predictions to the expected outcomes. This is especially useful when a more complex loss function is used, that may not be suitable for the gradient steps, but could still be optimized for in a last, additional adjustment step. Specifically, β_1 and

β_2 are determined for the following update rule, such that the loss function for the updated predictions is optimized.

$$\hat{y}_n \leftarrow \beta_1 \hat{y}_n + \beta_2 \tag{2.8}$$

It is noteworthy that in this case, using heuristic algorithms, it is also possible to optimize for non-differentiable loss functions, which is usually not feasible for the single iterations, since iteration numbers of up to 20,000 are not unusual and heuristic parameter search for our α_i's may take too long.

For this thesis, we implemented our own version of gradient boosting, which slightly differs from others in both, the gradient boosting algorithm, as well as in growing the single regression trees for each iteration. We compared our implementation with the popular implementation for the language R [Rid] on multiple publicly available datasets and significantly performed better in each instance. Furthermore, different from other implementations, ours is optimized for large datasets and parallelization, as well as for the use of non-differentiable loss functions — even in each single iteration's update. Finally, we combined our framework with the idea of bagging by having RandomForests as the iterations' base models instead of single regression trees. However, this is generally only applicable for smaller datasets due to the time requirements.

Since our *gradient boosted regression trees* will provide the best results of a variety of machine learning algorithms we will apply to our specific problems, it should be worth an independent comparison with as many other algorithms on as many datasets as possible. For evaluating and comparing the general performance of a new algorithm, it should be applied to a variety of data sets. MLcomp.org offers a web service for testing machine learning algorithms against data sets using common formats and interfaces. By using such a centralized framework, the significance of comparisons is maximized since all algorithms undergo the same procedure and are evaluated using the same functions, therefore eliminating possible sources of variance and errors. Furthermore, MLcomp only allows an algorithm being uploaded for an entire machine learning task, without the possibility to modify work flow or parameters for a specific datasets. As a result, the fundamental capabilities of algorithms are better comparable than when only using one data set, which would mean that also fine-tuning and feature engineering play a role.

For the regression task, however, it is noteworthy that the majority of the 50+ data sets come with only a few hundred training cases having rather low dimensionality in feature space, that are often derived from simple functions. Decision tree models, instead, rather perform well on large training sets with complex, non-linear features, since their power rather comes from identifying non-linear codependencies of the features. Those often merely exist in small datasets or are hard to fit, since typical regression trees only use constant at their leaves, instead of fitting lines to the training data associated with them. Nevertheless, in the combined rankings considering performance across all data sets, the proposed algorithm outperforms all 30 regression algorithms listed on the website in February 2013. For a single example data set, the performance of the three best algorithms, as well as for a few other prominent algorithms, is reported on in Table 2.1. This data set, car features are to be put into relationship to an Actuarian scale for comparing the insurance risk with the purchase price. In Table 2.2, the three best algorithms according to MLcomp's overall ranking system, as well as, again, other noteworthy examples, are reported.

Rank	Model	Mean Squared Error
1	GBRT	0.326
2	Random Subspace	0.537
3	KStar	0.568
7	k-Nearest Neighbors	0.623
9	Multilayer Perceptron	0.729
10	Linear Regression	0.779

Table 2.1.: MLcomp's Ranking for the "autos" Dataset

Rank	Model	Datasets	Rating
1	GBRT	35	61
2	Bagging	45	56
3	SVM light (Support Vector Machine)	57	55
4	REPTree	45	54
5	Linear Regression	43	53
15	Octave Ridge Regression	16	46
22	k-Nearest Neighbors	41	41
24	Multilayer Perceptron	35	40

Table 2.2.: MLcomp's Overall Ranking Across 50+ Regression Datasets

2.4.3 Multi-label Classification and Learning to Rank

In the previous sections on supervised learning, we have given examples of statistical machine learning algorithms for modeling the characteristics of data and, in the case of supervised learning, predicting the values of latent variables. Those examples typically assumed that one target variable exists per data point; however, supervised learning problems exists where not one value is associated with each label, but a whole set of values. For example, in text categorization, a text document may be assigned to multiple categories at once. The previously described algorithms obviously are not suitable for modeling such situations; however, techniques exist to either extend those algorithms or to reformulate the multi-label classification problem in a way that makes one-dimensional supervised learning methods applicable [TK07].

One of the most common approaches to problem reformulation is to introduce multiple binary output variables, one for each label that can be assigned to a data point, and to learn to predict each using binary classification algorithms. Specifically, if there are n possible values a variable can have, n models are learned through binary supervised learning by giving all data points having the respective label assigned as positive examples and all others as negative examples.

The aforementioned approach, however, requires a limited amount of labels to make learning models for each feasible. Also, in general, there have to be significant amounts of training examples for each label to ensure that each model is significant, even though techniques exist which still perform well on sparse training data. The most significant drawback, however, is that for new data, labels can only be assigned if they were observed in the training data. While this assumption can be made for some domains, the task of keyphrase extraction, for instance, deals with a variety of labels, which are typically closely tied to their respective texts, thus making it generally impossible to already having learned models

for all expected labels of new data. As a result, while possibly achieving high precision, most multi-label classification algorithms lack recall when data sets are diverse in terms of their documents.

An alternative is provided by the closely related field of learning-to-rank algorithms, in which models are not built for each possible label, but for the relations between data points and their labels, which in turn are also described by features. As we will see in later chapters, for our keyphrase assignment task, many features exist for describing also previously unseen label candidates, which makes selecting them possible. With the increased number of items in uncontrolled label candidate sets and possible redundancies, however, simply estimating whether or not a label should be assigned may lead to undesirable selections. To account for this, learning to rank does not only mean substituting concrete labels by label features, but also to rank label assignment candidates by their relevance. More generally, given any indication of preference as training data, learning-to-rank algorithms are able to rank any items by their relevance for a "query" (a data point and its features). This is also why most foundations in this field can be found in information retrieval and, more specifically, search engine research.

In this thesis, supervised learning to rank is used within the following framework. First, a score is assigned to any pair of a training point and a label candidate, reflecting the label's relevance (note that this can also mean a boolean indicator such as "is a keyphrase"). Second, any supervised learning algorithm from the domain of one-dimensional-outcome problems may be used for predicting a candidate's relevance, given the pair's features. Finally, those labels that are estimated to be most relevant (e.g., having the highest probabilities of being correct) are assigned. Other than for search engines, where the reader can browse through the results until he found the desired information, the data sets used in this thesis contain a specific set of expected values, instead of just a mere set of positive or negative examples. For example, in summarization, it is not enough to just rank sentences by their relevance; instead, for presentation purposes, it is necessary to select the minimum amount of sentences that are required for the reader to understand the content of a document. Therefore, we extend the third step of our algorithm with thresholds on the amount of selections and their minimum relevance estimates. Since appropriate thresholds vary between data sets and evaluation methods, we additionally propose a framework for estimating those based on the training data.

As mentioned, we train a base model which, for each selection candidate, estimates a relevance measure. Then, the best items surpassing a threshold t on their relevance estimates are selected — with a minimum of n selections. For selecting n and t automatically, the learned model is applied to, depending on the size of the training set, either the training data on which the model was learned, or a portion of the training data which was taken out before. Then, the model's predictions on the training data are used in conjunction with the objective function which is used for measuring the selection's quality to find optimal n and t. Typically, optimizing parameters for a particular objective function involves taking the partial derivatives for each parameter. However, since we only consider two parameters and this step is only executed once, therefore making it neglectable for performance concerns, we instead, as in Section 2.4.2, use a heuristic search algorithm, making it possible to easily optimize for any objective function. Specifically, we use the Nelder-Mead simplex method for optimizing n and t [NM65].

2.5 Reinforcement Learning

2.5.1 Introduction

The third large branch in today's statistical machine learning research and application is reinforcement learning, which, of the three, is closest to traditional artificial intelligence and operations research. Fundamentally different from unsupervised and supervised learning, it does not just consider isolated decisions, but decisions in a process, causing implications for further steps.

In this framework, an agent operates within an environment, observing the environment's (partial) state, performing actions, and observing the consequences of these actions on its environment. Depending on the concrete problem formulation, the agent may always receive feedback during performance, further called "reward", or only within a controlled training environment or phase. For instance, in the classic reinforcement learning task of balancing a pole by moving a cart on which it is placed [BSA83], the agent is aware of how close the pole is from falling and therefore the relations of cart moving and pole balance can always be observed and learned from. In another example, learning to converse through a human-machine interface, such as a automated call customer service, however, there usually does not exist a direct, non-ambiguous feedback from human interaction measurable by the machine and therefore utilizable for learning. Instead, during a learning phase, human "trainers" (or an external model of human response) have to provide numeric feedback for each of the algorithm's actions. This second example is related to most natural language problems, where success is not directly measurable without annotated training data. For example, in this thesis, reinforcement learning will be used to sequentially ensemble text summaries. While intrinsic characteristics of good summaries exist, ultimate judgment usually can only be given through human feedback or correspondence with reference summaries.

As best illustrated by our third example, learning to play chess, reinforcement learning models typically have to factor in actions' implications on the future. Heuristics may exist to rate intermediate positions, but ultimately, a strategy can only be judged once a game is over. Therefore, when comparing possible moves to be made, not only the immediate rewards of the next state (may a checkmate be possible for me or the opponent?) have to be calculated, but also the change of likelihood of winning has to be estimated using heuristics — at best also considering a model of the opponent. It is needless to say that even within an entirely deterministic, foreseeable environment, with large state spaces (such as in chess) and the combinatorial explosion induced when considering possible paths through them, it is usually infeasible to exactly calculate the optimal strategy and therefore necessary to be content with probability instead of certainty. This problem of unforeseeable outcomes, delayed feedback, and potentially missing intermediate feedback is often tackled by an approach called temporal-difference learning [SB98], which will later be discussed in detail.

Another frequent problem within reinforcement learning applications, that is also present in learning to play chess, is that the amount of possible strategies that can be applied is too large to be able to evaluate each. Known as the Exploration-Exploitation Dilemma, this enforces us to decide whether actions estimated to be most valuable are followed in order to further explore subsequent possibilities and fine-tune predictions, or if unlikely actions are executed since the problem may not be learned well enough to discard them with certainty. This is usually dealt with by introducing randomness to the action selection process. For example, with the ϵ-greedy strategy, there is probability of ϵ that for any step the reinforcement learning actor performs, a random action is performed instead of the one that is expected to be most suitable. In this thesis, we will typically consider episodes, such as crawling for a specific topics or writing a single summary. This allows us to define ϵ as being depend on the number of the episode, such that as, for example, with 0.5^{ep}-greedy, the likelihood of random action decreases

with each episode, thus allowing to exploit once our knowledge fully once enough exploration can be assumed.

2.5.2 A Q-Learning Framework

As mentioned in Section 2.5.1, temporal-difference learning is an approach to exploring strategies in large problem spaces, where executed actions have influence on all further states and actions, and thus rewards to be gained. How to explore strategies and how to handle temporal-difference rewards is subject to the individual algorithm. In the original TD(λ) algorithm [SB98], only the value of a state s is considered, which is formulated as an expected sum over future rewards r obtained when following policy π from s at any time t, discounted by a factor γ. As this takes the form of a recursive Bellman equation, practical value inference can be achieved by just estimating the immediate reward and the discounted state value of the next state s_{t+1}, determined by π.

$$V(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S_t = s \right] = \mathbb{E}_\pi \left[r_{t+1} + \gamma^k V(s_{t+1}) \mid S_t = s \right] \tag{2.9}$$

Hereby π represents a decision rule for selecting the next state s_{t+1} based on the current state s_t, which is typically just the state s_{t+1} with the largest expected state value. For being able to also explore decisions which are incorrectly estimated to be less valuable, it is furthermore typical to use a policy which will also occasionally follow random actions during training, as described for ϵ-greedy strategies before.

When obtaining rewards from executing a transition from one state to another, the expected value of the first state can simply be updated using the difference between the expected value and the corrected expected value based on the realized reward and the expected discounted value of the next state s'. In the following equation, α serves as a learning rate, comparable to what we used in our updates for gradient boosted regression trees, so that state value estimates do not oscillate based on large differences between former estimates and new estimates, which may still be far from optimal since $V(s')$ again is an estimate — r_{t+1} is the only certain factor in the approximation.

$$V(s) \leftarrow V(s) + \alpha[r_{t+1} + \gamma V(s') - V(s)] \tag{2.10}$$

Since typically state spaces are too large to store value estimates for each possible state individually, in practical implementations states are usually represented through feature vectors. For the above equation, this means that $V(s)$ typically becomes a linear combination of learned coefficients and the input state s's feature vector, so that in update steps, rather than a single-valued estimate, we update all coefficients of V which thus also lead to updated estimates for all other states, since they are all computed using V. Even more than before, a learning rate α therefore is necessary so that large updates for one step do not effect the function's approximation to the remaining states' values.

Under the assumption of feature representations and a single function determining predictions from them, we actually introduced a general supervised learning problem as being a part of reinforcement learning. Instead of directly propagating differences to $V(s)$ as above, we can practically train any model of state values in replacement of $V(s)$. In this thesis, we do so by collecting observed rewards for given state feature vectors and by then training any base supervised learning algorithm for predicting those rewards for new states. When predicting state values in a temporal-difference framework, however, rewards are not as reliable as labels from typical supervised learning problems. Since considering expected

discounted rewards for all future actions resulting from a transition to a new state, the immediate reward for this state has to be combined with the expected reward of future actions. In consequence, the labels for our supervised learning training examples effectively depend on earlier trained prediction models and thus errors can reinforce themselves. Therefore, in an algorithm that periodically updates its base model, we also have to "forget" early training examples which were likely to contain deviating labels.

In our applications of reinforcement learning, it may furthermore not always be the case that state values in isolation should be considered. For example, at any state, a crawler may perform different types of actions, such as following a link or updating its queue. To reflect those different possible choices and the different rewards associated with them, we would have to provide comprehensive state feature vectors representing not only the currently visited Web page, but also other variables, such as the state of the queue, since they are influencing our decision making and therefore the expected future rewards. Since such characterizations of state obviously are impractical and would lead to too high dimensions, we instead consider the Q learning framework for reinforcement learning.

In Q learning, values are not estimated solely for states, but for actions executed from a state [SB98].

$$Q(s,a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[r_{t+1} + \gamma^k Q(s_{t+1}, a_{t+1}) \mid S_t = s, A_t = a \right] \tag{2.11}$$

This allows us to not only explicitly distinguish between possible actions and their specific characteristics that can be represented through features as well. We are furthermore able to train individual models for different types of actions, since they may yield very different rewards that could be difficult to approximate using a single model of expected values for all kinds of actions. Also, for practical considerations, this separation explicitly allows us to compute a state s's feature vector only once, which then can be used in conjunction with the different actions' feature vectors that are available for s.

The principle of learning Q value estimates is closely related to learning a function $V(s)$ as before. For estimating the discounted future rewards from transitioning to a new state s', we just have to additionally determine an optimal next action a', as illustrated in the following update rule for a linear learning of Q values.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r_{t+1} + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{2.12}$$

3 Linguistic Features

3.1 Overview

Applying the text preprocessing and statistical machine learning methods described in the previous chapter, we are now able to make proper use of our data when addressing a variety of natural language processing tasks. Most tasks involving semantical analysis of written text, including those described in this thesis, use reoccurring patterns and approaches to extracting features from texts, such as counting term frequencies. Therefore, in this chapter, we will first introduce several approaches to natural language feature extraction that will be relevant to more than just one of our tasks, before we will describe our specific design choices in detail in the coming chapters. First, in Section 3.2, different ways of measuring term frequencies by just counting their occurrences within a text and reference corpora, will be explained. Second, in Section 3.3, we will explain the usage of human-annotated lexical-semantic resources to further support our understanding of text and term significance. Finally, in Section 3.4, techniques for describing similarities, on word level, as well as sentence- and text level, are described.

3.2 Term Frequencies

As already exemplified using the concept of n-grams in Section 2.2.4, there are different ways of looking at a text's contents when seeking for content representatives. Words may be regarded on their own, as well as in conjunction with their neighbors — and both have their benefits and drawbacks in statistically representing semantics. Having established an index of terms, regardless of using n-grams or not, it is straightforward to compute term frequencies by simply counting their occurrences within the text. The significance of such statistics, however, can be put into question when text lengths vary much and the same absolute term frequencies represent different importance for very short and very long texts. Therefore, a straightforward approach is to normalize each term frequency abs_i by dividing it by the total number of terms within a text, thereby forming a distribution. Other, less frequently used weighting schemes are boolean indicators of whether a text holds a term or not, logarithmic scaling, and normalization using the maximum frequency, as shown in the following equations:

$$tf_i = \frac{abs_i}{\sum_i abs_i} \tag{3.1}$$

$$tf^i_{bool} = \begin{cases} 1 & abs_i > 0 \\ 0 & else \end{cases} \tag{3.2}$$

$$tf^i_{log} = \begin{cases} 1 + \log(abs_i) & abs_i > 0 \\ 0 & else \end{cases} \tag{3.3}$$

$$tf^i_{max} = \frac{abs_i}{\max_i abs_i} \tag{3.4}$$

Even though being normalized and therefore better for comparison, term frequency distributions may still not be significant if the respective terms do not represent the documents well enough. For example, except for boolean weighting, the most frequent terms of a language (such as "the" and "of" in English) will always have larger values assigned to than generally less frequent and therefore more significant

content words — with the consequence that the most frequent terms of a language tend to dominate term frequency distributions. As already discussed in our section about stop words (see 2.2.4), due to their frequency, such terms are typically not suitable for characterizing a document. Therefore, another weighting scheme, called term frequency–inverse document frequency (tf-idf), takes into account the amount of documents $|\{d \in D\}|$ a term appears in, therefore penalizing a domain's more general terms:

$$tf\text{-}idf_i = tf_i * \log \frac{|D|}{|\{d \in D : i \in d\}|} \tag{3.5}$$

As later feature importance analysis will show, however, there does not exist a weighting scheme that always works better than any other, and that a single method alone may not be sufficient to convey all important information.

3.3 Lexical-Semantic Resources And Information Content

To further analyze a term's significance, not only corpus statistics, but also human-annotated resources can be utilized. Most lexical-semantic resources, like WordNet [M$^+$95] for the English language, encode semantic relationships between terms — such as synonymy or "is-special-case-of" relations (hyponymy). Using such information, one can not only analyze the semantic similarities between words, as we will see in the next section, but also the specificity of terms, as we will see below. With human-annotated resources, however, also come the problems of small vocabularies and missing relations, often making it hard to apply methods based on such resources to very specific language, such as in scientific literature.

To overcome the issues of sparse resources as much as possible, in our implementation we combine multiple resources of different languages to achieve as much coverage of natural language as possible. To do so, we employ UBY, "a large-scale unified lexical-semantic resource", which, among others, integrate WordNet, Wikipedia, Wiktionary, and GermaNet [GEKH$^+$12]. Using UBY, we query the combined lexicons for sets of synonyms ("synsets"), for the "parents" of a synset (those terms which are a direct generalization of the given concept), and for the "children" of a synset, which are its special cases, such as "red" for the synset representing color. Furthermore, using those relationships, we construct a tree of hyponymy and hypernymy relations, allowing to better explore transitive relations. Even though lexical-semantic resources are generally meant to have tree structures, they sometimes contain false annotations leading to cycles in which, for instance, a child of a synset can at the same time be its parent. Therefore, when traversing the resource tree on various occasions, such as calculating depth or finding all children, we also have to monitor for cycles and remove them if found. Implementations for utilizing single lexical resources, such as WordNet, already exist, ours, however, is different in the way, with UBY, it uses an interface to a much broader array of lexicons. Furthermore, by using multiple resources, implementations of related algorithms have to consider more than one results, for instance, multiple synsets from different resources.

Having established a synset tree, we now can analyze its structure to estimate the significance of words and their relations. One of such measures is the information content of a term, which reflects how significant the occurrence of a term is [Res95]. In order to not just fixate on the word itself, but the concept it represents, we will query our tree interface for a term's synsets, and, for each synset, collect all direct, as well as transitive children. As a result, we will obtain a rich representation of words that can be used to relate to the original term. For instance, such a set for "car" would include "auto" as a synonym, as well as "hardtop", as a special class of cabriolets, which themselves are a specification of a car. To estimate the information content, we use the following formula, which considers the relative frequency in which elements of the word set W appear within a reference corpus.

$$ic(W) = -\log\left(\frac{\sum_{w \in W} abs_w}{\sum_{w \in C} abs_w}\right) \tag{3.6}$$

Further following this notion of using a rich representation of a concept, we can furthermore introduce introduce the idea of a concept frequency, which is related to the term frequency measures of the previous section, but does consider all possible expressions a concept can take. In the following, we will address this frequency measure as cf_W, assuming that W is constructed as described above:

$$cf_W = \frac{\sum_{i \in W} abs_i}{\sum_i abs_i} \tag{3.7}$$

Again, other normalization schemes are possible, as well as using inverse document frequency:

$$cf\text{-}idf_W = cf_W * \log \frac{|D|}{|\{d \in D : W \subseteq d\}|} \tag{3.8}$$

3.4 Similarity Measures

A third important and frequently used characterization of language, be it single terms or whole texts, is their similarity towards other word and texts. Such measures are especially important when we try to learn from examples, since identifying closely related texts means that we can derive information, for instance, from the way they were annotated. There are generally two distinct approaches to similarity measures; one is a frequentist or statistical approach, while the other uses lexical-semantic resources to establish semantic relationship.

The most frequently used statistical measure for the similarity of two texts is the cosine similarity of their term frequency vectors. The idea behind this is that when expressing similar content, the language used will typically be similar as well, and as a result the term frequency vectors of similar texts should put its weights on similar terms. Here, the normalization of language is especially important, such as using stemming and lemmatization as described in Section 2.2.4. Also, stopword filtering is generally important for such measures since general terms of a language, while not indicating specific topics, outnumber the frequencies of all content words and therefore lead to misleading similarity results. For two vectors $\vec{a}$ and $\vec{b}$, cosine similarity is defined as follows:

$$\cos(\vec{a}, \vec{b}) = \frac{\vec{a}\vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} (a_i)^2}\sqrt{\sum_{i=1}^{n} (b_i)^2}} \tag{3.9}$$

Such a distance metric for vectors not only allows to compare texts directly, by their terminology, but also to compare terms by their use in various texts. The idea hereby is that when two terms relate to similar concepts, they should appear in the same documents of a training corpus. For expressing this quantitatively, we will shortly highlight two approaches. First, each term can be represented by a vector indicating the term frequencies for all texts. Second, term frequency vectors can be computed for each term by combining the term frequency vectors of texts they appear in, weighted by the importance a term has for the single texts. Those vector then again can be compared using cosine similarity or similar vector space distance or distribution similarity measures.

In the semantic approach, typically the similarities between words are quantified by leveraging the information encoded in lexical-semantic resources. Usually, this means that relationship information is

used for estimating a distance between two words, such as how many or what kind of links (such as synonymy and antonymy) connect both. An important concept in such consideration is the one of the lowest common subsumer, i.e., in a semantic hierarchy of terms such as discussed in the previous section, the one element that is the deepest common parent to both words. Using the notion of a lowest common subsumer, the similarity of two terms can be estimated through the significance of the parent, as word pairs with a very specific parent in common tend to belong to the same concept. Resnik's similarity measure, for instance, is defined as the information content (as in equation 3.6) of the lowest common subsumer of two terms [Res95].

4 Topical Web Crawling

4.1 Problem Description

With well over 10 billion Web pages, the Internet is a vast resource for all kinds of information needs. Unfortunately, very few formal standards exist for content on the Internet, and therefore the design of Websites, as well as the quality control over content and presentation — both visual and technical — is almost entirely subject to the Website's publisher. As one consequence, not only may it be difficult for the user to quickly comprehend a Website's structure and the quality of its content, but also only a limited technical structure for the Web, given through hyperlinking, is available for automatically accessing it and retrieving information on a large scale. With the lack of standardized meta information, which would allow to filter and search an index structure underlying the Internet, 3rd party services are required for efficiently searching the Web.

Modern search engines themselves maintain large indices in which they define an internal representation of Web pages which allows a fast retrieval and ranking for specific queries executed through their interfaces. Examples for such index information are frequency counts for content words and heuristic measures for estimating the significance of a Website for the rest of the Internet through its linking structure, such as PageRank [PBMW99]. For any entered query, search engines typically try to retrieve those Web pages with the highest probability of providing quality information on the specified topics. Such results typically do not consider redundancies or exploring the specific topics in their broader domain. For example, queries for "chemistry" typically yield links to large and prominent Websites considering chemistry in general — such as departments of universities —, but seldom yield direct links to resources for the various sub-disciplines and subtopics forming the field of chemistry research. Furthermore, it is usually difficult to specify expected meta information, such as a Website's source or target audience, except from formulating appropriate queries, such as "chemistry for kids". Web directories, on the contrary, only require limited prior knowledge about the domain or about specific information needs on part of the user. Instead, they offer an entry point to entire topics by providing an explicit categorization of quality resources into the different subtopics available. Since it is difficult to automate extensive research on topics through existent automated approaches, as well as it is difficult to automatically identify a Web page's relevance, Web directories are generally created manually.

Therefore, in topical Web crawling research, methods are tried to be found which allow to overcome the aforementioned issues when searching the Web for resources on a broader topic. Instead of following the traditional crawling approach of trying to cover as large quantities of the Web (or specific areas) as possible to index them in a subsequent step, topical Web crawlers are considered with finding search strategies which allow to retrieve as many relevant documents for a given topic in the shortest time possible, while avoiding the expensive crawling of unrelated Websites. In order to do so, algorithms have to be found which correctly assert the relevance of currently visited documents and are efficient in deriving from this information estimates regarding which links should further be followed for being most promising to lead to more relevant documents. As a result, other than with the usual separation of crawling, indexing, and analyzing into individual tasks, topical crawlers have to analyze and predict, and potentially perform further training, in the process of crawling to be able to avoid following irrelevant routes. Even more than generic crawlers accessing large parts of the Web, focused crawlers following a specific, often "greedy", search strategy are prone to many pitfalls of the Web, such as technical difficulties or dynamic pages which allow following endless paths without getting anywhere. For example, in a

Website's dynamic calendar application, a topical crawler not being designed or trained carefully enough might end up visiting the pages for all available months, year by year, due to erroneously expecting to eventually approach relevant content based on the Website's and links' features, where a traditional crawler would already have left the Website in favor of also visiting the many other domains of the Web.

For training topical Web crawlers, as it is the case with any other discriminatory statistical machine learning application, positive and negative examples of Websites and crawling behavior are required to be available and labeled as such for the algorithms to be able to learn to distinguish between positive and negative outcomes. Furthermore, a topical Web crawler based on machine learning methods such as reinforcement learning (see Section 2.5) should also be capable of leveraging such examples to automatically derive strategies for following optimal long-term sequences of actions, instead of just considering isolated Web pages or links. In classical reinforcement learning tasks, training examples are typically sampled directly from the application of the algorithm in its target environment. For instance, in the example of a cart balancing a swinging pole through its movements, typical algorithms start performing random actions for the cart, observing their influence on the pole, and then try to iteratively improve their balancing strategies to avoid the pole falling down based on continuously updated models derived from the observations of cart movement and measurable pole reactions. With the vast amounts of content on the Internet, however, it is near to impossible for an untrained crawler following links from any random Website to stumble upon enough training examples of positive observations by chance. Instead, even though potentially visiting thousands of Web pages, when only observing irrelevant content (i.e., negative training examples), a crawler would never be able to learn a strategy for finding relevant Web pages as it lacks examples of how to do so, or it even lacks an idea of how relevant Web pages look like. Therefore, topical Web crawlers learning and adopting from their own behavior have to be initialized by either presenting them positive examples before even starting to crawl, or by setting the initially available URLs to Web pages from which relevant content would quickly be found by following its outgoing links — even when doing so randomly.

In this chapter, we will follow the idea of using reinforcement learning methods for topical crawling to learn to navigate Websites for which we already know that they likely contain relevant content, instead of trying to automatically discover ways of even finding such "hot zones" in the first place. The practical validity and relevance of this particular approach to topical crawling, with its constraint of requiring a partial solution to its problems as input, is also given by the DIPF Eduserver — our reference dataset for all tasks in this thesis. Many large German Web portals, for example operated by state agencies, exist that host contents for various specific educational topics. For example, lehrer-online.de is a Website that provides teaching materials on a variety of different topics, covering all traditional school subjects and beyond. Therefore, in its various categories regarding specific topics, the Eduserver contains a total of more than 1,400 links to single relevant Web pages of lehrer-online.de. Nevertheless, this does not mean that any single Web page from this site would be relevant for the Eduserver and thus it would be sufficient for a topical crawler to index the entire Website. Instead, the 1,400 links only make a small percentage of the approximately 40,000 lehrer-online.de pages indexed by Google. Furthermore, it has to be noted that the 1,400 selected links are not all placed into the same category, but that rather many pairs of links from those 1,400 are mutually exclusive with respect to the Eduserver's approach of grouping similar Web pages together in a topical hierarchy. For example, while both providing teaching materials for science education and thus being similar in content and intention, one resource might consider physics while the other resource considers chemistry. As a consequence, it would already be a challenging task alone to develop an algorithm which would be able to select a relevant subset of all 40,000 pages from lehrer-online.de for a specific Eduserver category and the topic represented through it, after the entire Website was indexed before.

In this chapter, we further extend the task of learning to distinguish between highly relevant and only moderately relevant content by trying to design a topical crawler whose task is to incorporate such decisions for simultaneously retrieving as much relevant content on a specific topic in the shortest time possible, while avoiding unsuitable contents. This extension is motivated by the fact that, given the vast size of the Web, it would be practically impossible for small to mid-size applications to index entire Websites first for subsequent filtering. Our intuition is that this crawling task is also far more difficult compared to starting from anywhere on the Web and trying to "only" find educational content in general, which could, for instance, be achieved through search engines. With our task description, a crawler does not only have to identify vaguely relevant content, it also has to be able to discriminate between different kinds of highly similar, yet only semi-relevant educational content to avoid following red herrings in "hot zones" of the Web. In fact, our crawling task can be regarded as modeling the most difficult aspect of highly specific topical crawling, and solutions found for this task would by extension also solve most problems when trying to discover relevant areas of the Web in more unconstrained crawling applications.

In Section 4.2, we will first give an overview over related work in the field of topical Web crawling and reinforcement learning-based crawling in particular. Then, in Section 4.3, we will describe the specific design for our topical crawler and its use of reinforcement learning methods for navigating and retrieving relevant Web pages. In Section 4.4, we will then describe the results of our experiments targeted at retrieving complementary content for various Eduserver categories, given their already contained Web pages as reference data. Finally, in Section 4.5, we will introduce our qualitative evaluation study in which we asked DIPF authors to rate the findings of our topical crawler.

4.2 Related Work

Similar to our approach, reinforcement learning is often used in the literature for Web crawling, since the findings of many authors confirm that it is a suitable framework for anticipating the value of following links, and for automatically learning suitable crawling strategies.

One of the first promising applications of RL-driven Web crawling was presented by Rennie and McCallum [RM+99] [MNRS99]. In their approach, they considering the text surrounding link candidates for estimating the expected rewards for following them — which were modeled within the Q-learning framework described in Section 2.5.2. Their software was then evaluated by measuring how many actions were required to find the 5% of computer science papers indexed in a university's online library, and shown to significantly outperform baseline approaches such as breadth-first search. A similar approach was follow by Grigoriadis and Paliouras, who instead of Naive Bayes models used neural networks to derive link value predictions based on their surrounding text [GP04]. More recently, RL crawlers have also been used for exploring the "deep Web", which means documents which are not directly linked to and can only be accessed through specific queries [JWF+10]. Finally, RL in focused crawling has also been used for retrieving specific information, such as biographical data on given persons [KM12].

An approach similar to the idea of reinforcement learning was used by Chakrabarti et al., who did not predict long-term rewards, but the immediate relevance of links, and used those predictions to prioritize their crawler's queue [CPS02]. Their work is especially noteworthy since they analyzed the structure of entire HTML syntax trees and, for example, derived features based on a link's HTML surroundings and tags used, such as link color. In another non-RL attempt, diligenti et al. sampled subgraphs of Web pages "surrounding" target documents via their link structure in order to derive from features a probabilistic model for estimating which links would bring the crawler closer to its target when approaching such dense structures [DCL+00]. In our final example, Almpanidis et al. used an approach based on LSI for topic modeling and analyzing the Web graph that is constructing while crawling [AKP07].

4.3 Design

4.3.1 Overview

As described in the introduction to this chapter, our crawler will be designed for efficiently navigating promising areas of the Web in its search for content relevant for a specific topic. Instead of merely being able to utilize services of the Web, such as search engines, for finding Websites which may only remotely be related to specific educational topics, our Web crawler should be capable of even learning to navigate "hot" areas of the Web where most content is related to a given topic, but only very few pages actually contain content matching the very specific topic definition. One example for those areas of the Web are large portals for educational resources that provide information similar in content and intention, which therefore have to be discriminated to find data for a specific educational category. To do so, the assumption for the presented approach will be that the crawler will not only have reference data available as a specification of a topic for which more data should be retrieved. We also rely on a set of initial URLs for which we can assume that they will quickly lead to relevant Web pages from which our crawler then can learn how to discover more. With this second requirement, we bypass the need for automatically finding initial URLs, which would typically be done by querying search engines with several requests being generated from the given topic specifications. When evaluating the software on larger amounts of topics, search engines such as Google would typically ban us from issuing more automated request as they would fear we might want to scrape their indices.

For any Web crawler to operate efficiently, it has to be guarded against many potential problems when crawling the Web, stemming from errors and problematic technical and structural designs of Web pages. For example, hyperlinks sometimes are specified in an erroneous way which may lead to additional path or query information being appended to the current URL's path and query instead of replacing it. When Websites themselves are not prepared for such errors, they may end up showing the same page again and again while ignoring those additional parts of the URL instead of removing them. As a result, a crawler may end up in an infinite loop if, based on their URLs, it assumes it is discovering new pages with every action. Therefore, in Section 4.3.2, we will first introduce a variety of specific preprocessing and filtering techniques we employ in order to be able to correctly analyze Web contents as well as to avoid common threats for Web crawlers. Those techniques include heuristics and filters for URLs, as well as we will again present concepts for identifying only relevant information from a Web page.

Then, in Section 4.3.3 we will introduce our specific design of a learning topical Web crawler, using the reinforcement learning framework for adopting its own behavior to observations made while crawling. As the DIPF Eduserver consists of several hundred topical categories, each containing a variety of representative Web pages for the respective topics, we will use this information as the input to our algorithm for characterizing the specific topics on which further information should be found. Starting to crawl those given Websites from their front pages, by chance, our topical crawler will quickly discover content which is highly related to the reference Web pages and thus can be assumed to be relevant as well — as it will also discover content which is only vaguely related and thus should be ignored. Using the idea of rewards for reinforcing positive behavior, our system will continuously give feedback to itself regarding how good previous actions were in terms of finding more relevant Web pages, e.g., by measuring the difference in similarity to the reference data of the currently visited page before and after following a link. From those observations, our algorithm is designed to derive statistical models for predicting the quality of potential future actions prior to actually executing them — such as following a new link outgoing from the current page, as opposed to returning to an previously visited site or retrieving one from

the crawler's queue. As a result, an optimally trained topical crawler would be able to follow precisely those links which yield most relevant future findings.

For learning and predicting, in order to be able to abstract over specific Web pages, actions taken, and rewards received, those have to be representable through standardized features, so that when encountering new situations, the topical crawler is able to compare the new situations with past observations by calculating the same types of features for the new findings and choices available. Therefore, in the last subsection of this chapter, we will furthermore present the various features we defined in our work.

4.3.2 Preprocessing and Filtering

As mentioned before, several threats exist for crawlers when unconstrainedly accessing the Web. First, many file types, such as images, are shared over the Web, from which not only many are irrelevant for our specific task, but can also be harmful if the transmission of large irrelevant files takes up significant portions of the time spent crawling. Therefore, we started defining a partial list of file types which we seek to filter out if their specific endings are given in their URLs:

css, js, rss, xml, png, gif, jpg, jpeg, avi, m3u, mp3, mp4, ogg, wmv, mov, ram, zip, mobi, epub, ppt, exe, rtf, bookmark, flv, xls, doc, jar

Figure 4.1.: File Endings For Which URLs Will Be Ignored By Our Crawler

Second, there exist many large Websites to which links can be found throughout the Web as they typically offer popular "Web 2.0" services, such as Facebook, Twitter, Amazon, or Google do. The problem of those with respect to topical crawling is that in most cases such links do not lead to content relevant for out task, but instead to large Websites in which a yet untrained crawler can get lost. Therefore, to reduce the amount of training time wasted on Websites for which we can assume that they won't provide relevant content, we on one hand, prohibit the crawler to follow links to English top-level domains for our specific application. On the other hand, we defined a manual list of additional irrelevant Websites which often appeared during our tests on the DIPF Eduserver reference dataset. Finally, we also defined specific keywords for which we can assume that URLs containing them are not relevant for us and thus should be avoided from the beginning. Those keywords for filtering are defined in Figure 4.2.

sitemap, impressum, /shop/, /kontakt, versenden, empfehlen, drucken, fontsize, /rss, print, kalender, uderef.php

Figure 4.2.: Strings For Which URLs Containing Them Will Be Ignored By Our Crawler

In the end of this section, we want to recall necessary steps of preprocessing content in order to be able to extract the significant parts of content from it. As already discussed in Section 2.2.1, much so-called boilerplate content of modern Web pages is re-occurring information — such as navigation links —, and thus not representative enough for the specific content provided on a single Web page, in contrast to other pages of the same Website. To separate boilerplate content from the relevant text of a Web page, in Section 2.2.1 we defined an algorithm in which we visited the front page of a Website and from there followed all outgoing links pointing to subpages of the same Website. Then, we extracted all single sentences from these pages and defined a filter containing all those sentences which appeared at least 20 times — which then would be removed from the content of any of the Website's pages due to not being significant enough. This effort in extracting content from Web pages is not only necessary for being able to analyze the contents of pages and for making right choices regarding how to handle them. Furthermore, in our quantitative evaluation, we will also rely on proper representations of content to be able to measure precision and recall of the reference data in our crawling sessions.

4.3.3 Reinforcement Learning Setup

In the reinforcement learning framework described in Section 2.5, an agent "inhabits" a state space and is allowed to perform actions which eventually lead to modifications of its surrounding and to the transition to new states. Applied to crawling, we can regard our topical crawler as an agent which, as its state variables, holds all relevant information from past crawling steps — such as the current URL and already visited pages. Based on this knowledge, the agent then can perform actions, such as following a specific link, to further progress in its crawling, which will then again lead to updates for its state variables. In order to be able to learn from past mistakes, the agent must have a sense of quantitative feedback regarding its actions from which it then can derive models for predicting optimal future behavior. As we try to find Web pages with similar content to those of a reference set characterizing a specific topic, our crawler rewards will reflect the progress in finding such similar content.

Crawler States

In our definition of a crawler's state, four variables exist. First, the currently visited Web page is stored in order to extract outgoing links from it, as well as to extract its features as information for the relevance estimates for outgoing links. Second, the incoming link is stored in order to avoid that, except explicitly required by an action, the previous Web page is immediately visited again, which may lead to infinite cycles. Third, if desired, a queue is stored, which will keep URLs that have been found before but that so far were neglected in favor of following more promising links As our last information, we will store which URLs have already been visited and how often we did so successfully — which also helps keeping track of previously encountered errors. For our initial state, we add all front pages of domains from our set of reference URLs to the queue.

Crawler Actions

In principle, several different classes of actions occur in the work flow of a typical crawler. A crawler might follow links, add pages to the index, or add and retrieve URLs to and from a queue. We experimented with modeling all those actions individually, however, it turned out that for our specific task, it would be more efficient to just model the link following process, since it is at the core of our problem. Instead of deriving a probabilistic model from the reference Web pages, which then could be used for predicting a document's relevance, we rather use the reference data itself and determine the relevance of newly visited documents directly through their similarity. Therefore, uncertainty with respect to adding URLs to the index does not have to be considered explicitly in the reinforcement learning framework through specific actions. For each visited link, it is sufficient to return a properly designed single reward, regardless of another software module indexing or discarding the link in the end. Furthermore, with modern computers being capable of many parallel computations and storing large amounts of data in memory, it is not required for the agent to learn an efficient queue maintenance strategy, such as which URLs from one page should be added to the queue before continuing with visiting the next. This is especially important since during training the models may change significantly and URLs from the queue may become relevant while they might not even have been added to the queue under the assumptions of an older model in the first place. Therefore, for each visited Web page, we "automatically" add all outgoing links to our queue as long as they were not visited before — without considering specific reinforcement learnign actions to do so. Additionally, for efficiency, we limit the queue in size so that older links are removed when new ones are added. This is done with consideration of the computational efforts when crawling; otherwise we might come to a situation where for each step

in which a new action is to be selected, the crawler would have to rank several thousand links from its queue just to determine the most relevant one.

As as result, for each step the crawler performs, the only possible actions between which it has to decide are those of following links from the current Web page, following links from the queue, following previously visited links, as well as visiting another page from the initial set of URLs given as seeds to the crawler.

Crawler Rewards

The crawler's rewards depend on the relevance of the Web pages it visits. To measure the relevance of a Web page, we compare its tf*idf vector, as defined in Section 3.2, with those of the reference Web pages, and determine the highest pairwise correspondence. In order to define the document frequency statistics for our tf*idf vectors, we consider all documents from the Eduserver dataset, thus effectively reducing the weights of those words that only reflect a general educational terminology, instead of the specific terminology for the topic that is crawled. For the remainder of this chapter, we will call this measure "cosine precision" to emphasize that, different from the traditional precision measure for information retrieval, we measure similarity instead of exact agreement when trying to determine the presence of relevant content in the retrieved data. Furthermore, we will later also define a corresponding measure for recall to evaluate how many aspects of a topic are covered by the crawling results.

$$cos\text{-}precision_{page} = \max_{page' \in ReferenceSet} cos(tf\text{-}idf_{page}, tf\text{-}idf_{page'}) \tag{4.1}$$

As a result, re-discovering a Web page from the reference set would yield a cosine precision score of 1, while those pages entirely different from any reference Web page would obtain scores close to 0. For training purposes, it can be useful to further transform this relevance metric for emphasizing relevant findings more and to "flatten out" differences in cosine precision which are rather due to noise than due to an actual difference in content relevance. Therefore, when considering crawler rewards, we furthermore use the following function which takes the cosine precision score as its input. The resulting curve can be seen in Figure 4.3.

$$score_{page} = \frac{110}{1 + e^{4 - 8 \cdot cos\text{-}precision_{page}}} - 10 \tag{4.2}$$

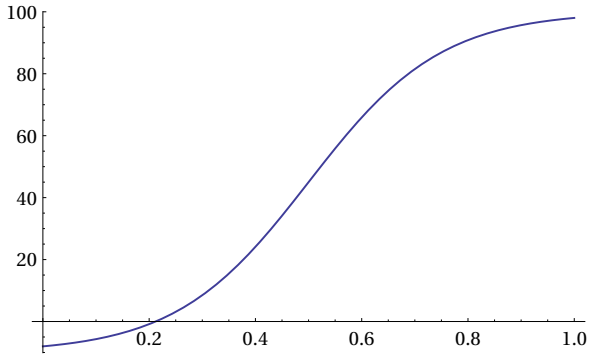

Figure 4.3.: Curve Of Our Function Mapping Cosine-Precision Values To A Modified Score

The actual rewards for our reinforcement learning crawler shall not only reflect the relevance of a visited Web page, but also its role in the progress of crawling. Therefore, we designed the following function which assigns a final reward to actions leading to the transition from the current Web page, $page_t$, to another, $page_{t+1}$:

$$reward(page_t, page_{t+1}) = \begin{cases} -20 & page_{t+1} \text{ unavailable,} \\ -10 & page_{t+1} \text{ visited before,} \\ \frac{3}{2} score_{page_{t+1}} - \frac{1}{2} score_{page_t} & \text{else} \end{cases} \tag{4.3}$$

The default condition in this equation also considers the relevance of the current Web page to reflect the improvement from a transition made. For example, executing an action which transitions from a -10 page to a 0-scoring page will yield a +5 reward just based on the improvement.

Crawler Training

Being able to observe states, to perform actions, and to receive rewards, training a reinforcement learning crawler is relatively straight-forward. As also discussed in Section 4.2, a typical approach is temporal difference learning of a Q-value function, as explained in-depth in Section 2.5. In this framework, we will consider features for both, the current state as well as candidate actions, in conjunction and try to learn models which are capable of mapping state-action pairs to reasonable estimates of future rewards. For predicting those rewards, we will not only consider immediate rewards as training data for our underlying regression models — we will also combine those rewards with expected discounted long-term rewards based on a prediction of the maximum Q value for the to be rewarded action's target Web page.

In the following sections, all features used in our implementation are given.

4.3.4 Features

URL Features

With the following features, we try to characterize three different URLs involved with any link. First, "target" represents the URL of the document that would be visited by executing this action; second, "current" represents the URL of the Web page that is currently visited by the crawler; third, "source" represents the URL on which the link was discovered — which is predominately different from "current" for links stored in the crawler's queue.

{current, source, target}_{com, net, org, de}

Four boolean indicators representing the respective top-level domain; thus can only be true for one indicator for each domain — current, source, and target.

{current, source, target}_hostLength

The total length of the current/source/target URL's host, i.e. the number of characters in subdomain and domain name.

{current, source, target}_hostParts

How many dot-separated parts the current/source/target URL's host is made of, including, for example, the amount of subdomains.

{current, source, target}_pathLength
The total number of characters describing the respective URL's path.

{current, source, target}_queryLength
The total number of query characters appended to the URL path.

{current, source, target}_pathQueryLength
The sum of pathLength and queryLength for the respective URL.

{current, source, target}_nums
The total number of digits found in a URL.

{current, source, target}_numRel
The relative number of digits found in the current/source/target URL.

{current, source, target}_contains{X}
For the following terms, a boolean feature indicates if the term is found within the URL:
schul, schuel, stud, kalender, rss, news, mitteil, feed, pdf, deutsch, press, lehrer, them, such, search, aktuell, bildung, html, php, year, month

{current, source, target}_visits
How often the precise current/source/target URL was already visited.

{current, source, target}_visitsDomain
How often the domain was already visited.

{current, source, target}_visitsHost
How often the host was already visited, i.e., the number of visits to domain+subdomain.

{current, source, target}_visitsPath
How often the URL's host and path combined were already visited.

{current, source, target}_visitsPath
Indicates if the current/source/target URL is already stored in our index.

{current, source, target}_sameDomain
Indicates if the target URL is hosted on the same domain as the current/source URL.

{current, source, target}_sameHost
Indicates if the target URL is found on the same host as the current/source URL, i.e., if it also shares the subdomain.

{current, source, target}_samePath
Indicates if the target URL is found on the same domain, host, and path as the current/source URL, i.e., only the query is different.

Page Features

The following features are used to characterize the currently visited Web page, as well as the source of a link, in terms of their relevance for the reference dataset. Most of these indicators appear to rather be relevant for characterizing the target Web pages of outgoing links instead, since we are interested in

only visiting relevant sites. However, obviously, computing those features for link targets would mean that they would have to be visited first, which would contradict our goal of avoiding irrelevant links. Still, those features are relevant for the current/source Web pages for at least two reasons: First, pages connected through links can be assumed to share common topics and thus characterizing the source page also provides implications for the targets of outgoing links. Second, by using non-linear models, we can compare the current Web page with the source page of a link to avoid leaving more relevant areas of a Website just in favor of a single link.

current_isHTML
: Boolean indicator for whether the current URL represents a HTML page, or something else, like a PDF document. This information is not required for the source document, as it had to be a HTML file for containing the link.

{current, source}_links
: The total number of outgoing links which are not filtered out (see Section 4.3.2).

{current, source}_textLength
: The total number of text characters found within the current/source Web page.

{current, source}_corpusSim
: The cosine similarity between the document's term frequency vector and the combined term frequency vector of all reference Web pages.

{current, source}_textSimMean
: The mean cosine similarity for comparing the document's tf vector which each reference page individually.

{current, source}_textSimMin
: The minimum cosine similarity for comparing the document's tf vector which each reference page individually.

{current, source}_ textSimMax
: The maximum cosine similarity for comparing the document's tf vector which each reference page individually, i.e., its cosine precision score, only using tf instead of tf*idf vectors.

4.4 Quantitative Evaluation

4.4.1 Overview

For being able to quantify the success, as well as the problems, of our approach on significant amounts of data, we will have to design a evaluation setup in which our crawler is presented with various tasks, and in which the quality of the results retrieved for each task can be measured automatically. For this purpose, we use the various categories of the DIPF Eduserver Web directory as both, reference for topics to be crawled as input to our crawler, as well as the gold standard with respect to what kind of information is expected to be retrieved from crawling. As described in the previous sections, our software is not so much designed to discover entire topics on its own by unrestrictedly crawling the entire Web, as it is designed to be able to find additional Web pages from given Websites, which discuss similar topics as the pages of a reference set do and thus can be regarded as possible additions to those sets. For a practical application of our approach to Web directory construction, this design implies that a human author first predefines topics and that he then provides a few example pages for each topic, which are then used by our crawler to find supplementary resources. This is also motivated by the DIPF Eduserver itself, as in its over 700 categories, many topics exist for which only 10-20 Web resources are stored in its database and that thus would likely benefit from an automatic discovery of further resources, providing additional information on the very same potentially underrepresented topics.

Resulting from this design, our main interest is to find out how efficient our crawler is in retrieving candidate resources for the extension of existent Web page collections on specific topics. We will evaluate this task by considering categories from the DIPF Eduserver database as our references, then selecting initial URLs to start crawling from, and finally measuring how much information could be retrieved within a fixed amount of actions performed by the crawler. This setup not only allows us to analyze the qualities and quantities of the Web pages found by our software, but also to compare it with baseline approaches performing actions within an equal amount of Web requests. For measuring the success of our topical crawler, we will compute cosine-similarity based interpretations of precision and recall. For comparison with other approaches, we will use typical learning-free crawling strategies — breadth-first search and depth-first search.

4.4.2 Metrics

Precision and recall are two typical metrics from the field of information retrieval, that measure the correspondence between a set of retrieved results and a gold standard reference set representing the information that the system is expected to provide. Precision represents the ratio of relevant content in the result set, while recall represents the ratio of gold standard data that is also available in the retrieved results. Both measures will be defined and discussed in more depth in our chapter on keyphrase extraction, since there they will be applied in their original sense. In this chapter, we will rather transfer their idea to our specific problem, which is not to find the exact same Web pages from the DIPF Eduserver database, but pages being closely related and therefore reasonable candidates for an extension of the gold standard collections. Thus, our measures of precision and recall cannot just rely on Web page equivalence, but must rather consider semantical similarities between them.

In Section 4.3.3, we have already defined cosine precision as the maximum pairwise cosine similarity when comparing a retrieved Web page with a set of gold standard pages, based on their tf*idf vectors:

$$cos\text{-}precision_{page} = \max_{page' \in ReferenceSet} cos(tf\text{-}idf_{page}, tf\text{-}idf_{page'}) \tag{4.4}$$

For those tf*idf vectors, we will use the Eduserver database as our reference corpus for document frequency statistics, which implies that it is not sufficient to share rather common terminology that can be found in most educational documents. Instead, for a high cosine similarity, both pages will have to share a terminology that is very specific to a subtopic of the Eduserver. In a fashion similar to cosine precision, we define the cosine recall of a reference Web page as the maximum pairwise cosine similarity between the page's tf*idf vector and the tf*idf vectors of the crawled results.

$$cos\text{-}recall_{page} = \max_{page' \in RetrievedSet} cos(tf\text{-}idf_{page}, tf\text{-}idf_{page'}) \tag{4.5}$$

To obtain precision and recall scores for the entire sets of retrieved and gold standard pages, we will simply average over the respective scores on single entries for each set. Finally, to combine both measures to a single score, we use the harmonic mean of the averaged precision and recall, that, different from the arithmetic mean, penalizes the mean more if one of both measures is very low. For example, an average cosine precision of 1.0 and a cosine recall of 0.1 would yield a arithmetic mean of 0.55, while the harmonic mean between both is only 0.18. For combining the traditional measures of precision and recall, the harmonic mean is also referred to as the F1 score, which is why we will name our version cos-F1.

$$cos\text{-}F_1 = 2\frac{cos\text{-}precision \cdot cos\text{-}recall}{cos\text{-}precision + cos\text{-}recall} \tag{4.6}$$

4.4.3 Baselines

When "blindly" crawling the Web, multiple a-priori strategies for selecting the hyper links to be followed are available. Breadth-first search and depth-first search are two approaches that follow very simple, yet almost orthogonal and often effective patterns, which make them popular baselines for any graph search algorithms — not just for the specific graph structure the Internet is.

In depth-first search, when visiting a new Web page, the first outgoing link whose target URL was not yet visited is followed. As a result of this strategy, depth-first crawlers tend to travel quickly between different Websites, as all others but the first link of a page will be ignored and it is unlikely that those will be found again. Breadth-first search crawlers, on the contrary, maintain an ordered queue of URLs from which always the oldest entry is taken as the destination for the next transition. The queue is populated by adding all unvisited outgoing links of a newly found Web page in order of their appearance. Therefore, theoretically, any link that is found will eventually be visited within a precise time frame — after all links that were discovered before are accessed. One of the implications of breadth-first search is that it is much more thorough in retrieving the neighborhood of the initial starting points than other strategies.

As a result, breadth-first search is a powerful baseline to our approach, considering the specific task we are evaluating against. Initializing our search at the front pages of relevant Websites, we can assume that by means of almost any strategy, many relevant subpages can be accessed through following only few "shallow" links. With breadth-first search's thorough coverage of the immediate neighborhood of entry points, it therefore exists a significant likelihood of reaching all close-by target pages within the time constraints for small- and mid-sizes Web sites. Furthermore, since our objective is to only select

links which are closely related to the reference pages of our specific crawling episode, we can extend our baseline crawlers with thresholds on the cosine precisions of retrieved pages, indexing only highly relevant URLs. As a result, for being able to outperform the baseline algorithms, and especially breadth-first search, our reinforcement learning crawler must be able to retrieve many of those "shallow" relevant documents the baselines will inevitably access, and it must furthermore be able to avoid unnecessary links for having more actions available to discover even more relevant URLs from "deeper" sections of Websites.

4.4.4 Topical Crawling Results

The specific crawling episodes forming the basis of our quantitative evaluation are derived as follows. First, all 700 categories and their respective link collections are loaded from the DIPF Eduserver database. Then, for each category, those pages are filtered out which do not have a ".de" top-level domain — as we want to focus on German Web sites alone —, or which are either not available or do not contain enough data for being a suitable crawling reference and gold standard entry. This last information is determined by applying our boilerplate-removal algorithm, as described in Section 2.2.1, for filtering out any Web page having less than 10 non-stopword tokens of original content. Third, for each category, all remaining URLs are grouped by their domain, e.g., lehrer-online.de, and only those domains are kept with their respective sub pages which have at least 5 individual URLs associated with them. The reason for this filter is that, as elaborated in Section 4.1, large Web sites exist in the Eduserver dataset that provide information on various kinds of educational topics and thus present challenging enough tasks for our crawlers. In contrast, this filter will remove small Websites being only dedicated to a single topic and therefore being only indexed in the DIPF Eduserver through their main URL. Finally, after this step, we will keep all Eduserver categories which have at least 30 single URLs left to provide a reasonable reference set, as well as a reasonable gold standard to compare our results with.

Altogether, there are 101 DIPF Eduserver categories left after applying the described filtering steps. Then, we use the front pages of those domains still being present in a single category as the initial URL seeds for our respective crawling episode. Furthermore, we will use the entire category data for reference to our crawlers, as well as we will use it as the goal standard for finally measuring our performance. The crawler's task is then to find as many comparable Web resources as possible within a window of 200 actions that can be performed for each category episode — leading to a total of over 20,000 Web pages being access throughout one simulation.

#	Model	Cos-Prec.	Cos-Rec.	Cos-F1	New_{MI}	#Sel.
1	BreadthFirst{minSim=0.5}	.8973	.1309	**.209**	.6328	6.1
2	SRL{RegTree{10, minObs=10}, 0.2, 0.5^{t-1}-Greedy, up=250}	.885	.1273	**.2008**	.7846	13.1
3	BreadthFirst{minSim=0.0}	.1455	**.2119**	.1664	**.9608**	188.2
4	DepthFirst{minSim=0.5}	**.9132**	.0986	.1594	.6972	5.6
5	DepthFirst{minSim=0.0}	.1323	.1832	.1481	**.9653**	**192.7**

Table 4.1.: Simulation Results for Topical Crawling, 200 Actions

In Table 4.1, the results of this evaluation are summarized. For both baselines, breadth-first and depth-first search, we used two different variants. The first, indicated by a "minSim=0.0" name tag, corresponds with the traditional application of indexing any Web page that is accessed while following the search strategy. The second variant measures cosine precision values for each visited page and only

stores those URLs for which a cosine precision score of at least 0.5 could be achieved. In our intuition, 0.5 is a reasonable threshold on our cosine precision metric, since due to using tf*idf vectors, a very high correspondence is required for achieving any fairly high similarity values at all. Furthermore, as we can see, using this threshold actually yields an average cosine precision close to 0.9 since we can assume an "all-or-nothing" situation for most pages — due to the non-linearity of cosine similarity, especially with tf*idf vectors, only near-exact matches achieve similarities over 0.5 while all other score poorly.

The results of our topical Web crawler can be seen in line #2. In this particular application, we used a regression tree with a maximal depth of 10 as our base model for prediction rewards. The choice of a single regression tree over more complex models, such as gradient boosted regression trees, is primarily due to performance concerns, since during 20,000 visits, many predictions and model updates will have to be performed. Furthermore, we used a discount factor of 0.2 for our rewards, as well as a $0.5^{episode-1}$-greedy policy for selecting among different choices of links to follow next. Finally, for including URLs in our final results, the crawler uses the same cosine precision threshold of 0.5.

When comparing the results, it can be found that our crawler appears to produce results being almost identical to those of breadth-first searching with thresholds. For all cosine precision-related metrics, there exists no significant differences between the means over all Eduserver categories. This actually comes as no surprise, since as we already discussed in Section 4.4.3, that initializing breadth-first crawling in the very neighborhoods of relevant documents would mean that most of those would be found. As a result, it already poses a challenge to learning crawlers to retrieve those documents as well, instead of, for instance, following red herrings to less relevant areas of the Web, as depth-first search typically does. Moreover, our crawler actually yields an important benefit over breadth-first search on the given crawling episodes — on average, it finds more than twice as many relevant Web pages. Furthermore, with almost 80% the ratio of new pages found by our crawler, i.e., Web pages which are not duplicates of pages from the reference set, is significantly higher than for breadth-first search. In our interpretation, this is due to the fact that our topical Web crawler is indeed capable of learning strategies which allow to not only retrieve rather "trivial" nearby documents, as breadth-first search does, but also to avoid irrelevant routes in favor of accessing additional parts of Websites where more relevant data can be found. Finally, as expected, our crawler significantly outperforms depth-first search, as crawlers following this strategy tend to quickly descent into less relevant areas of the Web.

4.5 Qualitative Evaluation

4.5.1 Setup

In the quantitative evaluation, we could see that our crawler was able to retrieve significant amounts of new links for a wide range of topics, only having a few examples given as input. The evaluation, as well as the crawler design itself, was based on the idea that relevant content could be identified as such through a high similarity of specific terminology when compared to a set of reference documents serving as a topic characterization. To validate this claim, and thus our results, we furthermore have to evaluate if high similarities in terms of such quantitative measures are indeed correlated with reasonable candidate proposals. In other words, we have to find out if new data retrieved by our crawler on the basis of those measures actually yield human agreement. To do so, we conducted a qualitative evaluation study in which we invited authors of the DIPF Eduserver to review crawled Web pages through an online evaluation Website. Furthermore, instead of only presenting our own results, we turned the evaluation into a blind study by randomly mixing them with links taken from the original Eduserver database — to find out if significant differences between both sets of suggested links exist.

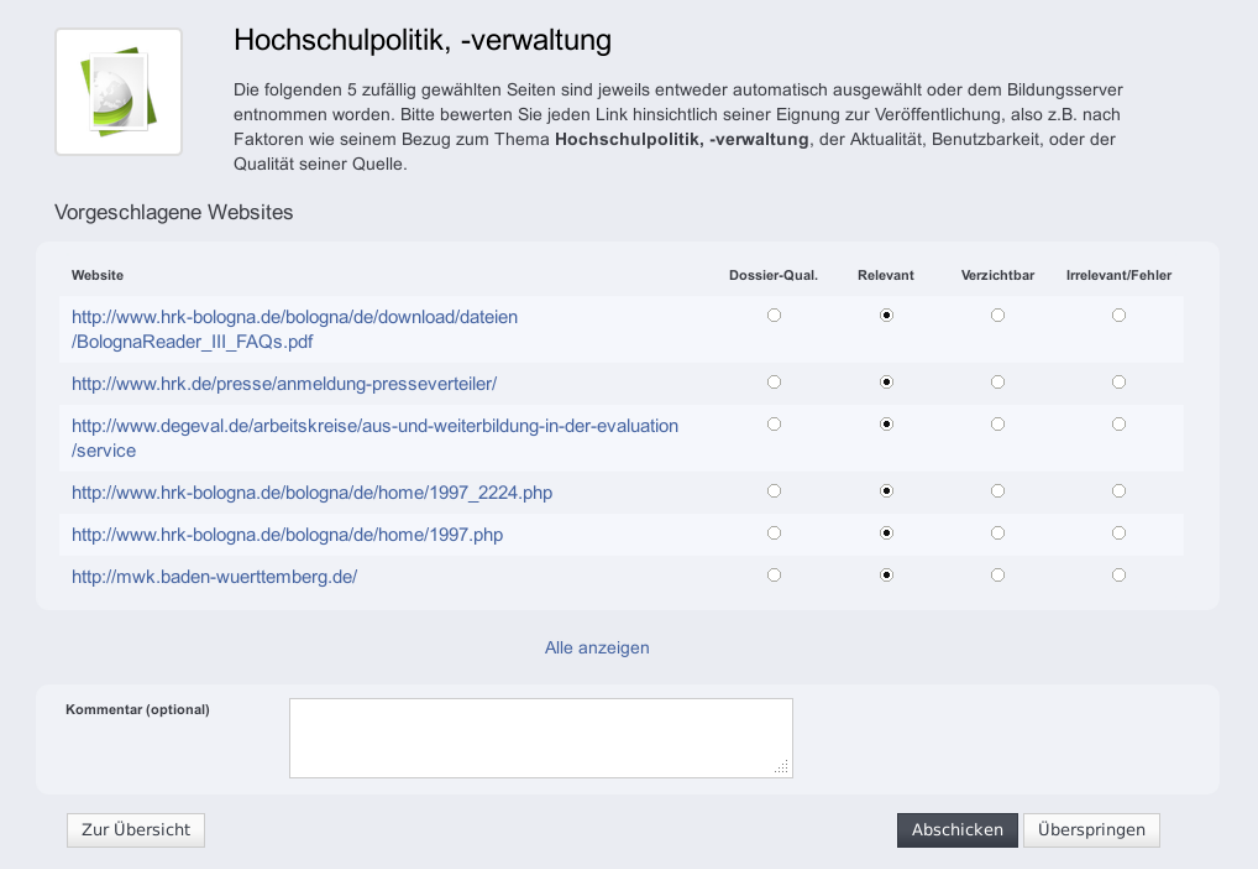

Figure 4.4.: Screenshot of Suggested Web Pages for an Eduserver Category

When accessing the online evaluation tool, a random category from the Eduserver dataset is selected and presented to the user as shown in Figure 4.4. First, the name of the category is given, along with a short description of the specific information we ask for in our evaluation. Then, six URLs are presented which are randomly drawn from the crawled results, as well as from the gold standard collection, while ensuring that at least three of them are crawler-retrieved candidates. A distinction between crawled

URLs and URLs from the DIPF Eduserver is not visible to the user in order to avoid a potential bias towards one or the other source. We then asked for relevance ratings for each of the six Web pages individually, being located on a 4-level scale. The best possibly rating a URL can receive is that it would be relevant for a DIPF dossier — which are specific pages on the Eduserver that present manual subsets of only the most relevant database entries for a given topic. The next possible ratings are either that a URL is generally relevant, or that a URL, while still relating to the given topic, can be left out of a selection since the information either is not relevant enough or already covered by other resources. Finally, an option exist to determine that a link is either entirely irrelevant or not even pointing to a proper Web page at all, e.g., presenting an error message.

4.5.2 Discussion

The preliminary results of this evaluation study are summarized in Figure 4.5. So far, we have received a total of 144 individual ratings for 24 different Eduserver categories. In the visualization, the single responses are divided by their rating classes, as well they are divided by their origin, i.e., between crawled candidates and gold standard references. The dark blue bars represent the frequencies of ratings for our crawled results, while the light blue bars represent the ratings for the Eduserver entries. In order to be able to compare both classes of entries side-by-side, we normalized each rating frequency by dividing it through the set's total number of ratings. Therefore, precisely, the bars represent empirical distributions over ratings for crawled and human-selected results.

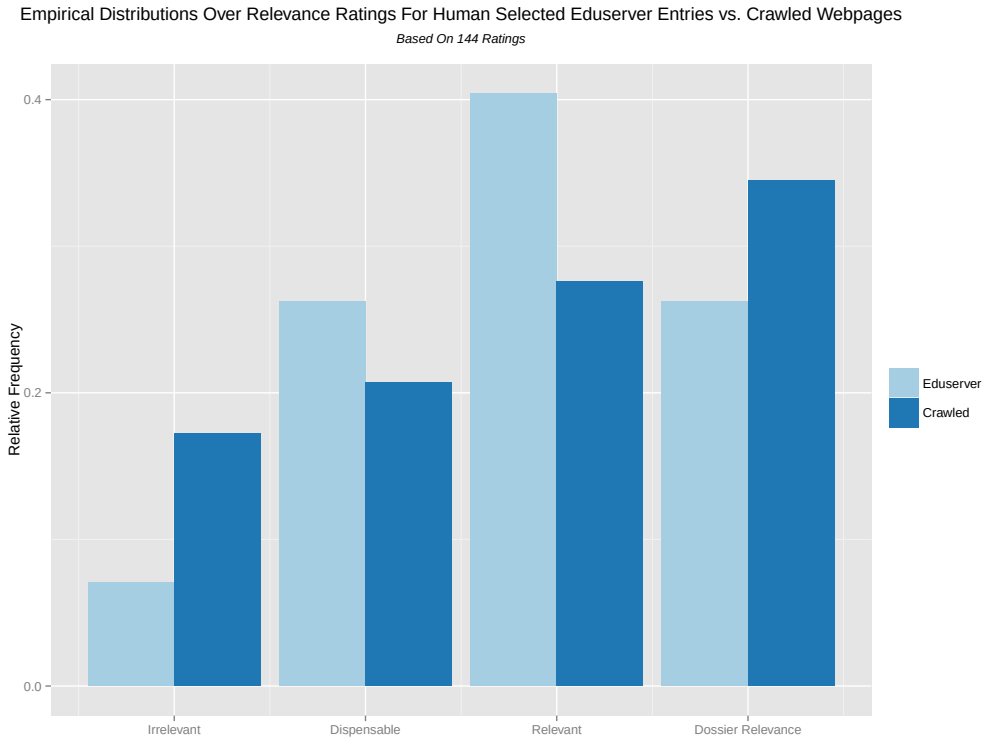

Figure 4.5.: Distributions Over Relevance Ratings For Human Selected vs. Crawled Webpages

What can be found in the data is that a majority of crawled results could actually achieve ratings of "regular" relevance or even dossier relevance. Surprisingly, with 34%, the frequency of dossier relevance is even higher than the regular relevance rating with a frequency of only about 28%. This is different from the DIPF Eduserver entries for which a relative majority of 40% is only rated as relevant. A possible

explanation for this observation is that high quality content being selected by our crawler is significantly different from the average content in the Eduserver database. As we could see in the quantitative evaluation, on average, our crawler retrieved only 13 URLs in 200 steps for each topic. With front pages to relevant Web sites as our initial seeds for crawling, it is likely that those retrieved results are generally located in "upper" areas of the Web sites, that can be accessed by following sequences of only few links. Therefore, it can be assumed that most of those pages are rather general with respect to a given topic and thus more likely to serve as a good introduction to the user — on which thus also most DIPF authors can agree. The DIPF results, on the other hand, are much larger in size and therefore it can be assumed that they additionally contain many URLs to highly specific content, which is typically found in "deeper" areas of the Websites, that are not well enough explored by our crawler. For those entries, it is more likely to observe a certain disagreement between DIPF authors regarding their relevance as dossier entries. Such a disagreement between authors would also help explaining why one third of their own links are rated as irrelevant or even erroneous by the DIPF authors. Overall, as expected, the Eduserver entries still received better ratings on average since there is also a significant amount of crawling results being rated as entirely irrelevant. For a better comparison between relevant and irrelevant content, we additionally provide Figure 4.6, where the ratings on both halves of the rating scale are merged.

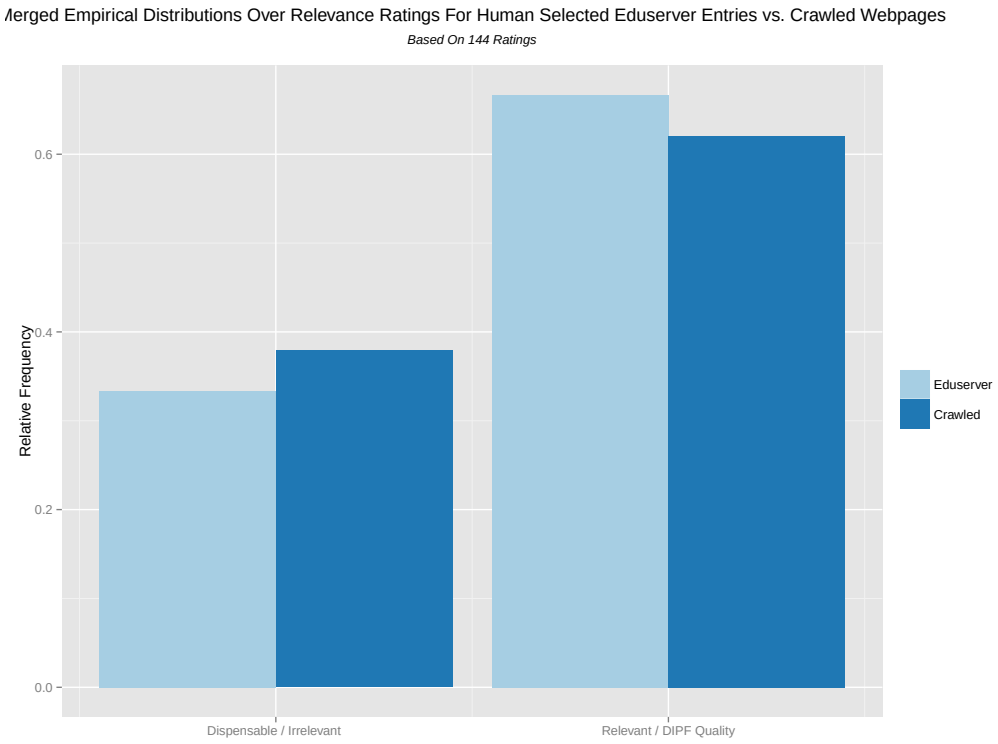

Figure 4.6.: Merged Distributions Over Relevance Ratings For Human Selected vs. Crawled Webpages

In conclusion, the qualitative evaluation of our topical Web crawler has shown that it is indeed capable of retrieving highly relevant Web pages for specific topics. Showing comparable distributions in terms of quality ratings by DIPF authors, it can even be argued that this approach could be used for fully automatically expanding the Eduserver Web directory. At least, it would provide a valuable preselection to a human author who then determines the final selection of links to appear on the Eduserver. Furthermore, by showing that cosine precision with respect to reference Web pages is indeed a valid measure of a document's relevance, this evaluation has further supported the results of our quantitative evaluation,

which relied on cosine-based performance metrics. However, for both evaluations conducted, more data and ratings will have to be obtained to allow drawing significant conclusions on more detailed levels.

4.6 Review

To summarize, in our first chapter on automating the construction of Web directories via statistical machine learning methods, we introduced the design of a focused crawler leveraging the reinforcement learning framework for inferring suitable search strategies. Being only provided with a set of exemplary Web pages for a specific topic, as well as having URLs to start searching from, the crawler then follows links it discovers with the intention of retrieving further Web resources on the given topic. While discovering new links, the crawler evaluates their contents with respect to the example Web pages by using a quantitative similarity measure — on which also a threshold for indexing found URLs is defined. Additionally to deciding on whether to index a page or not, the crawler also uses the similarity measure as a feedback for its search strategy that eventually leads to statistical models being derived for predicting the relevance of following new links without even having to load them beforehand. With the DIPF Eduserver and its many links to large educational Web portals as our motivation, we specifically designed and evaluated our crawler for its capabilities of navigating areas of the Web where most content is relevant to some extend, but for which sophisticated search strategies have to be found for efficiently retrieving very specific content.

In the quantitative evaluation of our results, we used over 100 crawling tasks in which, starting from given URLs, crawlers had to retrieve as many relevant documents as possible within just 200 actions performed. We compared our crawler with typical baseline approaches and could show that we not only kept up with their results, but could also find considerably more highly relevant resources within the given time limits. In just 200 steps, for the average topic — no matter how specific —, we retrieved over 13 new resources, of which 80% were new while still maintaining a very high cosine precision of 0.9.

Finally, in the qualitative user study conducted with the help of DIPF authors, we evaluated our fundamental claim that a high measure of similarity with respect to exemplary data actually implies a high relevance of a document for the same topic. And indeed, not only did a majority of those Web pages retrieved by our crawler and filtered based on their relevance estimates yield positive ratings by the DIPF authors, we could even observe that the distribution over ratings is comparable to the distribution for entries taken from the Eduserver Web directory. Therefore, in conclusion, we have shown that automatic systems, such as the one described, are actually capable of discovering relevant Web content for even highly specific topics in a reasonable time. Even if manual interaction would be required for removing a few entirely irrelevant URLs from the crawling results, the presented approach could still be considered as significantly improving over the human efforts necessary for designing and retrieving collections of educational documents from scratch.

For further work, we suggest that more qualitative feedback is retrieved than the time constraints on this thesis allowed, to further back our claims with more significant data. Furthermore, the described approach should be evaluated on more common, more artificial datasets which would allow a better comparison with other results published in the topical crawling literature. Finally, for improving the predictive capabilities of our Web crawler, we plan to extend its feature sets using topic modeling algorithms, as well as more specific characterizations of a link's surroundings on the source Web page.

5 Keyphrase Assignment

5.1 Problem Description

The display of a small set of 5 to 10 highly descriptive keyphrases is a fast and easy way for publishers to describe and advertise their content. Comparable to the role of document titles, keyphrase sets are designed so that the target audience is able to distinguish between information that is relevant to them and information that is not. Especially for Web directories, such information can be a useful addition to document titles and abstracts since for very specific topics, many similar publications might be hard to discriminate at first sight. Furthermore, for any search interface, keyphrases can increase the ease and accuracy of (semi-)automatically finding content for a specific topical query.

For providing fast insight into the topic and intent of a document, multiple classes of keyphrases can be used. First, on the meta level, relatively general keyphrases, such as "chemistry" for a document about a specific chemical process, help establishing the content domain if it may not become immediately clear to the reader from the document's title and publication context alone. Second, on the meta level as well, intentional keyphrases such as "teaching material" give further hints to its target audience and the intended use of the information. Finally, specific content words indicate which topics and subtopics are precisely dealt with, so that even for domain experts, such keyphrases might be helpful clues.

While prototypical content words are usually found within the document itself, for instance in its title or abstract, keyphrases on the meta level more often are a product of the publisher's own interpretation and classification with regard to the potential reader. This is especially true for Web directories like the DIPF Eduserver, which provide content with varying target audiences and levels of detail. For example, any document on photosynthesis randomly retrieved from a general education Web directory might as well be written for a University student as it may be written for a fifth grader. Therefore, a separation between different target audiences (or along any other relevant axis) either has to be made using categorization, or it has to be made clear with the use of meta information such as keyphrases. As opposed to specific topical nuances usually being denominated by a very specific terminology that is also found in the text, said meta information can be expressed in various ways and is typically not made explicit by the author, who generally can assume that his publications will only reach its target audience. As a result, a significant portion of meta keyphrases a human publisher could assign to a document might be vague, ambiguous, and often not found within the text itself, making it necessary for any automated approach, trying to approximate human annotations as good as possible, to not only extract content words, but also to assign terms based on an interpretation of the text.

The variance of possible keyphrase selections, predominately with meta keywords, but also with "competing" topical keywords, furthermore makes it difficult to train and evaluate any automated approach — even if most keyphrases could be found extractively. For example, this chapter might as well be annotated with "keyphrase assignment", as it might be annotated with "keyphrase extraction", "keywords assignment/extraction", "keywords", etc., while typically only one would be chosen to avoid redundancies. As a consequence, quantitatively measuring the agreement between two keyphrase sets, in our scenario one being provided automatically and one being a manually annotated gold standard, can become very difficult, as a precise measure would require a sense of semantic similarity between terms, which is usually difficult to establish when dealing with very specific topics, as discussed in Section 3.4. Also, with gold standard datasets like the DIPF Eduserver, which started in 1996, keyphrases used to de-

scribe a document's domain might vary over time, as well as authors, internal guidelines, and reference vocabularies do, making it even more difficult to identify a consistent policy within the data.

In this chapter, we will first give an overview of state-of-the-art automated keyphrase extraction and keyphrase assignment technology in Section 5.2. Then, in Section 5.3 we will introduce our supervised learning framework for combining keyphrase extraction with keyphrase assignment, and describe our features used to model keyphrase candidates' relations with their respective document. Third, we will quantitatively estimate the success of our approach in comparison to baseline approaches, on datasets covering multiple domains, sizes, and languages. Finally, similar to the qualitative user study conducted for our crawling approach in Section 4.5, we present the results of a study with DIPF authors on keyphrases assigned to the Eduserver dataset in Section 5.5.

5.2 Related Work

Traditionally, the field of keyphrase extraction and keyphrase assignment research is primarily split into supervised and unsupervised learning applications. In the supervised learning framework, much as we do in our approach, it is typically tried to identify relevant features for keyphrase candidates and their relations with bodies of text for the purpose of predicting their relevance for the respective documents. Unsupervised learning methods typically try to leverage the internal structures of documents and the broader semantic information contained within for identifying salient sentences and keyphrases.

Early keyphrase extraction algorithms started off with only a few basic keyphrase candidate features. 1999's KEA, for instance, is one of the earliest but also most popular baseline approaches, and uses Naive Bayes models built upon 3 to 4 features for characterizing the candidate's frequency, significance, and position within a text [WPF+99]. In the following years, much research was conducted for extending the set of basic features, as well as for applying more complex models and statistical machine learning methods to it. For example, Hulth et al. leveraged a hierarchy of thesauri for better estimating a keyphrase's role in its domain, as well as they used part-of-speech tags as features of a term's linguistic class [HKJ+01] [Hul03]. With another class of features, exemplified by Ercan et al. [EC07], the semantic structures of documents were analyzed and keyphrase candidates' contributions to those structures, such as their occurrence in lexical chains, were estimated. In her 2009 PhD thesis, Medelyan provided an extensive study of keyphrase assignment algorithms and proposed Maui — an extension of KEA and our main baseline for this chapter [Med09].

Recent state-of-the-art approaches to supervised keyphrase extraction can also be studied using publications contributed to the 2010 SemEval-2 workshop on keyphrase extraction [KMKB10], whose dataset we will use for our quantitative evaluation in Section 5.4. Placing 2nd in the workshop's competition among algorithms proposing keyphrases for a test set, the *HUMB* team built a model of bagged decision trees which used basic features, as well as multiple external resources for estimating a candidate's keyphraseness [LR10b]. The *SZTERGAK* team used a more complex set of features, covering phrase-level features, document-level features, as well as information from external resources, and thus achieved the 1st place using only a comparatively simple Naive Bayes model. [BF10] Another noteworthy approach, which ranked 3rd in the competition, is that of team *WINGNUS*, who trained a model of typical research article structures and extracted keyphrase features from them, such as if the keyphrase is present in the document's summary, abstract, or conclusion [NL10].

The most typical idea of unsupervised keyphrase extraction follows the notion of a document's graph representation in which vertices, representing either single terms or whole sentences, are analyzed and weighted through their neighborhood, thus laying explicit focus on context. In an early publication from 2002, Zha used sentence similarities based on language overlap for a spectral clustering of sen-

tences and then used the resulting clusters for computing salience scores among semantically similar sentences [Zha02]. In the iterative computation of those scores, co-occurrences of sentences and keyphrases reinforced each other, leading to a simultaneous extraction of keyphrases and summaries. In 2007, Wan et al. extended this framework by incorporating more types of relationships between sentences and words [WYX07]. Following a similar idea, Mihalcea et al. built co-occurrence graphs for keyphrase candidates, which were then used to compute salience scores using the PageRank algorithm [MT04]. This approach, as we will discuss in a later chapter of this thesis, can also be extended to automatic summarization by executing PageRank on sentence similarity graphs. In 2008, Litvak et al. introduced directions to co-occurrence graphs, thereby making it possible to apply further graph rank algorithms, such as HITS [LL08]. In their work, they also evaluated a supervised learning system which used features extracted from the co-occurrence graphs. Another possible use of co-occurrence information was proposed by Matsuo et al., who computed χ^2 statistics for keyphrase candidate's distributions over frequently occurring neighbors [MI04].

5.3 Design

5.3.1 Overview

Similar to other publications on keyphrase extraction and assignment research, our supervised learning approach is based on identifying keyphrase candidates for each document and learning to automatically rank them from training data, for which it is known if a candidate is also used in the gold standard selection we are trying to model. Due to the nature of our development datasets and especially the DIPF Eduserver, which feature both, assigned meta keywords, as well as extracted topical terms in an almost equal proportion, we had to develop a hybrid approach for achieving high quality results. To cover both, extraction and assignment, keyphrase candidates for each document are retrieved from the texts themselves, as well as from the set of keyphrases used in the training data. We then seek to find an approximation to a pointwise relevance rating function, which assigns a score to each document-keyphrase pair in the training data, reflecting its suitability. With the formulation of a training examples score, we try to find best possible supervised learning labels so that when those are predicted for new data, the induced ranking on keyphrase candidates yields best possible results. To find a most flexible approximation, we do not take documents and keyphrases "literally", such as when inferring models of individual keyphrases (e.g., distributions over terms in respective documents), but instead model the relations between documents and keyphrase candidates using a set of features being measurable for each possible pair.

5.3.2 Candidate Selection

As discussed before, gold standard keyphrases may either stem from the documents themselves, or from the interpretation and "world knowledge" of the person assigning them with respect to the specific context in which documents are presented. Therefore, we not only have to identify promising candidates from the texts, but also make use of keyphrases found in the portion of the dataset that we will use as training data for our algorithms. Luckily, as the datasets we use in this chapter only contain a few hundred to about 27,000 documents at most, fairly accurate models can be training and predictions made in a reasonable time — even with large quantities of candidate keyphrases. Therefore, instead of using various heuristics for candidate selection and selection reduction, such as only retrieving assignment candidates from similar training set documents or restricting extraction candidates to specific word

classes, we use almost all candidates and rather leave it to the more sophisticated ranking algorithms for making a finer distinction between relevant and irrelevant keyphrases. This is especially relevant for datasets with large texts, where, for example, part-of-speech tagging might not be feasible, as well for highly specific datasets where simple heuristics may yield high false negative rates.

For assignment candidates, this means that we will consider any term used more often than a fixed threshold of times within the training set — a parameter which is set per individual dataset. For keyphrase candidate extraction, in principle, we consider any n-gram occurring within a document as a potential keyphrase — even if they are only used once. To recall, an n-gram in this thesis is defined as a string of at most n subsequently appearing tokens within the text, i.e., extracting 3-grams would also produce single-word candidates. The consideration of any candidate, despite its frequency, is especially important for small texts where it may only be necessary to mention certain topical words once; or synonyms are used as a stylistic device, making it difficult to establish a sound frequency measure, which would require a more semantic than literal approach and therefore would induce more computational effort. For practical considerations, however, we have to limit the maximum amount of extracted candidates for each document, since especially for datasets such as the DIPF Eduserver, with documents of multiple hundred pages, the thousands of candidates may render the algorithms inefficient. To do so, we rank candidates by their tf*idf score, as described in Section 3.2, which, by considering document frequencies, favors candidates being both frequent within a document and rather specific in general. Furthermore, we use the same stopwords lists mentioned in Section 2.2.4 to eliminate terms for which we are certain enough that they will not help describing significant amounts of relevant semantic information from our data sets. Also, we apply stemming as described in Section 2.2.4 to reduce the grammatical variances between terms of the same or highly similar meaning. This is a perfectly valid reduction since stemming of results and gold standard references is the de-facto standard for evaluation of keyphrase extraction algorithms in scientific literature. Again, the maximum number of extracted candidates per document depends on the size of the dataset and therefore is individually set as a parameter of the dataset.

Finally, as a third source of candidate input, for some classes of text formats, meta data encoded within the file or the content markup language can be leveraged. For the DIPF Eduserver dataset, this idea predominately refers to keywords given in the header information section of HTML pages, which will also be the only additional candidate source used in our evaluation. Furthermore, candidates might be extracted from meta descriptions provided with other formats, such as PDF files.

At this point, an important consideration for candidate selection, but also feature computations, has to be made. Often, supervised learning applications are evaluated using the so-called *n-fold cross-validation* approach, in which the training data is split into n parts, and then, for each of the n parts ("folds"), predictions are made from a model trained on the remaining $n-1$ portions of the training data. Finally, the scores for all n test sets are averaged. With such a method, there exists the danger of "leaking" gold standard information into the training data parts by not carefully separating folds. Simulating keyphrase assignment, one of the main sources of falsified results is using keyphrase statistics computed before the data is split into n parts and then trained individually on the respective training portions. In particular, one might be tempted to establish a set of candidate keyphrases using all training data, and therefore also the test sets which will be designated for the n folds. As a consequence, a sophisticated predictive model could infer that given a keyphrase is provided as an assignment candidate, but not used in the fold's training data, there is a high probability for it being the right choice for a test set document dealing with a topic related to said keyphrase. Therefore, for unbiased results, assignment candidate selection, as well as computing features using other training data points, such as nearest-neighbor statistics, always

has to be strictly separated from test set data if its expected outcomes are known and should not be precomputed/cached.

5.3.3 Learning to Rank

For predicting if a keyphrase candidate should be used for our final selection, we use the learning-to-rank technique already described in Section 2.4.3. In this approach, documents are paired with single candidates and then those pairs have a score assigned to them which shall indicate relevance, so that when said score is predicted for new documents, a reasonable ranking of candidates can be derived. Other publications using a similar pairing approach typically just assign boolean indicators to training example candidates and apply classification algorithms to predict the likelihood of being relevant. While our specific datasets as well only allow boolean indicators for being in the gold standard or not, we still formulated our framework as a regression problem for continuous scores so that when other information, such as ratings or the agreement percentage among multiple annotators, would become available, they could be incorporated for more precise predictions. For instance, in our qualitative study, we asked the reader to rate keyphrases from 1 to 4, which afterwards could be used for training. Having established training labels, any standard supervised learning / curve fitting algorithm then can be used for learning the relationship between an input vector for a pair and its associated score. This particular relationship vector has to represent abstract features which can be computed for any pair, so that our models will not be restricted to the specific documents and keyphrases found in the training data:

$$\hat{\phi}(features_{document+candidate}) \approx \phi(document, candidate) \tag{5.1}$$

The features used for describing the document-keyphrase relationship will be discussed in the next section.

For our particular learning-to-rank framework we not only try to approximate the score function $\phi(document, candidate)$. We also want to automatically determine thresholds on the predicted scores and the maximum number of keyphrases to select for any document in a dataset. By doing so, we are better able to approximate the number of gold standard keyphrases, which typically varies between documents, and to better balance between precision and recall. When deriving thresholds that optimize measures such as F1 score (see Section 5.4.2), that rank precision and recall equally, we approximate thresholds where the expected gain of recall is generally lower than the expected loss of precision and thus selection should be stopped. This as well is different from most published algorithms since they typically just produce large lists of predicted keyphrases which are then cut off for evaluation. Additionally, automatically determining thresholds is important for our framework as it should also be open to continuous score functions, as described above. Different from binary classification tasks, where usually a predicted likelihood of 0.5 is a reasonable threshold for selecting or discarding items, a good threshold for any other interval or probability distribution may be hard to guess a priori.

To estimate reasonable parameters, as also described in Section 2.4.3, we have to select an objective function which will measure our success, such as precision and recall combined, and derive those parameters in a way that they optimize the objective function for our training data, and, transitively, improve it for our test data. If training data is plentiful, a separated validation set can be used so that those parameters are tuned on data which was not used for training the base model. Otherwise, models have to be selected carefully to avoid overfitting, since a too good approximation of the score function due to overfitting may lead to too optimistic thresholds, with the result that even the best candidates on a test set might be omitted. For further reducing this effect, one can also specify a minimum amount of keyphrases, which will be selected regardless of surpassing the threshold on the score estimate or not.

5.3.4 Features

In this section, we will present various features we found for representing the document-candidates pairs we use for our point-wise scoring/ranking framework described before. Please note that not every feature might be applicable or reasonable for each document and each dataset; for instance, only the DIPF Eduserverdataset provides HTML meta information. This is rather a description of all features which found an application with at least one of our datasets and for a more detailed, set-specific factor analysis, please refer to the subsequent quantitative evaluation section.

Basic Features

nthKeyword
The keyphrase's position in the candidate list sorted by first occurrences in the three candidate sources, with meta data keyphrases first, then keywords extracted from text, and assignment candidates last.

keywordLength
The total number of the candidate's characters.

keywordTokens
The number of tokens, i.e., words in the n-gram separated by punctuation.

keywordUsage
The absolute number of times the candidate has been used as a keyphrase in the training data.

keywordDocFreq
The absolute number of documents in which the keyword's stem appears at least once.

keywordness
keywordUsage/keywordDocFreq with a value of 1 for keywordDocFreq< 4 to avoid overfitting.

textLength
The total number of text characters as an indicator for the document type, which may yield different scoring policies.

firstOccRel
The relative position of the keyphrase's first occurrence within the text.

firstOccStemmedRel
The relative position of the keyphrase's first occurrence within the text after keyphrase and text have been stemmed.

lastOccStemmedRel
The relative position of the keyphrase's last occurrence within the text after keyphrase and text have been stemmed.

occSpanStemmed
The difference of characters between the stemmed keyphrase's first and last occurrence.

occSpanStemmedRel
The difference of characters between the stemmed keyphrase's first and last occurrence, relative to the total text length.

Frequency Features

tfRel
The keyphrase's frequency for the respective document, relative to the total count of terms.

tfRelLemma
The lemmatized keyphrase's relative frequency for the original respective document.

tfStem
The stemmed keyphrase's absolute frequency for the original respective document.

tfStem2
The stemmed keyphrase's absolute frequency for the respective document, after the document has been stemmed entirely.

tfRelStem
The stemmed keyphrase's relative frequency for the original respective document.

tfRelStem2
The stemmed keyphrase's relative frequency for the respective document, after the document has been stemmed entirely.

tfIdf
The keyphrase's tf*idf score, using the entire dataset's document frequencies as idf reference.

tfIdfRel
The keyphrase's tf*idf score as before, normalized by the sum of all tf*idf scores for this document.

tfCorpus
The total number of times the keyphrase has been used in the dataset's texts, excluding meta data.

tfCorpusLemma
The total number of times the lemmatized keyphrase has been used in the dataset texts.

corpusDE
The keyphrase's total count in a list of frequent terms derived from a German subtitles corpus[1].

corpusENlemma
The lemmatized keyphrase's total frequency derived from a English subtitles corpus.

corpusENstem
The stemmed keyphrase's total frequency derived from a English subtitles corpus.

Lexical-Semantic Resources

When trying to estimate the significance of a keyphrase candidate, it is important to know how specific a keyphrase is for a given topic. To some extend, unspecific terms are already targeted using stopwords filtering and tf*idf ranking in the candidate selection phase, as well as through the tf*idf features presented before. Furthermore, lexical-semantic resources such as WordNet, as described in Section 3.3,

[1] http://invokeit.wordpress.com/frequency-word-lists/

can be used to determine a candidate's role in a hierarchy of terms and ideas. The primary concepts in such a resource usually are sets of synonyms ("synsets") and their relationships, e.g., one synset being a generalization of another. We query UBY, an interface to multiple lexical-semantic resources and compute various features representing the candidate keyphrase's significance in the concept hierarchy.

synsetTFMean
The mean term frequency for words from the keyphrase's synset.

synsetTFSum
The total number of occurrences from words of the keyphrase synset.

synsetInformationContent
The synset's information content, as described in Section 3.3.

Neighborhood Features

For our final category of document-candidate features, we compare each document with all others from the training set and try to figure out how often a candidate is used in similar document. To measure two documents' similarity, we compute their term frequency vectors after having removed stop words, having stemmed all terms, and having computed n-grams, and compare them using cosine similarity, as described in Section 3.4.

trainingUsesSimilaritySum
The total sum over cosine similarity values with training documents having the keyphrase candidate in their gold standard.

trainingUsesSimilarityMean
The mean cosine similarity with training documents having the keyphrase candidate assigned.

trainingUsesSimilarityMax
The max. cosine similarity with training documents having the keyphrase candidate assigned.

trainingNotUsesSimilarityMean
The mean cosine similarity with training documents *not* having the keyphrase candidate assigned.

trainingNotUsesSimilarityMax
The max. cosine similarity with training documents *not* having the keyphrase candidate assigned.

What Did Not Work

During the development of our approach, we experimented with various other features which, however, turned out to be redundant or infeasible for larger data sets. Among may examples for basic features, such as using lemmatization additionally to stemming, there are also larger classes of features we intentionally ignored. First, we found that grammatical analysis, such as part-of-speech tagging or even extracting lexical chains did not provide enough gain which would justify the additional computational complexity. Second, unsupervised graph algorithms other than the TextRank baseline do provide some further discrimination that is picked up by most models. However, the gain again is to little to justify the additional efforts in computing the information. Finally, we also experimented with more complex

clustering and topic modeling algorithms, such as latent Dirichlet allocation, and extracted features related to distributions over keyphrase-topic relations to identify if keyphrase candidates were strongly associated with certain topics. Again, most probably due to the use of tf*idf as an approximation to this idea, the additional discriminative power was marginal. In further work, some of those approaches may, however, be revisited for further feature candidates.

5.4 Quantitative Evaluation

5.4.1 Overview

Compared to other natural language processing applications, such as topical crawling, language generation, or question answering, quantitatively evaluating keyphrase extraction and assignment methods is relatively straight-forward. Since they are mostly concerned with single, independent terms, we avoid many complexities and uncertainties of dealing with entire texts yielding manifold dependencies and quality aspects. Still, an evaluation of keyphrase selections has to be concerned with the issues of ambiguity and low-quality/incomplete gold standards.

For quantifying the success of keyphrase assignment, in the scientific literature, typically measures of correspondence between a set of predicted and a set of expected keyphrases for each document are used. For the sake of simplicity and being able to compare results, such correspondence is usually measured literally, meaning that a keyphrase selection is only deemed successful if the exact same term is also contained in the set of expected keyphrases. As a result, there exists a significant potential for false negatives in the error classification, as perfectly valid keyphrases may falsely classified as bad assignments. On one hand, topics may be expressed using various terms, such as synonyms or hyponyms, meaning that a keyphrase could be rejected even if it is more specific than its gold standard correspondent. On the other hand, the gold standard itself could be of a low quality for various reasons, such as being incomplete, using an outdated terminology, or being unspecific as the gold-standard annotator was not familiar enough with a subject. Still, being the standard in keyphrase extraction publications, we focus our quantitative evaluation, as well as the supervised learning approach itself, on the exact correspondence between predictions and expected results. To better understand the holistic quality of our keyphrase assignment results, we will furthermore present a qualitative evaluation study in the next section.

In this chapter, we will first define all employed objective functions in Section 5.4.2. Then, we will in detail discuss the results of our approach for the DIPF Eduserver dataset in Section 5.4.4, also in comparison to several baseline algorithms from Section 5.4.3. Finally, in Section 5.4.5, we will extend this evaluation to other datasets of different languages and domains to better be able to compare our results with others published on those datasets. Furthermore, we will also provide distributions over the features' significance, as determined by our best performing algorithms, for each dataset.

5.4.2 Metrics

Precision and recall are common measures in information retrieval when the correspondence between expected and provided information is to be evaluated. These two measures independently consider the problem from both perspectives: Precision measures the percentage of retrieved information being also presented in the set of expected information, while recall reversely measures the percentage of expected information being found in the retrieved data. As a consequence of this definition, a keyphrase extraction algorithm could achieve a precision of 100% if it only assigns one keyphrase that happens to be correct; for recall with 10 expected keyphrases, however, it would only achieve a score of 10%.

For both measures, there also exists the distinction between a micro precision/recall and a macro precision/recall, which, unfortunately, often is not made explicit in publications. For micro precision, we take the ratio of two sums, one being the total number of positives over all retrieved keyphrase sets, and the other being the total number of keyphrases retrieved for all documents.

$$precision_{micro} = \frac{\sum_i |\{predicted_i\} \cap \{expected_i\}|}{\sum_i |\{predicted_i\}|} \tag{5.2}$$

The definition of micro recall is comparable as it only swaps the roles of the sets of predicted and expected keyphrases.

$$recall_{micro} = \frac{\sum_i |\{predicted_i\} \cap \{expected_i\}|}{\sum_i |\{expected_i\}|} \tag{5.3}$$

Such measures, however, can give bias to unusually large predicted or expected sets of keyphrases. For example, if there was one document with 10 expected keyphrases being perfectly retrieved, and another with 100 expected keyphrases not being retrieved at all, the resulting micro recall would be 10%, while the reader's intuition might be that the success rate should be somewhat closer to 50%. Therefore, the macro formulation of precision and recall computes the ratios of matches for each document individually and then averages over all, thus leading to a macro recall of 50% in the given example.

$$precision_{macro} = \frac{1}{N} \sum_i^N \frac{|\{predicted_i\} \cap \{expected_i\}|}{|\{predicted_i\}|} \tag{5.4}$$

$$recall_{macro} = \frac{1}{N} \sum_i^N \frac{|\{predicted_i\} \cap \{expected_i\}|}{|\{expected_i\}|} \tag{5.5}$$

As mentioned before, both metrics are mostly uncorrelated, meaning that a very high recall might as well have a low corresponding precision value, as well as it may have a high precision value. For example, when choosing all keyphrase candidates for our prediction sets, recall will converge to 100%, while precision will converge to 0. Just using the arithmetic mean of both values would result in a score close to 50%, while, obviously, a precision close to zero should not be regarded as a success at all and thus rather lead to a combined score of 0. Therefore, the usual practice is to combine both scores to the so-called F1 score, using the harmonic mean and thus forcing both values to be high for gaining a high combined value.

$$F_1 = 2 \frac{precision \cdot recall}{precision + recall} \tag{5.6}$$

5.4.3 Baselines

Unsupervised Baselines and Bounds

For comparison with our algorithms, as well as for analyzing further characteristics of the datasets, such as reasonable upper bounds, several simple heuristics and unsupervised learning models can be formulated. Having obtained a large list of candidate keyphrases, as described in Section 5.3.2, and having them ordered both by origin, with meta data first and keyphrases extracted from the document second, as well as using either a natural or tf*idf-based ordering within each origin category, a self-evident idea is to evaluate simple selection heuristics on those. For said candidates list, we will select the first n candidates, where n is a parameter to the baseline; we will choose n candidates randomly;

and we will select all candidates together to find an upper bound on the recall for those candidates. Furthermore, we will assign term frequency and tf*idf scores to each candidate, as we also do for our supervised learning features (Section 5.3.4), and select the top n entries based on this ranking. Finally, for estimating upper bounds on models solely based on keyphrase extraction, we will evaluate how many stemmed candidates are actually found in the document, as well as how many gold standard keyphrases can be found using keyphrase extraction.

Furthermore, we will also report on the results for TextRank, which, as already described in Section 5.2, performs the PageRank algorithm on a co-occurrence graph [MT04]. For this purpose, a graph is constructed where each n-gram in the document is represented by a vertex and undirected edges are added between n-grams that co-occur at least once within a window of a fixed word count. The intuition is that PageRank will assign the highest scores to important vertices which are either very frequent themselves or are frequently located near other important terms, thus leading to "recommendations" between neighbors.

Kea and Maui

Kea [WPF+99] was one of the first open-source[2] supervised keyphrase extraction algorithms and therefore is established as a frequently used baseline in keyphrase extraction literature. Similar to our approach, it extracts n-gram candidates from documents or, if provided, from a controlled vocabulary, and learns a Naive Bayes model in order to distinguish between relevant and irrelevant candidates. For this Naive Bayes model, only four features are considered per candidate: its tf*idf score, the relative first occurrence of the keyphrase in the document, the total number of single words the candidate is made of, and a so-called node degree which is based on how many other candidates are semantically related to the specific keyphrase. Since typical Naive Bayes implementations work with discrete feature distributions, the computed values are discretized for computing the models.

In her PhD thesis, Medelyan describes an extension of Kea which was developed and tested on most of the datasets which are also used in our evaluation [Med09]. The Maui[3] algorithm is based on several dozens of features from multiple categories: basic features such as the tf*idf score and the candidate's index of its first occurrence; term frequency and inverse document frequency provided individually; further features related to positions, length and keyphraseness of candidates; and finally, information provided for a term lookup on a Wikipedia database. Comparable to Kea, Maui computes those features for each candidate, and based on training data candidates, models are trained for correctly predicting if a keyphrase also exist in the gold standard or not. Different from the Naive Bayes approach, however, Maui uses bagged decision trees, comparable to our implementation described in Section 2.4.2, thus making it possible to better consider the continuous nature of most features.

The use of openly available software packages for comparison is especially important, as even with small differences in the way the evaluation is set up and success is measured, the results can become incomparable. For instance, the stemming algorithms applied to predictions and gold standard keyphrases prior to computing agreement metrics might differ or be associated with different preprocessing steps, such as replacing hyphens with spaces. Furthermore, as already described in Section 5.3.2, problematic evaluation setups such as defining a domain vocabulary without excluding the test sets can effect the validity of the study and cause unrealistic results to be published. This is also partly true for Maui, which used controlled vocabularies for the reported results and therefore achieved significantly better results than when run with the same parameters but without such data available. In this case, however,

[2] http://www.nzdl.org/Kea/

[3] http://code.google.com/p/maui-indexer/

results should not be considered invalid since very broad domain vocabularies were used that also seem to contain terms which are not found in the gold standard, thus limiting the dangers of "leaking" test set results. The scope of this work rather is a different than ours; it is optimized for performance on very specific domains where controlled vocalubaries are available, while we are rather interested in more generalizable approaches, since, for instance, the DIPF Eduserver dataset covers a wide range of topics. For the sake of comparability, we call Maui exactly at the same point in our evaluation pipeline, using the same input data as we give to our own algorithms, i.e., a domain-wide vocabulary is not provided and has to be extracted from the training data on-the-fly by the algorithms.

5.4.4 DIPF Eduserver Dataset

The DIPF Eduserver database originally consists of over 27,000 links to HTML pages, PDF documents, and, in a few instances, even videos, audio files, or images. Many of those links, however, are not suitable for our evaluation of keyphrase extraction and assignment for several possible reasons. First, links may be outdated and thus leading to HTTP 404 error codes, or alike; second, we decided to solely focus on German resources, which make up close to 90% of the entire database; third, gold standard keyphrases may not be assigned carefully enough and we assume that for documents with less than 4 keyphrases, those instances are not significant enough for our evaluation; finally, documents with too few sentences are excluded from the dataset as well. This is typically the case when error messages are displayed for an URL, but no proper technical status codes are returned which would make errors better be recognizable as such. Furthermore, some Web pages may only contain very few text since most of the information is provided as pictures or downloadable material — for those examples no reasonable predictions can be expected from our approach. We set the minimum sentence count to 15. This may seem a little too high for certain Websites, but our specific segmentation algorithm (see Section 2.2.3) defines sentences even for single terms if they appear in a separate section of the page, such as individual navigation items. Therefore, for many Web pages, more than 15 "sentences" are already given through the so-called boilerplate sections of modern sites.

#	Variable	Value
1	Training Documents	8016
2	Test Documents	-
3	Language	DE
4	Mean Keyphrases	8.4 ±4.5 [4, 124]
5	Median Keyphrases	7
6	Mean Keyphrase Characters	11.6 ±4.7 [1, 38]
7	1-Gram Keyphrases	87.1%
8	2-Gram Keyphrases	11.6%
9	Evaluation	50/50 Split
10	Minimum Selection	4 Keyphrases
11	Maximum Selection	10 Keyphrases
12	Min. Training Freq. for Assignment	15
13	Top n TF Extraction Candidates	150
14	Max. N-Grams for Extraction	2

Table 5.1.: Statistics and Parameters for DIPF Eduserver

As the result of our filtering, 18,763 URLs are considered as relevant for our quantitative evaluation. However, as listed in Table 5.1, which summarizes various statistics and parameters for this dataset, we further reduced the amount of training documents to 8030 by random sampling. This is done because on one hand, once statistical machine learning algorithms achieve a certain point of confidence, adding more training data will hardly yield significant improvements in performance anymore. On the other hand, using straight-forward pairwise similarity calculations for finding nearest neighbors of documents results in an exponential complexity and more sophisticated approaches would have to be employed in order to deal with even larger quantities of training data.

Naturally, the dataset does not have a predefined separation into training and test data, so we will randomly split the training set into 50% training data and 50% evaluation data on our own. This amount of training data should be sufficient, compared to other keyphrase extraction datasets only providing a few hundred documents in total, and also allows faster computation of models than with using typical training data ratios of 70-80%. Furthermore, we see that typical documents have about 8.4 ± 4.5 keyphrases assigned to in the gold standard, and that we also enforce a minimum of 4 selections for any of our algorithms. Finally, it can be found that 87.5% of all keyphrases only consist of a single word, which, as we will see, is significantly different from English datasets where different compounding rules typically lead to more n-grams with $n > 1$. To limit the amount of keyphrase candidates, we only consider those assignment candidates appearing at least 15 times in the 50% training data portion, and the 200 most frequently appearing extractive terms in a document, excluding stopwords.

In Table 5.2, our results for various algorithms evaluated using the aforementioned settings are reported.

#	Model	F1	Prec	Rec	$F1_{MI}$	$Prec_{MI}$	Rec_{MI}	#Sel.
1	*All{ExpectedInText}*	.7165	1	.6058	.7455	1	.5942	5
2	F1{GBRT{1000,0.01,0.05,10,10}}	**.3455**	**.3884**	**.344**	**.3642**	**.405**	**.3309**	6.8
3	F1{RegTree{8, minObs=10}}	.2919	.3211	.3016	.3074	.3281	.2892	7.4
4	F1{PaceRegression}	.2608	.2523	.2959	.2614	.2523	.2711	9
5	All{NearestNeighbor}	.2339	.2406	.2468	.2392	.2264	.2535	9.4
6	Maui{12, 2, 0.01, 0.005, 5}	.1984	.17	.265	.1999	.1704	.2417	**11.9**
7	First8	.1757	.1773	.189	.1732	.1773	.1694	8
8	Top8{Tf}	.1636	.1646	.1768	.1609	.1646	.1573	8
9	Top8{Tf*Idf}	.1491	.1495	.1617	.146	.1495	.1428	8
10	All{CandidateInText}	.0474	.0259	.5274	.0458	.024	.5119	178.8
11	All	.0209	.0106	.7192	.0209	.0106	.7075	557.4
12	Random8	.01	.0107	.0102	.0105	.0107	.0102	8

Table 5.2.: Simulation Results for DIPF Eduserver

First of all, the selection of all candidates (#11) tells us that for the candidate retrieval methods and settings reported before, with an average of 557 candidates per document, a macro recall of 71.92% is achieved. This recall, however, is drastically reduced when filtering candidates based on their appearance in the document (#10). Now, only 52.74% of all gold standard keyphrases are covered by the filtered candidates set. This figure is further supported by the upper bound labeled "ExpectedInText" (#1), which selects all gold standard keyphrases that can be found in the text and which thus indicates that for any keyphrase extraction method, a maximum coverage of 60.58% of all gold standard keyphrases can be achieved. From those results we can conclude that at least 40% of all keyphrase assignments by

DIPF authors are based on abstraction rather than extraction, which not only makes it more difficult to find relevant candidates, but also to approximate the intelligent reasoning behind it.

From the baselines derived from selecting the highest weighted terms from the document's tf and tf*idf vectors (#8 and #9), we can furthermore observe that unweighted term frequencies tend to be slightly more precise than term frequencies penalized by document frequency. One possible interpretation for this is that, being restricted to a controlled vocabulary and targeting at a general audience, DIPF authors rather assign abstract terms to documents, more describing domain and intent than using specific domain vocabulary. Therefore, the most significant terms for document could be too specific and more general, still frequently used terms tend to better match the gold standard characteristics. Furthermore, copying the keyphrases of the nearest neighbor from the training set for each validation set document (#5) yields the highest baseline F1 score, which furthermore seems to indicate that similar documents have similar keyphrases assigned, which requires a certain level of abstraction to be observed in this extend. The supervised learning baseline Maui (#6) achieves the second best baseline result with an F1 score of .1984.

Using our framework and features, we produced results for multiple supervised learning base algorithms. Those are identified by the "F1" prefix, which indicates that in our learning-to-rank framework, we used the F1 metric as the reference for automatically deriving selection thresholds which optimize the training data performance. First of all, we used a linear regression algorithm called Pace Regression (#4) [WW99], which is regularized through the use of dimension reduction, and which can be found implemented in the Weka data mining library [HFH+09]. Second, we used our own implementation of a basic multi-level regression tree, which, compared to linear regression methods, is better capable of capturing higher-level dependencies between features and is better guarded against overfitting. And indeed, a single regression tree with a maximum depth of 8 and with a minimum number of 10 training examples being represented by a leaf of the tree significantly outperforms Pace Regression, as well as all baselines. Finally, the by far best performance could be achieved by using our implementation of gradient boosted regression trees (#2), which is described in Section 2.4.2. The various numbers given in the model name indicate that a total of 1000 trees of a maximum depth of 10 and a minimum observation count of 10 training examples per leaf were grown, each on a 5% sample of training examples, with a learning rate of 0.01.

For the task of keyphrase extraction and assignment, we also formulated the problem in the reinforcement learning framework described in Section 2.5. However, we were not able to identify any discriminatory features for the specific actions of adding new keywords to existing sets, that would yield significant performance gains. Instead, results of reinforcement learning algorithms were randomly scattered around the results of their learning-to-rank counterparts, which indicated that they modeled almost the same patterns, just with slightly different methods. A possible interpretation of this is that the typical keyphrase assignment process that human annotators follow itself does not consider dependencies between selected keyphrases to an extend that it would result in patterns in the data being easily exploitable by reinforcement learning. For the sake of a clean presentation, we therefore left out the description of our reinforcement learning framework for keyphrases, and will also ignore its results for the single datasets presented in this section.

Finally, as we will also do for the other datasets in this study, we report on the distribution over feature importance as determined by our best performing algorithm in Table 5.3. Easily recognizable, nearest-neighbor features are by far the most influential features when training our best performing model.

#	Feature	Importance
1	trainingUsesSimilarityMax	.1491
2	nthKeyword	.1287
3	trainingNotUsesSimilarityMax	.0843
4	trainingUsesSimilarityMean	.0683
5	tfRelStem2	.0553
6	trainingUsesSimilaritySum	.0502
7	textLength	.0452
8	trainingNotUsesSimilarityMean	.0393
9	keywordUsage	.0354
10	synsetInformationContent	.031
11	firstOccRel	.0294
12	occSpanStemmedRel	.0278
13	tfCorpusLemma	.0248
14	tfRelLemma	.0248
15	firstOccStemmedRel	.0247
16	lastOccStemmedRel	.0239
17	keywordLength	.0215
18	corpusDE	.0198
19	tfCorpus	.0197
20	synsetSensesTFMean	.0151
21	tfStem2	.0146
22	tfRelStem	.013
23	synsetSensesTFSum	.0111
24	keywordness	.0098
25	tfIdf	.0058

Table 5.3.: Top 25 Features DIPF Eduserver

5.4.5 Secondary Datasets

DIPF Pedocs

Founded in 2005, Pedocs is a second Web directory operated by the DIPF, which is specifically dedicated to openly accessible scientific literature on educational research. Due to academic publications being almost exclusively provided as PDFs (or similar file types) when offered online, for this dataset we did not have to consider the various possibilities and problems stemming from content published as Web pages. Furthermore, we were supplied with a preprocessed version of the dataset, where text was already extracted from the PDF files, which is why our specific parsers and preprocessing steps as well do not factor into the results of this evaluation.

Again, statistics regarding the dataset and general parameters to our approach are provided in Table 5.4. Compared to the Eduserver dataset, there now exist fewer documents, for which a split into training and test data was already provided by a fellow researcher. Furthermore, there are more

#	Variable	Value
1	Training Documents	2799
2	Test Documents	311
3	Language	DE
4	Mean Keyphrases	11.4 ±7.3 [1, 61]
5	Median Keyphrases	9
6	Mean Keyphrase Characters	12.2 ±4.6 [3, 47]
7	1-Gram Keyphrases	86.9%
8	2-Gram Keyphrases	12.1%
9	Evaluation	Training/Validation Split
10	Minimum Selection	1 Keyphrases
11	Maximum Selection	20 Keyphrases
12	Min. Training Freq. for Assignment	30
13	Top *n* TF Extraction Candidates	200
14	Max. N-Grams for Extraction	2

Table 5.4.: Statistics and Parameters for DIPF Pedocs

keyphrases assigned to the average document than before, which is why we also raised the maximum selection threshold.

#	Model	F1	Prec	Rec	$F1_{MI}$	$Prec_{MI}$	Rec_{MI}	#Sel.
1	*All{ExpectedInText}*	.8551	1	.7664	.8645	1	.7614	8.2
2	F1{GBRT{1000,0.01,0.05,10,10}}	**.4226**	**.497**	**.416**	**.4118**	**.461**	**.3721**	8.5
3	F1{RegTree{8, minObs=10}}	.3524	.4558	.3329	.3452	.4141	.2959	7.6
4	F1{PaceRegression}	.3241	.3308	.3785	.3266	.3243	.3289	**10.7**
5	Maui{8, 2, 0.01, 0.005, 5}	.2724	.3175	.28	.273	.3175	.2395	8
6	All{NearestNeighbor}	.2298	.2307	.2674	.2172	.193	.2482	14
7	Top8{Tf}	.1835	.2122	.1872	.184	.2122	.1624	8
8	First8	.1563	.1768	.1621	.1528	.1768	.1345	8
9	Top8{Tf*Idf}	.155	.1756	.1603	.1517	.1756	.1336	8
10	All{CandidateInText}	.047	.0247	.6092	.0482	.0251	.5831	246.6
11	All	.0364	.0189	.6678	.0366	.0188	.642	362.3
12	Random8	.02	.0241	.0195	.0208	.0241	.0183	8

Table 5.5.: Simulation Results for DIPF Pedocs

The results of the quantitative evaluation for the Pedocs dataset are given in Table 5.5. Again, an upper bound for the recall of any selection algorithm performed on our set of retrieved candidates is given in line #11. Compared to the Eduserver dataset, this time the bound on recall is lower, even though the amount of gold standard keyphrases being present in the documents is significantly higher. A simple solution for recalling more of these would be to just increase the amount of extracted keyphrases; however, for an extension of this work, a more sophisticated candidate retrieval method should be designed to avoid having to compute our models and predictions on even more predominately negative candidate examples. The remaining baseline models achieve results comparable to those for the Eduserver dataset, with Maui reaching an F1 score of .2724.

For the Pedocs dataset, our learning-to-rank framework used with any base model performs significantly better than its Eduserver dataset counterparts, and again all baselines are significantly outperformed using any base models. The best results again are achieved using gradient boosted regression trees (#2) with the same parameters used as in the last section. A possible explanation for the better results on this dataset is that, as the baselines indicate, more keyphrases are found in the text as the annotated documents are more targeted towards the research community and thus typically very specific domain terminology is available and best suitable for representing the content to the reader — while the Eduserver dataset had a much larger share of abstractive keyphrases.

Finally, the feature importance presented in Table 5.6 shows a distribution which is comparable to the feature distribution for the Eduserver dataset.

#	Feature	Importance
1	trainingUsesSimilarityMax	.1148
2	trainingUsesSimilarityMean	.1118
3	tfRelStem2	.1079
4	trainingUsesSimilaritySum	.0644
5	trainingNotUsesSimilarityMax	.0613
6	nthKeyword	.0476
7	firstOccRel	.0399
8	trainingNotUsesSimilarityMean	.0365
9	textLength	.0359
10	keywordUsage	.0348
11	synsetInformationContent	.0345
12	tfRelLemma	.0333
13	firstOccStemmedRel	.0298
14	occSpanStemmedRel	.0286
15	synsetSensesTFMean	.0282
16	tfCorpusLemma	.0281
17	corpusDE	.0239
18	lastOccStemmedRel	.0238
19	keywordLength	.0207
20	tfCorpus	.0181
21	tfStem2	.0172
22	tfIdfRel	.0157
23	synsetSensesTFSum	.0141
24	tfRelStem	.0136
25	tfStem	.0049

Table 5.6.: Top 25 Features DIPF Pedocs

In 2010, the SemEval-2 workshop for semantic evaluation[4] was held for the 5th time, offering 18 competitions primarily dealing with word sense disambiguation. For the automatic keyphrase extraction task, a total of 244 scientific articles were provided, of which 144 had the matching keyphrases published during the development phase of the competition. Then, the participants had to submit their predictions for the 100 test set documents which were evaluated on the gold standard that was only published after the competition ended.

The task was further split into two categories of keyphrases: for the reader-only task, each document had about 12-13 terms assigned by readers of the papers, while in the combined dataset those reader keyphrases were supplemented by keyphrases given by the authors. Since authors often include their keyphrases in a specific section of the paper, the combined task typically yields better results than the reader-only data. The statistics and parameters for both datasets are given in tables 5.7 and 5.8.

#	Variable	Value
1	Training Documents	144
2	Test Documents	100
3	Language	EN
4	Mean Keyphrases	15.4 ±3.9 [8, 37]
5	Median Keyphrases	15
6	Mean Keyphrase Characters	6.7 ±2.7 [2, 19]
7	1-Gram Keyphrases	20.2%
8	2-Gram Keyphrases	52.4%
9	3-Gram Keyphrases	20.6%
10	4-Gram Keyphrases	4.7%
11	Evaluation	Training/Validation Split
12	Minimum Selection	8 Keyphrases
13	Maximum Selection	12 Keyphrases
14	Min. Training Freq. for Assignment	2
15	Top n TF Extraction Candidates	1000
16	Max. N-Grams for Extraction	4

Table 5.7.: Statistics and Parameters for SemEval-2 Combined

As already discussed before, comparing results between different evaluation frameworks can be difficult, even if they are designed almost identically. However, even though the evaluation scripts are publicly available for SemEval-2, we will primarily use our general evaluation framework to calculate our typical metrics, as well as to have access to the results of the single documents for significance calculations. For validation purposes, though, we will also report on the results of our best algorithm using SemEval-2's evaluation script in the end of this section.

The official results[5] for both competition tasks are presented in Table 5.9. While the published results also contained an evaluation on the first 5 and first 10 keyphrases from each test set document, we are primarily interested in the evaluation of all 15 keyphrases the participants had to provide for each document.

[4] http://semeval2.fbk.eu/

[5] http://semeval2.fbk.eu/semeval2.php?location=Rankings/ranking5.html

#	Variable	Value
1	Training Documents	144
2	Test Documents	100
3	Language	EN
4	Mean Keyphrases	12.7 ±3.3 [7, 34]
5	Median Keyphrases	12
6	Mean Keyphrase Characters	6.8 ±2.7 [2, 19]
7	1-Gram Keyphrases	17.7%
8	2-Gram Keyphrases	53.2%
9	3-Gram Keyphrases	21.8%
10	4-Gram Keyphrases	5%
11	Evaluation	Training/Validation Split
12	Minimum Selection	6 Keyphrases
13	Maximum Selection	15 Keyphrases
14	Min. Training Freq. for Assignment	2
15	Top n TF Extraction Candidates	1000
16	Max. N-Grams for Extraction	4

Table 5.8.: Statistics and Parameters for SemEval-2 Reader-Only

#	Team	Precision	Recall	F1
1	SZTERGAK	**30.80%**	**31.51%**	**31.15%**
2	HUMB	27.20%	27.83%	27.51%
3	WINGNUS	24.93%	25.51%	25.22%
4	KP-Miner	24.93%	25.51%	25.22%
5	ICL	24.60%	25.17%	24.88%
6	SEERLAB	24.07%	24.62%	24.34%
7	KX_FBK	23.60%	24.15%	23.87%
8	DERIUNLP	22.00%	22.51%	22.25%
9	Maui	20.33%	20.80%	20.56%
10	DFKI	20.27%	20.74%	20.50%
11	BUAP	19.00%	19.44%	19.22%
12	SJTULTLAB	18.40%	18.83%	18.61%
13	UNICE	18.33%	18.76%	18.54%
14	UNPMC	18.13%	18.55%	18.34%
15	JU_CSE	17.80%	18.21%	18.00%
16	likey	16.33%	16.71%	16.52%
17	UvT	14.60%	14.94%	14.77%
18	POLYU	13.87%	14.19%	14.03%
19	UKP	5.27%	5.39%	5.33%

#	Team	Precision	Recall	F1
1	SZTERGAK	**21.47%**	**26.74%**	**23.82%**
2	HUMB	21.20%	26.41%	23.52%
3	KX_FBK	20.33%	25.33%	22.56%
4	WINGNUS	19.80%	24.67%	21.97%
5	ICL	19.47%	24.25%	21.60%
6	SEERLAB	19.33%	24.09%	21.45%
7	KP-Miner	19.33%	24.09%	21.45%
8	DERIUNLP	17.53%	21.84%	19.45%
9	DFKI	17.40%	21.68%	19.31%
10	UNICE	16.00%	19.93%	17.75%
11	SJTULTLAB	15.60%	19.44%	17.31%
12	BUAP	14.93%	18.60%	16.56%
13	Maui	14.87%	18.52%	16.50%
14	UNPMC	14.47%	18.02%	16.05%
15	JU_CSE	14.40%	17.94%	15.98%
16	likey	13.80%	17.19%	15.31%
17	POLYU	12.00%	14.95%	13.31%
18	UvT	11.93%	14.87%	13.24%
19	UKP	4.67%	5.81%	5.18%

Table 5.9.: SemEval-2 Official Results for Combined (Left) and Reader-Only (Right) Tasks

For the combined task presented in Table 5.10, we find results which are very close to those of the Pedocs dataset, which is likely caused by both datasets consisting of scientific publications. A significant

#	Model	F1	Prec	Rec	$F1_{MI}$	$Prec_{MI}$	Rec_{MI}	#Sel.
1	*All{ExpectedInText}*	.8739	1	.7877	.8822	1	.7892	11.6
2	F1{GBRT{1000,0.01,0.05,10,10}}	**.4235**	**.4926**	.3843	**.4266**	**.4875**	.3793	11.2
3	Top15{GBRT{1000,0.01,0.05,10,10}}	.4033	.3967	**.4192**	.4052	.3967	**.4141**	**15**
4	F1{M5Rules}	.3752	.4142	.3532	.3772	.4128	.3472	12
5	SRL{RegTree{8, minObs=10}, 0.2, 0.99^{t-1}-Greedy, up=100}{1000}	.3644	.4357	.3267	.3596	.4051	.3233	11.4
6	F1{RF{20,0.8,10,10}}	.3623	.471	.3	.3656	.4761	.2967	9
7	F1{RegTree{8, minObs=10}}	.3353	.3851	.3026	.338	.3868	.3001	11.2
8	F1{M5P}	.232	.2667	.2133	.2349	.2696	.208	11.1
9	Maui{15, 2, 0.01, 0.005, 5}	.1713	.1673	.1799	.1716	.1673	.176	**15**
10	Top15{Tf*Idf}	.1329	.1293	.1396	.1321	.1293	.1351	**15**
11	First13	.1291	.1346	.1266	.1281	.1346	.1221	13
12	Top15{Tf}	.1052	.102	.1109	.1042	.102	.1064	**15**
13	F1{PaceRegression}	.0947	.1025	.0903	.0939	.1025	.0866	12
14	Top15{TextRank{3}}	.0829	.0807	.0872	.0819	.0807	.0832	**15**
15	All{NearestNeighbor}	.0361	.0358	.0385	.0348	.0336	.0362	15.8
16	All{CandidateInText}	.0173	.0088	.6157	.0173	.0088	.6153	1011.6
17	All	.0153	.0077	.6274	.0153	.0077	.6269	1170.3
18	Random15	.0117	.012	.0115	.0121	.012	.0123	**15**

Table 5.10.: Simulation Results for SemEval-2 Combined

difference, however, exist between our results and those of the teams participating in the public competition. While the best team only achieved a F1 score of .3115, we were able to achieve a F1 score of .4235 (#2) when using a variable number of keyphrases for each document, based on selection thresholds. When selecting a fixed amount of 15 keyphrases, as all other contestant did, our best model still achieves a F1 score of .4033 and thus still takes the lead (#3). For this dataset, we were also able to find the only instance of a reinforcement learning application (#5) that significantly outperformed its learning-to-rank counterpart (#7). However, as we will see, this unfortunately cannot be repeated on the reader-only dataset.

Comparable results can be observed from Table 5.11 for the reader-only datasets. Surprisingly, the best published result for this task only got a F1 score of .2382, while we still were able to score relatively high at .3899 (#2) and .3567 (#5), respectively. A possible explanation for this is that models were trained by other participants that were primarily focused on retrieving the author-assigned keyphrases from the beginning of the research papers, and with those missing from the gold standard for the reader-only task, many author-assigned keyphrases were still retrieved but not regarded as correct with respect to the gold standard. Our models, on the other hand, were individually trained for both tasks and thus were able to better focus on relevant keyphrases in the reader-only task.

To confirm those results for our best models, we additionally transformed their predictions for the test sets into the competition's format and executed the supplied evaluation scripts. And indeed, for our gradient boosted regression trees model achieving a F1 score of .4033 on the combined set with a fixed number of 15 keyphrases, we obtained an F1 score of .4053 from the competition's evaluation script. As mentioned before, such difference typically stem from different normalization methods for comparison of keyphrases and in this case the difference is by no means significant. Also for the reader-only dataset,

#	Model	F1	Prec	Rec	$F1_{MI}$	$Prec_{MI}$	Rec_{MI}	#Sel.
1	*All{ExpectedInText}*	.8758	1	.7911	.8836	1	.7915	9.5
2	F1{GBRT{1000,0.01,0.25,10,10}}	**.3899**	**.4866**	.3347	**.3937**	**.4813**	.3331	8.3
3	F1{RF{20,0.8,10,10}}	**.383**	.4761	.3306	**.3845**	.4694	.3256	8.3
4	F1{RegTree{8, minObs=10}}	.3667	.4534	.3185	.3662	.4427	.3123	8.5
5	Top15{GBRT{1000,0.01,0.05,10,10}}	.3567	.3213	**.4089**	.3568	.3213	**.4012**	15
6	F1{M5Rules}	.3499	.3934	.3286	.3499	.3781	.3256	10.3
7	SRL{RegTree{8, minObs=10}, 0.2, 0.99^{t-1}-Greedy, up=100}{2000}	.3466	.4146	.3056	.3462	.4079	.3007	8.9
8	F1{M5P}	.1833	.2255	.1589	.187	.2313	.157	8.2
9	Maui{15, 2, 0.01, 0.005, 5}	.1486	.1327	.1718	.1475	.1327	.1661	15
10	Top15{Tf*Idf}	.1149	.1027	.1327	.1142	.1027	.1287	15
11	First13	.1123	.1069	.12	.1114	.1069	.1163	13
12	F1{PaceRegression}	.1013	.0795	.1429	.1013	.0795	.1395	**21**
13	Top15{Tf}	.0877	.078	.1019	.0869	.078	.098	15
14	Top15{TextRank{3}}	.0688	.0613	.0799	.068	.0613	.0764	15
15	F1{RLinear}	.0658	.0682	.065	.0651	.0682	.0623	11
16	All{NearestNeighbor}	.025	.0246	.026	.0243	.0237	.0249	13.1
17	All{CandidateInText}	.0142	.0072	.6035	.0142	.0072	.6013	1005.4
18	All	.0133	.0067	.6099	.0133	.0067	.608	1085.1
19	Random13	.0084	.0077	.0094	.008	.0077	.0083	13

Table 5.11.: Simulation Results for SemEval-2 Reader-Only

the difference between an F1 score of .3567 measured by us and an F1 score of .3536 measured by the evaluation is close to zero.

Finally, as before, we present the distributions over feature importance of each task's best model in Table 5.12.

#	Feature	Importance
1	nthKeyword	.1503
2	keywordDocFreq	.1131
3	trainingUsesSimilarityMax	.0904
4	tfRelStem2	.0634
5	keywordness	.0618
6	keywordLength	.0454
7	occSpanStemmedRel	.0446
8	trainingUsesSimilarityMean	.0407
9	trainingNotUsesSimilarityMax	.0357
10	textLength	.0352
11	firstOccStemmedRel	.0303
12	lastOccStemmedRel	.0272
13	tfRelStem	.0262
14	trainingUsesSimilaritySum	.0253
15	trainingNotUsesSimilarityMean	.0252
16	tfIdf	.0177
17	tfIdfRel	.0173
18	firstOccRel	.0157
19	synsetSensesTFSum	.0151
20	tfStem	.0146
21	keywordUsage	.0138
22	synsetInformationContent	.0119
23	keywordTokens	.0114
24	synsetSensesTFMean	.0114
25	corpusENlemma	.0104

#	Feature	Importance
1	nthKeyword	.1246
2	keywordDocFreq	.1182
3	trainingUsesSimilarityMax	.117
4	keywordness	.0787
5	tfRelStem2	.0753
6	keywordLength	.0459
7	trainingNotUsesSimilarityMax	.0443
8	textLength	.0383
9	occSpanStemmedRel	.0381
10	firstOccStemmedRel	.0329
11	trainingUsesSimilarityMean	.0314
12	trainingNotUsesSimilarityMean	.0303
13	trainingUsesSimilaritySum	.028
14	lastOccStemmedRel	.0271
15	tfIdfRel	.0166
16	synsetInformationContent	.0148
17	tfIdf	.0143
18	firstOccRel	.0136
19	tfRelStem	.0124
20	tfCorpusLemma	.0123
21	synsetSensesTFMean	.0104
22	synsetSensesTFSum	.0101
23	tfStem	.0097
24	tfCorpus	.0088
25	keywordTokens	.0086

Table 5.12.: Top 25 Features for SemEval-2 Combined (Left) and SemEval-2 Reader-Only (Right)

FAO-780 and NLM-500

In this final section of our quantitative evaluation of keyphrase extraction and assignment, we will present our results for two additional datasets that were used in [Med09] to develop and evaluated Maui. The dataset we will consider first is referred to as FAO-780 and contains a 780 document collection from the UN Food and Agriculture Organization's (FAO) online database[6] on agricultural topics [MW08]. Comparable to the DIPF Web directories, the documents in the FAO online database are also manually annotated by experts. The second dataset from the PhD thesis on Maui we will use is a 500 document collection of medical publications. These documents as well are annotated by professionals at the U.S. National Library of Health[7] and will be referred to as NLM-500.

The statistics in Table 5.13 and Table 5.14 indicate that both share common characteristics in terms of keyphrase lengths and n-gram distributions. The only noteworthy difference is that on average, documents from the NLM-500 dataset are annotated with significantly more keyphrases. Therefore for this dataset, we defined different threshold for minimum and maximum keyphrase selections than for

[6] http://www.fao.org/documents/en/docrep.jsp
[7] http://ii.nlm.nih.gov/

#	Variable	Value
1	Training Documents	779
2	Test Documents	-
3	Language	EN
4	Mean Keyphrases	8 ±3.5 [2, 23]
5	Median Keyphrases	8
6	Mean Keyphrase Characters	8 ±2.8 [2, 18]
7	1-Gram Keyphrases	42.3%
8	2-Gram Keyphrases	53.7%
9	Evaluation	70/30 Split
10	Minimum Selection	2 Keyphrases
11	Maximum Selection	15 Keyphrases
12	Min. Training Freq. for Assignment	5
13	Top n TF Extraction Candidates	200
14	Max. N-Grams for Extraction	2

Table 5.13.: Statistics and Parameters for FAO-780

#	Variable	Value
1	Training Documents	500
2	Test Documents	-
3	Language	EN
4	Mean Keyphrases	14.2 ±4.8 [2, 29]
5	Median Keyphrases	14
6	Mean Keyphrase Characters	8.4 ±3.4 [3, 31]
7	1-Gram Keyphrases	35.8%
8	2-Gram Keyphrases	44.4%
9	Evaluation	10-Fold Cross-Validation
10	Minimum Selection	6 Keyphrases
11	Maximum Selection	20 Keyphrases
12	Min. Training Freq. for Assignment	5
13	Top n TF Extraction Candidates	300
14	Max. N-Grams for Extraction	2

Table 5.14.: Statistics and Parameters for NLM-500

FAO-780. Additionally, as we will see, tf*idf weights provide a better baseline and candidate selection approach to FAO-780 than to NLM-500, which is why for FAO we don't need to extract as many candidates from the documents.

For FAO-780 and NLM-500, the results of the quantitative evaluations are presented in Table 5.15 and Table 5.16, respectively. The ordering of the results is comparable to those already observed for the previous tasks, with all our learning-to-rank models, except for PaceRegression in FAO-780, significantly outperforming all baselines and gradient boosted regression trees achieving the best results of all. The only noteworthy difference between the results of both datasets is that for NLM-500, all models perform better with respect to their F1 scores. A possible explanation, which would also confirm findings from

#	Model	F1	Prec	Rec	$F1_{MI}$	$Prec_{MI}$	Rec_{MI}	#Sel.
1	*All{ExpectedInText}*	.8471	1	.7643	.8634	1	.7596	5.8
2	F1{GBRT{1000,0.01,0.05,10,10}}	**.3725**	**.4352**	**.3738**	**.3841**	**.4201**	**.3538**	6.5
3	F1{RF{20,0.8,10,10}}	**.3677**	.4207	**.3821**	**.3771**	.3955	**.3604**	7
4	F1{M5P}	.3361	.3517	.3658	.3457	.3467	.3448	7.6
5	F1{RegTree{8, minObs=10}}	.3075	.3456	.3341	.3156	.3151	.316	7.7
6	Maui{8, 2, 0.01, 0.005, 5}	.2868	.2833	.3214	.2894	.2833	.2959	8
7	F1{PaceRegression}	.2668	.2375	.3558	.2758	.2331	.3377	**11.1**
8	All{NearestNeighbor}	.2216	.2242	.2465	.2363	.2248	.249	8.5
9	Top8{Tf}	.1398	.139	.155	.142	.139	.1452	8
10	First8	.0908	.0898	.1022	.0917	.0898	.0938	8
11	Top8{Tf*Idf}	.0908	.0898	.1022	.0917	.0898	.0938	8
12	All{CandidateInText}	.0326	.0168	.6288	.0328	.0169	.6154	279.6
13	All	.0252	.0129	.7333	.0253	.0129	.7233	430.8
14	Random9	.0167	.0159	.0193	.0171	.0159	.0186	9

Table 5.15.: Simulation Results for FAO-780

#	Model	F1	Prec	Rec	$F1_{MI}$	$Prec_{MI}$	Rec_{MI}	#Sel.
1	*All{ExpectedInText}*	.6265	1	.4733	.6365	1	.4669	6.6
2	F1{GBRT{1000,0.01,0.05,10,10}}	**.4281**	.49	**.4067**	**.4323**	.4705	**.4003**	12.1
3	F1{RF{20,0.8,10,10}}	.4124	**.509**	.3714	.4182	**.4935**	.3638	10.5
4	F1{RegTree{8, minObs=10}}	.3814	.429	.3683	.3836	.4143	.3587	12.4
5	F1{PaceRegression}	.3262	.2939	.3938	.3319	.293	.3856	**18.8**
6	Maui{12, 2, 0.01, 0.005, 5}	.2623	.2893	.2539	.2648	.2893	.2444	12
7	All{NearestNeighbor}	.222	.2309	.2304	.2207	.2187	.223	14.5
8	Top14{Tf}	.0806	.0801	.0869	.0796	.0801	.0792	14
9	First15	.0742	.0713	.0827	.0733	.0713	.0754	15
10	Top14{Tf*Idf}	.0731	.0726	.0787	.072	.0726	.0716	14
11	All	.0342	.0176	.6746	.0342	.0176	.6771	549.2
12	All{CandidateInText}	.0332	.0174	.3922	.0334	.0175	.3864	314.5
13	Random15	.0138	.014	.0142	.0143	.014	.0147	15

Table 5.16.: Simulation Results for NLM-500

other datasets, is that more gold standard keyphrases on average represent more significant selections which are also more likely to match in terms of precision.

Finally, again the distributions over relevant features for both datasets are provided in Table 5.17.

#	Feature	Importance
1	trainingUsesSimilarityMean	.1332
2	trainingUsesSimilarityMax	.1173
3	trainingNotUsesSimilarityMax	.0675
4	trainingUsesSimilaritySum	.0526
5	tfRelStem2	.0519
6	nthKeyword	.0455
7	trainingNotUsesSimilarityMean	.0429
8	tfCorpusLemma	.0413
9	synsetSensesTFMean	.04
10	tfRelLemma	.0388
11	tfIdfRel	.0379
12	synsetInformationContent	.0351
13	textLength	.034
14	keywordUsage	.0309
15	firstOccRel	.0299
16	tfCorpus	.0252
17	firstOccStemmedRel	.0243
18	occSpanStemmedRel	.0219
19	synsetSensesTFSum	.0202
20	tfStem2	.0196
21	keywordLength	.0194
22	lastOccStemmedRel	.0183
23	corpusENlemma	.0147
24	tfRelStem	.0128
25	tfStem	.0086

#	Feature	Importance
1	trainingUsesSimilarityMax	.1759
2	trainingUsesSimilarityMean	.1446
3	nthKeyword	.0777
4	trainingUsesSimilaritySum	.0743
5	trainingNotUsesSimilarityMax	.0697
6	trainingNotUsesSimilarityMean	.0584
7	textLength	.0474
8	keywordUsage	.0352
9	synsetSensesTFMean	.031
10	tfIdfRel	.0309
11	keywordLength	.0292
12	synsetInformationContent	.0266
13	tfCorpusLemma	.0256
14	tfRelStem2	.0255
15	firstOccRel	.0227
16	tfCorpus	.0199
17	tfRelLemma	.019
18	firstOccStemmedRel	.0158
19	synsetSensesTFSum	.0122
20	lastOccStemmedRel	.0107
21	occSpanStemmedRel	.0102
22	corpusENlemma	.0084
23	tfStem2	.0058
24	keywordTokens	.0054
25	tfRelStem	.0052

Table 5.17.: Top 25 Features for FAO-780 (Left) and NLM-500 (Right)

5.5 Qualitative Evaluation

5.5.1 Setup

The qualitative evaluation for our keyphrase extraction and assignment approach to the DIPF Eduserver dataset is designed in a similar fashion as the qualitative evaluation for topical crawling was in Section 4.5. Being not as error-prone as other natural language processing tasks with a higher complexity, less tolerance towards noisy input data, or a higher potential for problems with their natural-language output, such as in automatic summarization, we again hope to achieve results which cannot immediately be identified by a less experienced reader as automatically retrieved. To evaluate this claim, we again perform a blind study, where we will compare the ratings DIPF authors assign to our results to the authors' ratings for content from their own database. As a side effect, we are also able to better evaluate the quality of the gold standard, which has severe implications for our ability to automatically learn making meaningful decisions from it.

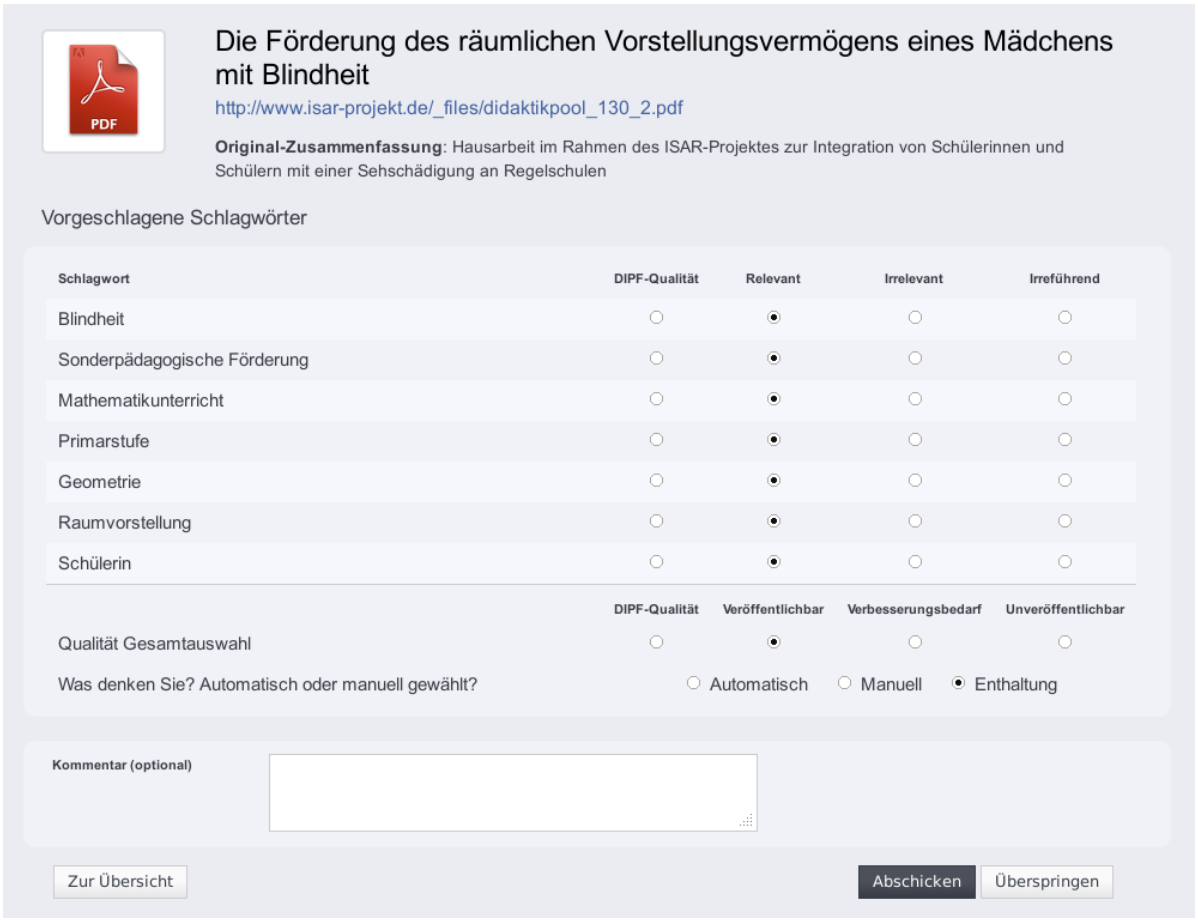

Figure 5.1.: Screenshot of a Random Keyphrase Assignment Example

When accessing the keyphrase evaluation webpages, the user is presented a set of keyphrases for a link which was randomly drawn from the DIPF Eduserver database, as depicted in Figure 5.1. Again, with an equal probability for both cases, the presented keyphrase set is either the gold standard for the respective document, or the predicted set, as retrieved by our best algorithm. Similar to our evaluation of topical crawling, our immediate interest is to find out how individual keyphrases are perceived, and if the ratio of relevant keyphrases is equal to or higher than the precision measured in the quantitative evaluation. However, different from crawling, only dealing with a small amount of proposals per database entry, we are also able to ask for an evaluation of the selection's quality as a whole. While the single keyphrases' quality roughly corresponds to precision, the entire set's rating can rather be regarded as also considering its "recall", since comprehensiveness is one of the key aspects of a keyphrase set's quality.

	No Value	-	-	**Internal Quality**
Individual Words	Misleading	Irrelevant	Relevant	DIPF Quality
Keyphrase Sets	Unpublishable	Need for Improvement	Publishable	DIPF Quality

Figure 5.2.: Rating Scheme for Individual Keyphrases and Keyphrase Sets

Finally, we also asked for the DIPF authors' estimate regarding whether the presented keyphrase selection was taken from the Eduserver database or retrieved by our algorithms. As it was noted by some of the authors, for most keyphrase sets created by themselves, they would be able to re-identify them or even recognize that their set wasn't showing if they remembered the presented link, and therefore a bias

would be introduced to the results for this question. Therefore, we marked the answer to this question as optional, hoping the authors would only give their estimates if they did not remember the specific link from their Web directory.

This point also sheds light on another potential thread to the significance of some results. If from the entire set of presented keyphrases, a participant of the evaluation would be able to be certain about the keyphrases' origin, he/she may, consciously or subconsciously, rate individual keyphrases, as well as the set, differently than without this information. This is either the case when, as mentioned before, the DIPF author remembers the correct keyphrases from the Eduserver dataset, or when one of the keyphrases is obviously misplaced and therefore the automatic nature of the set is clear. As a consequence, we would have to present keyphrases individually in order to be certain about preventing a bias on part of the user. However, we still decided to keep the described setup as measuring "recall" qualitatively is equally important and a design change might make the interface more complicated and thus more likely to demotivate the user. Furthermore, we had already received over 400 ratings when this issue became apparent, and an obvious bias in the numbers luckily was not visible, as we will see in the discussion of the results.

The automatic keyphrase sets for this evaluation were retrieved from training our best algorithm for this dataset — a gradient boosted regression trees model in the learning-to-rank framework — on 50% of the training data and applying the model on the remaining 50%.

5.5.2 Discussion

In Figure 5.3, the empirical distributions over quality ratings are given for all single keyphrases. For being able to compare automatically retrieved keyphrases with the gold standard, for each mark on the quality rating axis, the relative frequency is presented for each source individually. Due to the gold standard containing more keyphrases for each document on average, we are not able to compare absolute counts. Instead, the frequencies are normalized for each source, meaning that, for example, we can see from the bar chart that more than 10% of all automatically assigned keyphrases is deemed "irrelevant", while less than 5% of all DIPF Eduserver keyphrases fall into this category.

The probably most surprising information from a first glance at this figure is that with 41%, the ratio of "DIPF-quality" keyphrases is larger for the retrieved keyphrases than for the Eduserver keyphrases themselves, only achieving a ratio of 39%. The results, however, become more plausible when considering the other 3 ratings, since they show that across all categories, the gold standard still performs significantly better than our automatic approach. This can also be witnessed in Figure 5.4, which shows similar empirical distributions after both indicators for relevance, as well as both indicators for irrelevance, have been merged into one rating class each. Still, with almost 80%, the qualitatively observed precision is significantly higher than the quantitative precision of 39%. Furthermore, it can be argued that with models trained with a larger weight on precision than recall, the ratio of irrelevant keyphrases, which currently is little over 20%, could be reduced even more.

While in Figure 5.3, the automatic keyphrases present a pattern of either being relevant or irrelevant, there is a significant distinction between "relevant" and "DIPF quality" keyphrases for the gold standard. However, other than expected for a gold standard, with 53% the absolute majority of keyphrases are only rated as "relevant", thus supposedly not meeting the expectations with respect to the institute's internal quality standards. Furthermore, a surprising 8% of all gold standard keyphrases are either irrelevant or even misleading, leaving only 39% of "perfect" gold standard keyphrases. This observation, of course, has strong implications on the quality of the results that can be expected from learning from such a data set: A perfect adaption of the latent policies underlying the gold standard selection could never achieve

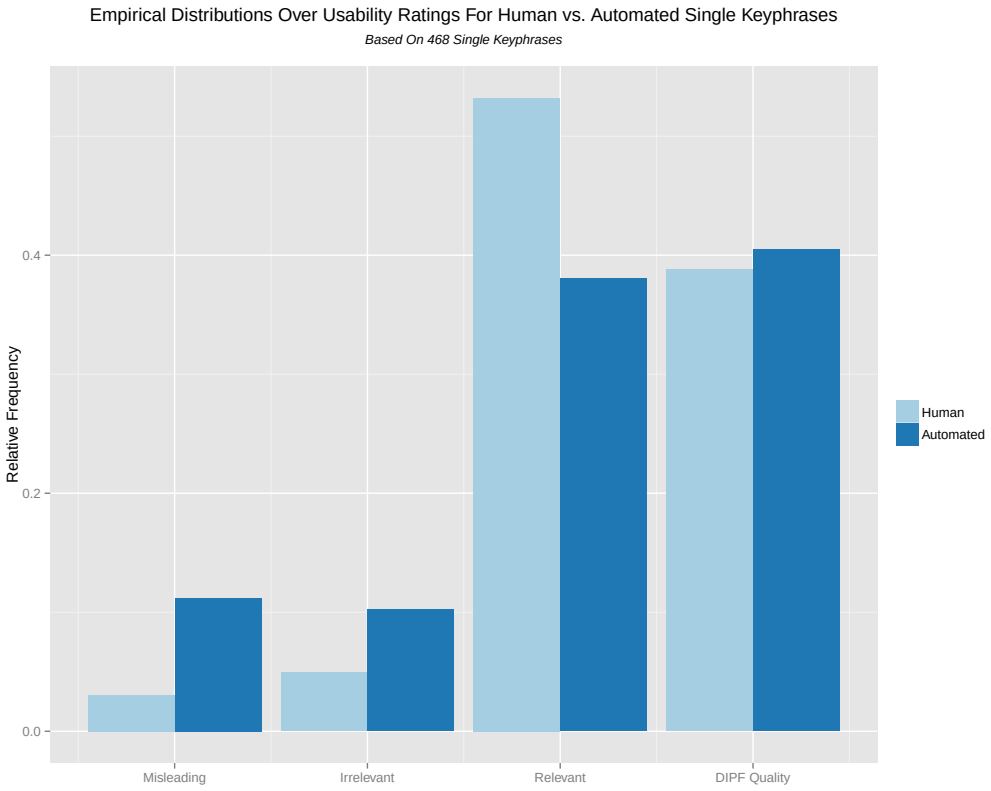

Figure 5.3.: Distributions Over Relevance Ratings for DIPF vs. Retrieved Keyphrases

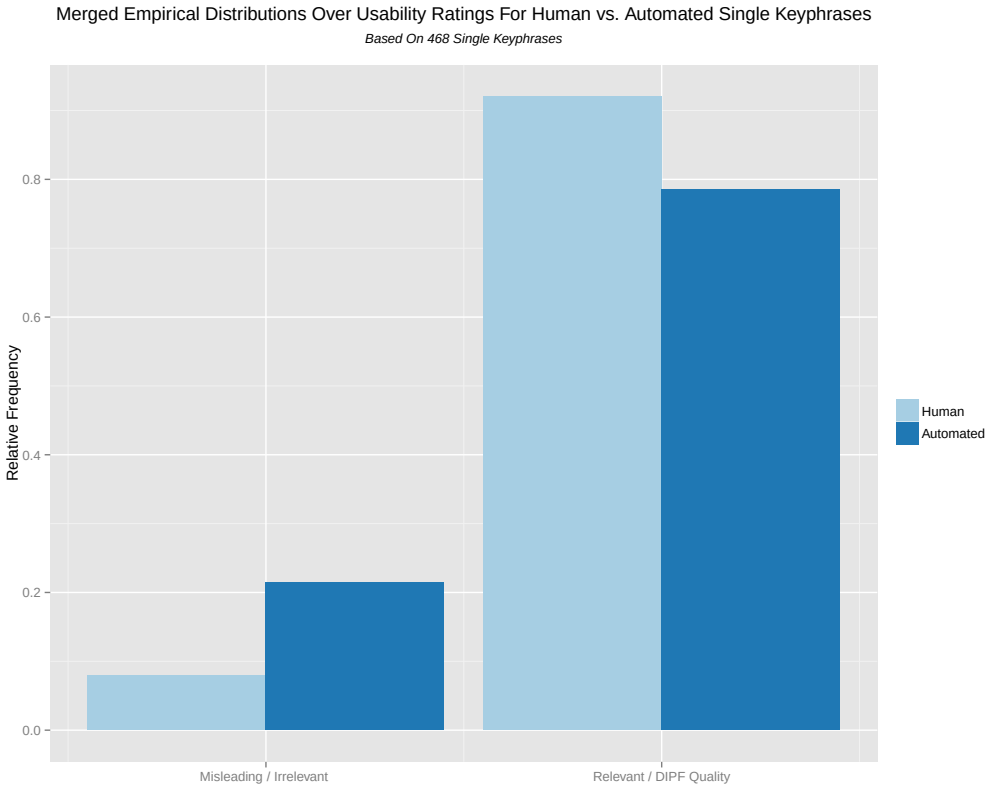

Figure 5.4.: Merged Distributions Over Relevance Ratings for DIPF vs. Retrieved Keyphrases

optimal results, and furthermore, patterns observable from such a dataset can be expected to be more diffuse than for higher quality datasets and therefore more difficult to approximate.

Figure 5.5 shows the empirical distributions over ratings for entire keyphrase sets, computed and presented in the same fashion as the single keyphrase ratings were. First of all, the aforementioned problems with the gold standard can again be observed, since the majority of sets is only rated as "publishable" or even in need for modifications, leaving only 26% of all Eduserver keyphrase sets being rated as achieving DIPF quality. Even more than before, the difference in the average quality between retrieved sets and gold standard is apparent, as it can also be seen in Figure 5.6, which again provides a visualization using merged ratings.

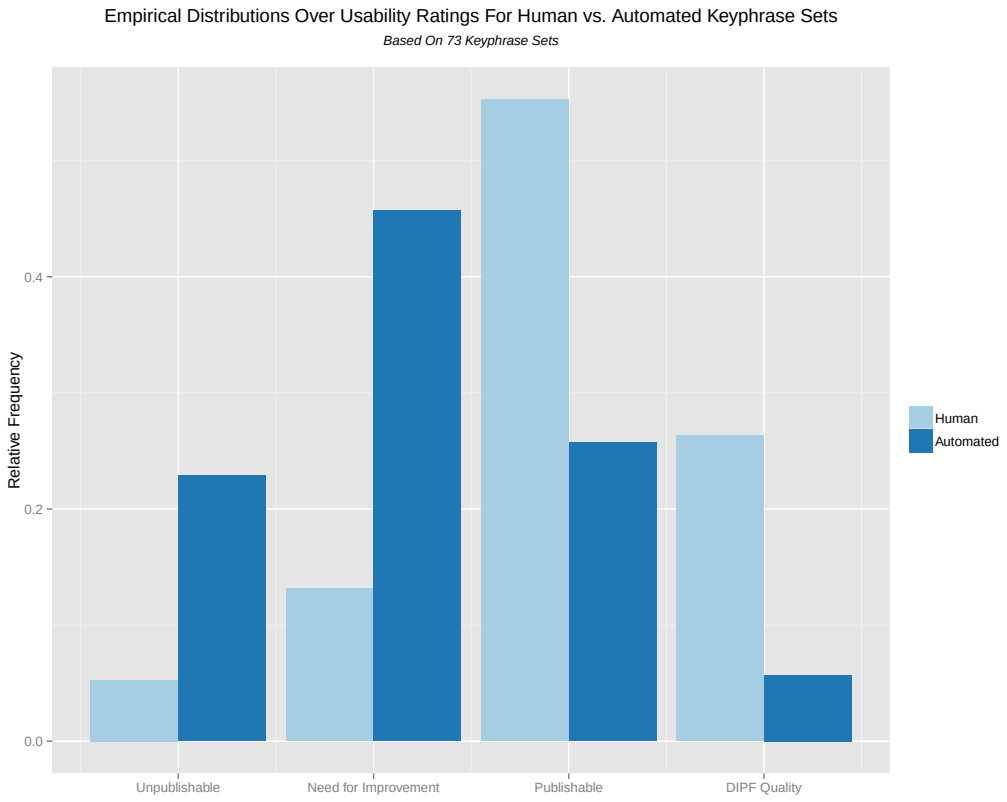

Figure 5.5.: Distributions Over Usability Ratings For Human vs. Automated Keyphrase Sets

One explanation for these results is that again, with a training dataset not meeting the authors' own expectations for the majority of examples, it is very difficult to automatically learn to retrieve high-quality sets of keyphrases. Additionally, it can be argued that the expectations for entire sets are much higher, while at the same time it is more difficult to obtain high quality results. With our approach being focused on optimizing single-keyphrase selections for a high precision, model decisions only had to be made for independent items and when in doubt, a high enough threshold on the expected keyphrase likelihood could prevent many false positives. Therefore, it is more likely to satisfy the reader when single keyphrases are regarded in isolation. For qualitative set evaluations, however, the expectation of the reader is that enough keyphrases will be presented to cover all relevant topical aspects of a document — something which is much more difficult to model precisely. Furthermore, in the perception of a user of our evaluation tool, even with a perfect topical recall and 9 of 10 keyphrases being highly relevant, just a single misplaced keyphrase could render the entire set as in need for improvement. As a result, the qualitative ratings for keyphrase sets therefore best correspond with the F1 score in quantitative evaluation.

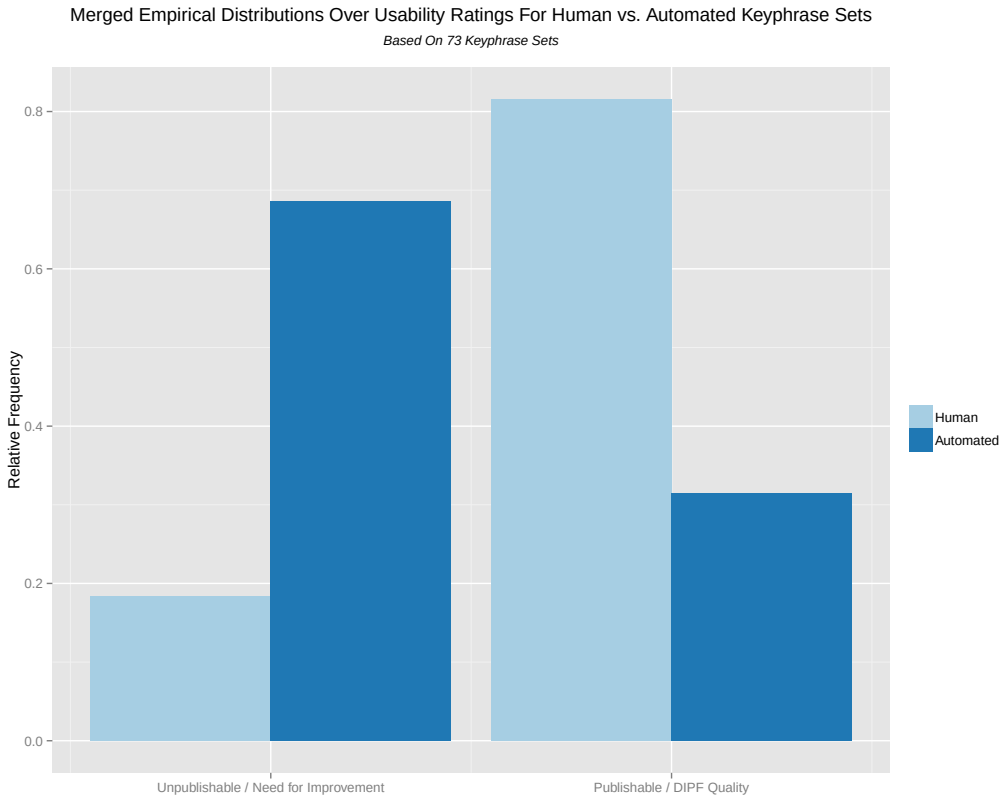

Figure 5.6.: Merged Distributions Over Usability Ratings For Human vs. Automated Keyphrase Sets

Our interpretation of the qualitative results is that for individual keyphrases, the approach works very well — even better than it could already be expected from the results of the quantitative evaluation. For the quality of an entire keyphrase set, however, multiple risks exist, such as an insufficient characterization of the respective document, or single keyphrases being inappropriate and therefore the set as a whole not publishable without limitations. As a consequence, our approach can be used for very large datasets where manually assigning keyphrases would be infeasible and a certain error tolerance or even a willingness for manual corrections can be expected on part of the target audience. For high quality Web directories, such as the DIPF Eduserver, however, a more suitable application would be one in a semi-automated workflow, where an automatic selection is presented to a DIPF author and he then applies manual corrections to it. Our hope is that this way, the manual effort would be reduced significantly as in most cases a human author would just have to spot obviously false or irrelevant information, instead of having to read parts of the document for identifying relevant keyphrases on his own from scratch. Such a claim, however, would have to be evaluated in a further user study.

In our final question to the user of our evaluation tool, we wanted to know whether he assumed that the keyphrase set was retrieved automatically or taken from a gold standard. With the limitations of entire keyphrase sets already identified in the previous paragraphs, it is obvious that our algorithms would not yet pass a Turing test for this task. Still, in Figure 5.7, it can be seen that even for keyphrase sets as a whole, it is not always as easy to differentiate between manually and automatically provided information. On the horizontal axis, the ratings are separated between keyphrase sets automatically retrieved and those taken from the gold standard; on the vertical axis, we see how many instances of each set were expected to be generated and how many not. For the gold standard keyphrase sets, aforementioned problems again become visible, as almost half of all gold standard sets were assumed to be provided automatically. For the retrieved sets, the distinction is clearer. Still, only 65% of automatically retrieved keyphrase sets were correctly identified as such, while 35% were assumed to be part of the DIPF Eduserver dataset — a distribution which roughly corresponds to the ratios of publishable and unpublishable ratings for those sets.

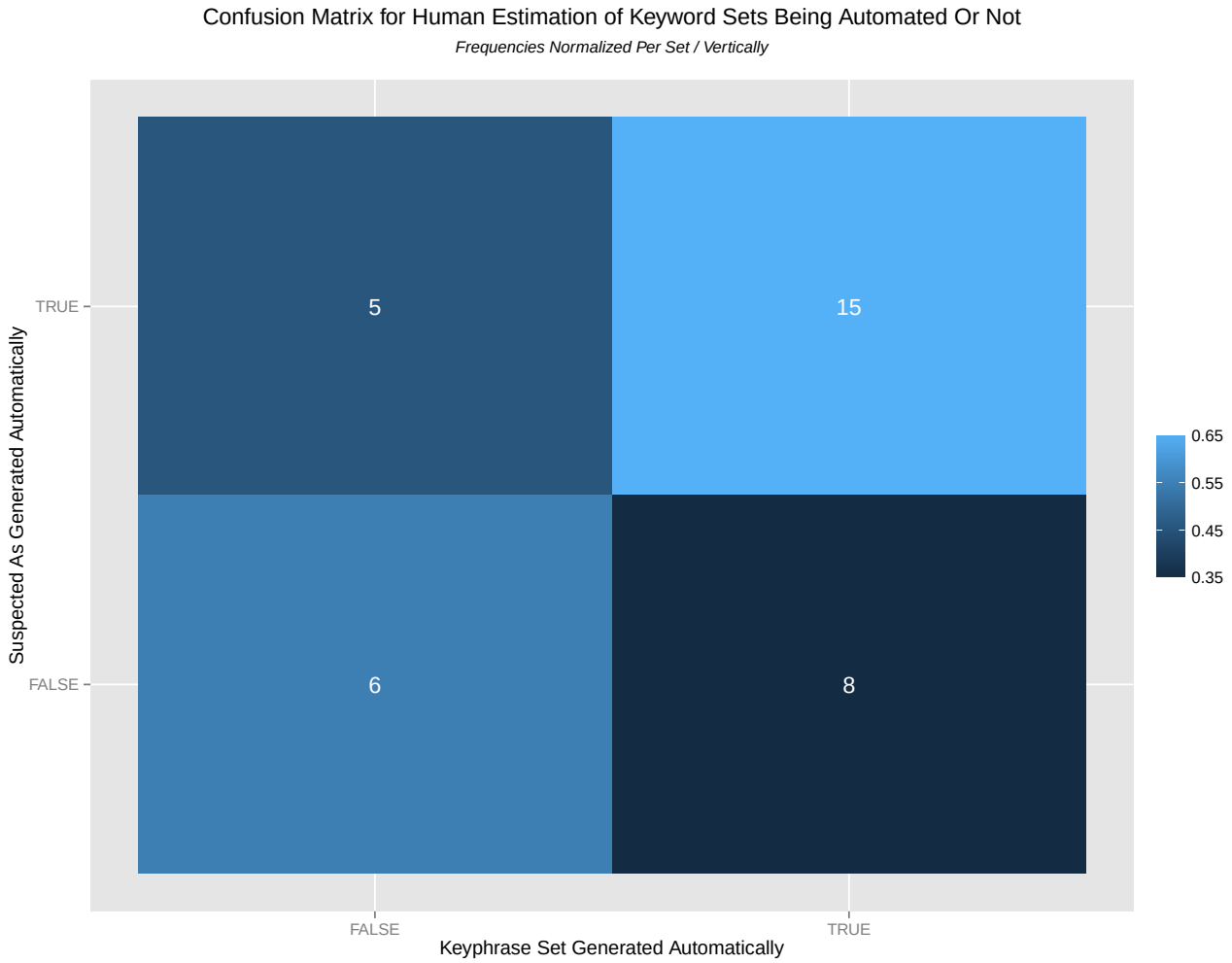

Figure 5.7.: Confusion Matrix for Human Estimation of Keyword Sets Being Automated Or Not

5.6 Review

In this chapter, we have presented our approach to automatic keyphrase retrieval based on a combination of extraction and assignment methods, applied to the learning-to-rank framework. For each document from a dataset, we identified relatively large quantities of keyphrase candidates, stemming from both, the document itself, as well as the training dataset. We then trained supervised learning models by characterizing document-candidate pairs by various features and providing a label, indicating whether the candidate was also present in the gold standard for the respective document or not. As a result, we obtained predictors of a candidate's likelihood for being a gold standard keyphrase solely based on its features for the respective document. This likelihood could then be used to rank candidates and also for filtering them based on a likelihood threshold automatically derived from the training data.

We then applied our approach to the DIPF Eduserver database, as well as to other publicly available datasets, for which we not only could compare our algorithms to open-source baselines, but also results found in keyphrase assignment publications. Throughout all datasets, we not only achieved quantitative results that were within expectations for the respective datasets and performance metrics; we also significantly outperformed all baseline approaches. Futhermore, we also emulated the setup of a recent workshop competition on keyphrase assignment, and achieved test set scores which significantly outperformed the published results of all participants.

For the DIPF Eduserver dataset, we furthermore conducted a qualitative evaluation with the help of the DIPF authors responsible for curating the Eduserver. This evaluation showed that for single keyphrases, almost 80% of all information retrieved by our algorithms is correct, which is significantly higher than the precision of 39% suggested by the quantitative evaluation. In extension to other datasets achieving an even higher precision in the quantitative evaluation, it can be assumed that for high-quality datasets with a very specific terminology, close-to-human performance can be achieved in determining whether a keyphrase is relevant or not. For automatically retrieved keyphrase sets as a whole, however, we have seen that a significant majority would not be qualitative enough for immediately being presented on Web directories such as the DIPF Eduserver. In our analysis, we formulated the hypothesis that this predominately is the result of either too few keyphrases, or single keyphrases being irrelevant. Since when increasing precision through higher thresholds, recall tends to decrease due to even less presented keyphrases, this problem cannot be solved immediately without even more accurate algorithms available. Instead, we propose that the developed approach should be used as the basis for a manual revision by a human author, thereby hopefully increasing the author's productivity.

Our major contributions to keyphrase assignment research are the specific setup and components of our supervised learning-to-rank approach, as well as the extensive study across multiple datasets, including a qualitative evaluation. Instead of following the standard approach of training a binary classifier, we defined a framework in which any continuous scoring function may be used for rating training examples, which also supports inducing more powerful ranking functions to be trained. Based on optimizing the results for any objective function, we also showed that meta parameters such as thresholds on the base model outputs can be derived automatically within this framework. Furthermore, we identified a large set of powerful features, which, as a whole, and sometimes even for individual features, cannot be found in other publications. Additionally, we introduced complex models based on ensemble of trees, which typically perform well across many task, and often outperform less complex models used in other publications, such as linear/logistic regression, simple classification trees, or support vector machines. The significance of those findings is supported by the observation that even with weaker models applied on our features, we still significantly outperform any baseline approach. When using more complex supervised learning models, a further significant improvement can be observed across all datasets used

in our quantitative evaluation. Finally, our qualitative study provides further insight into the effects of data quality and parameter tuning, and may serve as inspiration for further qualitative studies or the development of even more significant quantitative metrics as precision and recall are.

6 Single-Document Summarization

6.1 Problem Description

Similar to keyphrases, short summaries allow the readers to quickly identify the topics, focus, and intention of a document, which further enables them to decide on its relevance without even having to start reading the document itself. Even more than keyphrases, using full natural language sentences, short summaries establish a context by putting topical keywords into relation and thus can be more informative. However, the comparative benefits of summaries come with the significant increase in effort required to formulate a high-quality summary. For the definition of acceptable keyphrases, it can be sufficient for an experienced annotator to quickly scan over the title and first sentences of a document to roughly identify the main topics; but with summarization, the readers are used to much more precise formulations and therefore an author of a summary has to have a much more assured and thorough understanding of the contents to avoid false assertions.

When manually writing summaries, even more than with keyphrases — for which we already identified a significant portion of meta or abstractive terms in real-life datasets —, abstracting over content is crucial for a compact but comprehensive description. As a consequence, the summary writing process typically does not involve the mere quoting of sentences from the text, but instead reading it, understanding it, and writing a novel abstract about it. Therefore, any attempt at directly modeling and approximating the human summary writing process would require the content to be analyzed automatically, topics and their relation to be identified, and natural language sentences to be created, presenting the findings. This is especially difficult as natural language is manifold and complex in grammar, which means that even if it is was possible to sufficiently describe the content in a more formal language, with today's technologies, it is nearly impossible to generate natural language summaries comparable to human summaries in terms of content, precision, and linguistic style and variance for a broad range of topical domains. Luckily, this does not mean that any attempt at automatically creating summaries would still be in vain. For purposes where text understanding is the central goal, such as in Web directories, and presentation, different from traditional publications like news papers, only plays a secondary role, it might be sufficient to automatically identify the most representative sentences for a text as a substitute summary. Furthermore, in the primary formats present in the DIPF Eduserver dataset — Web pages, papers, and reports — it is common for the authors to include short summaries on their own, whose presence would then only require an algorithm to correctly identify them for extracting quality summaries. Therefore, in this chapter, we present an extractive summarization approach which tries to avoid the many sources of error associated with abstractive summarization and instead seeks to find sentences suitable for a standalone presentation to the reader. It is not our immediate goal to achieve summaries being able to compete with human-written counterparts on any quality aspects, but to either provide a valuable starting point for human authors, or to provide additional content indicators where writing human summaries is expensive or infeasible and therefore otherwise would not take place — such as in large Web directories.

Comparable to the previous chapters, we will first give an oversight over state-of-the-art technology in automatic summarization in Section 6.2. Then, we will describe our two approaches to extractive summarization, one again following the learning-to-rank idea, one using the reinforcement learning framework, in Section 6.3. Finally, we present both quantitative and qualitative evaluations of our algorithms on the DIPF Eduserver dataset in Sections 6.4 and 6.5.

6.2 Related Work

As for keyphrase retrieval, the research on automatic summarization can mainly be discriminated by their use of supervised or unsupervised learning methods. Again, unsupervised learning algorithms typically try to identify the role a sentence has for a document's internal structures, while supervised learning approaches typically assign features to candidates and try use those for predicting sentence relevance.

One of the earliest unsupervised summarization algorithms was proposed by Barzilay et al., who computed lexical chains for documents and then used multiple heuristics for finding sentences which best represented central lexical chains [BE+97]. In 2001, the idea of sentence centrality was further explored by Gong et al., who used cosine similarities between sentences and documents, as well as Latent Semantic Analysis to define sentence salience scores [GL01]. Comparable to their research on keyphrase extraction, Erkan et al. and Mihalcea et al. both published results on graph-based salience measures in 2004. The work of Erkan et al. lead to LexRank, which used PageRank for obtaining vertice weights based on cosine similarity graphs [ER04]. Mihalcea et al. also used cosine similarity graphs for experimenting with PageRank and HITS for graph centrality computations in TextRank, which therefore follows the same basic principles of LexRank, but differs in a few details [Mih04].

Also similar to keyphrase extraction, supervised learning methods for extractive summarization started with few features on rather simple statistical machine learningmodels and then gradually improved on both aspects. For instance, in 2002 Osborne used a maximum entropy classifier on features such as sentence length, sentence position, and indicators for the presence of certain n-grams within sentences [Osb02]. In 2007, Svore et al. added features specifically tailored to news articles, such as the overlap between a candidate sentences and certain regions such as the title, as well as information extracted from Wikipedia, for the pairwise ranking algorithm RankNet [SVB07]. In the same year, Shen et al. used conditional random field models on basic features such as similarity to neighbor sentences, as well as results from unsupervised learning, such as HITS scores [SSL+07]. Furthermore, other approaches to semi-supervised learning, i.e., making use of labeled as well as unlabeled data, have been studied, for example, by Amini et al. [AG02] and Wong et al. [WWL08]. It is noteworthy that all examples except for Svore et al. use binary classification, which makes it necessary to manually annotate each training data candidate with regards to being relevant for a summary or not.

Different from keyphrase extraction, for which we could see that basic reinforcement learning approaches were not able to outperform traditional supervised learning models, there already exists research on reinforcement learning for automatic summarization. In 2012, Ryang and Abekawa proposed a reinforcement learning for automatic summarization, in which states are defined by the set of currently selected sentences, and actions are allowed to either insert further sentences or to finish the process [RA]. As learning feedback, penalties were assigned for finishing with too few sentences, and rewards were given based on a score function for the final summary after finishing properly. For learning the expected long-term rewards of actions, transitions between states were characterized by feature vectors. Those features included basic information such as a sentence's length and position, but also coverage information regarding the top 100 document terms, ranked by tf*idf.

6.3 Design

6.3.1 Overview

The design for our approach to automating extractive single-document summaries follows the idea of a ranking derived from a point-wise candidate scoring approximation, similar to what was already described for keyphrase extraction in Section 5.3. In this chapter, however, we will not only explore point-wise document-candidate relations in our learning-to-rank framework, but also apply reinforcement learning techniques for being able to consider the context of sentences already selected for our summaries. Considering context and modeling a subsequent decision making process, we hope to better approximate design actions made by humans and to improve accuracy over the learning-to-rank baseline.

To recapitulate, in the learning-to-rank or candidate selection framework, for each document a set of candidates is retrieved, and then, for each document-candidate pair a score is assigned which expresses the relevance of the candidate and implies a ranking of all candidates. For single-document summarization, those candidates are sentences found in the documents, and the final selection is a subset of those candidates, which, under some circumstances, could even be joined to a continuous text. This is either the case when only adjoining sentences are extracted from the text that therefore form a unit consistent in content and grammar, or, in an extension of the approach, templates and heuristics exist for joining independent sentences. However, as mentioned before, our immediate intention in this chapter is to being able to identify relevant sentences from the often large documents of the DIPF Eduserver with their fair share of noise and content entirely irrelevant for summaries.

6.3.2 Preprocessing

Preprocessing steps as described in Chapter 2 have been important throughout this entire thesis, since analyzing content always requires the central information to be available in a mostly unambiguous and error-free representation. Still, with documents of several dozens to multiple hundred pages, errors in single sentences or tokens were still tolerable for tasks such as topical crawling as long as there was still enough error-free content available for estimating a document's content. Summarization, however, focuses on very limited sets of sentences which, as we will see later when discussing our results, for the DIPF Eduserver dataset typically only comprise 3 to 4 sentences in both, our predictions as well as the gold standard summaries. Therefore, representations suffering from severe parsing errors might make it very difficult to extract meaningful summaries from text when the only plausible candidates might not be automatically identifiable as such. Technical considerations aside, even when the best summarizing sentences can be extracted from a text, the results being in a bad format will harm the user's satisfaction with the service.

During the development of our solution to single-document summarization, we identified the most severe threat in the input data to be erroneous results from parsing HTML and PDF files. For HTML files, many content extraction software packages, such as *HTMLUnit* and *JSoup*, within their standard interfaces only offer to extract visible text as a single string with often issues in separating the different content regions of modern Web pages. Therefore, as described in Section 2.2.1, we used the *Boilerpipe* library for being able to better distinguish between textual elements and thus avoiding that, for example, menus might be regarded as a single sentence with the menu items as its token — an obviously improper candidate for a summary. For PDF files, not only the problem of separating regions — for example titles from subtitles — exist, but also often with problematic file encodings even errors on the word-level occur. Mostly, this is the case when the file encoding does not contain words as units, but single

characters and, using heuristics, those characters then have to be joined to words and sentences based on their x-y coordinates for the document. With both major PDF parsers for Java, *PDFBox* and *iText*, we experienced severe issues in their output, sometimes also with erroneous character encodings, so that we had to implement our own content extraction heuristics as a *iText* plugin (see Section 2.2.2). To illustrate the importance of preprocessing, after replacing standard content parsers for HTML and PDF, a metric which measures the correspondence between generated summaries and gold standard summaries increased from 0.22 to over 0.32 for our results.

6.3.3 Candidate Selection

As discussed in the previous sections of this chapter, the idea to our single-document summarization approach is to extract and present those single sentences from a document which appear to be most suitable for indicating the document's content to the reader. Therefore, as it was done in the chapter on keyphrase extraction (Section 5.3.2), as a first step, candidates have to be retrieved from a document, which are then subsequently ranked by a statistical machine learning algorithm. While, however, with keyphrase extraction, we were able to consider large quantities of plausible candidates, e.g. those not being stop words, and to leave it to more sophisticated algorithms to make our choices, with automatic summarization, it typically is required to impose a more strict filtering on the input for obtaining quality results in a reasonable time. For instance, as mentioned before, an inappropriate parsing algorithm might join words from HTML listings to a single sentence despite not forming a proper natural language sentence. As a consequence, when learning to extract summaries based on objective functions which primarily measure the terminological overlap between generated and reference summaries, such "sentences" might not be identified as problematic, but rather selected as sentences for our summaries due to their high density of relevant keywords.

Therefore, our approach to candidate retrieval for single-document summarization comprises multiple steps and thresholds for decreasing computational efforts and filtering ill-formatted content which may be difficult to identify as such by our statistical machine learning models. First, candidate sentences are filtered based on their number of characters and tokens, since even if containing relevant information, too short or too long sentences are typically inappropriate for presenting them to a human reader. Those thresholds are set by automatically optimizing the lead baseline, which selects the first n candidates as its summary output, for our primary objective function, as well as from manual observations on the results. For the DIPF Eduserver dataset, we determined optimal thresholds of a minimum of 100 characters, a maximum of 560 characters, a minimum token count of 7 and a maximum token count of 80. Second, a few patterns are defined which typically serve well as indicators for a sentence's quality as a summary candidate. Those patterns include terms like "Impressum" to be forbidden from appearing in a sentence, and the requirement for an uppercase letter starting the sentence and proper punctuation finishing it. Third, the ratio of characters from the class of upper- and lowercase roman letters and Arabic digits must exceed 60%, as an indicator of proper text in distinction to mathematical formulas and sentences with erroneous character encodings. Finally, we apply our stop words filter to a candidate sentence and require it to have at least one token left after filtering to already prevent nearly information-free sentences at this stage. To further reduce the computation efforts of our statistical machine learning algorithms, we also limit the amount of candidates for a summary to the first 30, as with very long documents, sentences tend to become increasingly specific and therefore generally unsuitable for a summary.

As it was also the case with keyphrase extraction, often candidates of a high relevance and quality can also be retrieved from the document's meta data. For the DIPF Eduserver dataset, we therefore

also include sentences from HTML meta descriptions as long as they surpass the minimum tokens count threshold of 7 and contain non-stopword terms.

6.3.4 Learning-to-Rank Approach

In this chapter, we follow two different supervised learning approaches to automatic summarization; the first being a learning-to-rank approach, while the other follows the reinforcement learning framework. As already discussed throughout this thesis and also in the previous sections of this chapter, the learning-to-rank framework considers all candidates for a document and either tries to approximate a point-wise score indicating the relevance of a document-candidate pair, or it tries to approximate a pairwise score indicating the relevance difference for a candidate-candidate pair.

For keyphrase extraction, but also for summarization, there tend to exist large amounts of candidates and a unconstrained pairing of candidates would lead to a combinatorial explosion in the order of n^2. Therefore, for both tasks, we prefer the point-wise scoring idea over the pairwise approach, leading to the following task, where a function of document-sentence features has to be found which best approximates a score function for document-sentence pairs:

$$\hat{\phi}(features_{document+sentence}) \approx \phi(document, sentence) \tag{6.1}$$

While for keyphrase extraction, the scoring function ϕ was effectively boolean, since from the gold standard, we only knew if a candidate was an expected keyphrase or not, several possibilities for characterizing a sentence's relevance for a summary can be applied. Since for the results of our statistical machine learning approach, the objective function to be optimized is a key driver, the "best" score function always depends on the expectations for said results. In automatic summarization workshops and publications, the *ROUGE* scoring framework, which measures terminological overlap between summaries, has established itself as the main quality indicator. Therefore our main objective function for the quantitative evaluation is a F1-score-like formulation which considers both a "precision" and a "recall" in the sense of ROUGE (see Section 6.4 for more details). For scoring single training examples of sentence candidates, we use the so-called BLEU-1 score of unigram overlap, which measures precision comparable to ROUGE, which actually just is a recall-oriented reformulation of BLEU. As a result, statistical machine learning algorithms making use of those training labels effectively rank sentences by their expected percentage of terminology also being found in a gold-standard summary.

As also discussed later, however, terminological overlap typically is not sufficient for determining two summaries' correspondence, as well as for quantifying the individual summaries' qualities. First, on one hand, only measuring the similarity of the vocabularies often does not say enough about the actual statements in the summaries, since, using the same terms, a statement as well as its negation can be expressed. Second, on the other hand, with an increasing focus on context, such as through the use of n-grams, it becomes increasingly difficult to measure semantical correspondence, since similar statements might be expressed using entirely different terms and expressions. Finally, many other factors such as grammatical correctness, linguistic style, and the correctness and completeness of a summary's content are key factors for its quality, as perceived by the readers. Therefore, as an extension of this work, a more quality-oriented metric with a corresponding scoring function for learning-to-rank could be derived based on holistic ratings for summary sentences, such as we obtained through our qualitative evaluation of automatically generated summaries. As already discussed, our specific learning-to-rank implementation is capable of handling and automatically deriving optimal thresholds for any continuous scoring function.

6.3.5 Reinforcement Learning Approach

Reinforcement learning considers optimizing processes of subsequent decision making, which typically correspond to the search for an optimal strategy in a defined state-action space. To summarization, this framework can applied by considering the current selection of sentences as summary states and adding or removing sentences to the summaries as possible actions. With this formulation of states and actions, we follow the work of Ryang et al. [RA]. By doing so, we extend the idea of trying to estimate single candidate sentences' relevance scores independently to iteratively improving entire summaries by trying to estimate the effects that changes have to their overall score.

As with topical crawling, the strategy of performing actions is driven by a reward function, which, during a training phase, teaches the algorithms to estimate the quality of performed actions. Trying to optimize specific objective functions, such as the F1 combination of *BLEU-N* and *ROUGE-N* metrics, suitable reward functions come intuitively: For each performed action, as a reward, we provide the difference between the score for the former summary s and the score of a summary s' resulting from applying action a as an indicator for the summary's change of quality.

$$reward(s,a,s') = \begin{cases} 0, & a = stop \\ score(s') - score(s), & else \end{cases} \tag{6.2}$$

With a reward of 0 for an action explicitly causing the summarization process to stop, we additionally provide a stopping criterion to the algorithm. A properly trained reinforcement agent will continue adding sentences to our summary as long as it estimates its actions to cause a long-term improvement to our objective function. As soon as all remaining candidate sentences yield a negative expected quality difference for our summary in the making, the selection process is stopped.

For most scoring functions, rewards function derived in a similar fashion to what is presented above tend to become non-linear with respect to the number of selections made. As a result, equally relevant sentences with similar features might be difficult to model using a linear approach since the difference in rewards, solely based on their order, is typically hard to capture without specifically added "technical" features or using kernel methods. For example, the *ROUGE-N* metrics are recall-based measures of terminology overlap, and therefore, the first sentences added to a summary typically yield a much higher increase than later sentences, which might still add valuable nuances to the summary, but have much of their terms already "covered" by earlier sentences. One possible solution to this problem is to provide a linearizing normalization, for example through a logarithmic transformation with respect to the number of sentences already added. Different to many applications of reinforcement learning, however, due to the relatively small amounts of training data for automatic summarization, we are able to use more complex, non-linear supervised learning algorithms for predicting rewards and therefore overcome this issue by the choice of our algorithms.

It is noteworthy that for such difficult tasks as summarization, especially in conjunction with problematic training data, it is near to impossible to find an optimal model of the decision boundary, where every estimated gain translates to an actual improvement and any expected loss actually means that adding a sentence would decrease the quality of the summary. During the training phase, the models might overfit to the data and either become too optimistic and therefore generate summaries with many sentences, or they become too pessimistic and stop early, with the extreme scenario of not adding any sentences since expecting a negative effect for each. Therefore, comparable to the parameters estimated in the learning-to-rank approach, we also introduce minimum and maximum sentence boundaries, which cause the algorithm to abort even if there would be more sentences with expected positive gains, or to continue even with negative expectations.

6.3.6 Features

Document-Sentence Features

isMeta
Indicates if the candidate sentence is taken from the HTML meta description.

nthSentence
The position of the sentence in the list of all candidates, starting with meta description sentences.

nthSentence2
The position of the candidate sentence within the document.

numTokens
The total number of tokens the sentence comprises.

length
The total amount of single characters the candidate sentence has.

lengthToTokens
The total character count divided by the number of tokens.

stopwords
The absolute number of stopwords found in the candidate sentence.

stopwordsRel
The absolute number of stopwords divided by the total number of tokens.

fleschKincaid
The sentence's Flesch-Kincaid readability score: $.39 \cdot words + 11.8 \cdot \frac{syllables}{words} - 15.59$

numToTokens
The total number of digits divided by the number of tokens.

uppercase
The total number of single uppercase characters.

uppercaseToTokens
The average number of single uppercase characters per token.

textTfStemmed{Sum,Mean,Max}
Statistics for comparing pairwise cosine similarities between the tf vectors of the candidate sentence and all other individual sentences from the text.

textTfStemmedDist{Sum,Mean,Max}
Statistics for the relative term frequencies of all stemmed tokens found in the candidate sentence, where the term frequencies are derived from the original document.

icSum
The sum of the sentence's tokens' information content, as defined in Section 3.3.

icMean
The average information content of the sentence's tokens.

textLength
The total number of characters for the reference document.

textSentences
The total number of sentences in the reference document.

{tf, tfidf}_{Mean, Max}
The mean/maximum absolute frequency of a stemmed term in the candidate sentence.

{tf, tfidf}_DistMean
The mean relative frequency of a stemmed term in the candidate sentence.

{tf, tfidf}_Cosine
The cosine similarities between the sentence's and the document's tf and tf*idf vectors.

{tf, tfidf}_CosineStemmed
The cosine similarities between sentence and document using stemmed tokens.

Reinforcement Learning Features

Reinforcement learning provides a convenient framework for efficiently searching through the space of possible sentence collections for defining an automatically generated summary. The following features are devised to capture the expected improvement or loss of quality when adding a sentence candidate to the already selected sentences.

minDist
The minimum distance between the position in the list of candidates of the sentence to be added and the position of a sentence already added to the summary.

newTotalLength
The total number of characters found in the entire summary after adding the new sentence.

bleuDiff
The difference in BLEU-1 scores with respect to the document (*not* the gold standard summary), as defined in the next section, between the old summary and the new summary after adding the candidate sentence.

rougeDiff
The difference in ROUGE-1 scores with respect to the document, as defined in the next Section, between the old summary and the new summary after adding the candidate sentence.

cosineDiff
The differences in cosine similarity of term frequency vectors with respect to the document, before and after adding the sentence to the summary.

cosSelected{Mean, Min, Max}
The mean/minimum/maximum cosine similarity of term frequency vectors measured between the candidate sentence and those already added to the summary.

6.4 Quantitative Evaluation

6.4.1 Overview

In the quantitative evaluation of our single-document summarization approach, we again want to measure the correspondence between the output of our algorithms, and the expected results as given by the dataset's gold standard. For summarization, however, such an endeavor is even more complicated than for the previous two tasks, as a semantic correspondence between summaries is the primary focus, but also the presentation of a summary is a key factor. Different to keyphrase extraction, where a majority of expected keyphrases could directly be found in the respective documents, the Eduserver gold standard for single-document summaries typically has an abstractive character, i.e., the summaries rather describe the content than containing representative quotes from it. As it is challenging for an algorithm to identify sentences which appear to be semantically close to what can be expected as an abstractive summary for a document, it is also very difficult to correctly measure how well an actual abstractive summary is resembled by an extracted one. One of the key difficulties in this is that natural language is manifold and ambiguous, and therefore the same information could be conveyed using two completely distinct terminologies. On the other hand, what to include in a summary is subject to the author's interpretation and therefore two semantically different summaries might still be relevant on their own. Finally, it is either difficult or inefficient for large datasets to measure further aspects of summary quality, such as grammatical correctness or semantic coherence.

Still, for being able to compare our results with other publications, we will focus on quantitative metrics which only consider the terminological overlap between generated and gold standard summaries, thereby ignoring the other mentioned factors. For a broader insight into the performance of our algorithms, as well as for better identifying difficulties which further work should try to resolve, we again perform a quantitative user study supported by the DIPF, as we did in the previous chapters.

6.4.2 Metrics

BLEU ("Bilingual Evaluation Understudy") [PRWZ02] was first developed as a metric for measuring the precision of machine translation algorithms. However, it can also be reformulated to consider the precision of an automatic summary with respect to a gold standard reference summary. It does so by computing the relative overlap of n-grams found in the automatic summary $\hat{S}$ with those of the reference summary S:

$$\textit{BLEU-N} = \frac{|\{\textit{n-gram} : \textit{n-gram} \in S\} \cap \{\textit{n-gram} : \textit{n-gram} \in \hat{S}\}|}{|\{\textit{n-gram} : \textit{n-gram} \in \hat{S}\}|} \tag{6.3}$$

Herby, an n-gram corresponds to the single occurrence of a term, i.e., an n-gram is considered as also being characterized by the single position of the term, instead of a "class" view for which the set notation would only indicate that said n-gram was found at least once in the summary. In other publications this notion is also expressed by considering n-grams as word classes instead and introducing $count(\textit{n-gram})$ functions into the equation. Additionally, different from the rest of this thesis' interpretation that the n in n-gram denotes an upper boundary, the BLEU-N metric only considers n-grams which comprise exactly n single words. The fundamental idea of the BLEU metric is that each concept used in an automatic summary should also be present in the reference summary.

In a similar fashion, the ROUGE-N ("Recall Oriented Understudy for Gisting Evaluation") [LH03] [Lin04] [LO04] metric was formulated to express the recall of n-gram overlapping between automatic and gold standard summaries:

$$\textit{ROUGE-N} = \frac{|\{\textit{n-gram} : \textit{n-gram} \in S\} \cap \{\textit{n-gram} : \textit{n-gram} \in \hat{S}\}|}{|\{\textit{n-gram} : \textit{n-gram} \in S\}|} \tag{6.4}$$

As the reader may notice, both metrics are defined on the level of single generated summaries and their respective gold-standard counterparts, and thus are macro definitions of precision and recall, since a combined score for multiple summaries would amount to averaging over single overlap scores. Since this is the predominate use of this measure in the automatic summarization literature, we will omit the micro formulation of precision and recall in this chapter.

Again, for obtaining a combined score of precision and recall, which emphasizes both, providing relevant information, as well as covering all important aspects of the source document, we consider the F1 score. Precisely, we introduce the F1-N score, where N denotes the n-gram statistics used:

$$F_1\text{-}N = 2\frac{\textit{BLEU-N} \cdot \textit{ROUGE-N}}{\textit{BLEU-N} + \textit{ROUGE-N}} \tag{6.5}$$

Furthermore, as we also did in previous chapters, we additionally use cosine similarity between term frequency vectors for comparing automatic summaries with their gold standard counterparts. Again, for computing Tf*Idf vectors, we will use the DIPF Eduserver dataset as our reference corpus for document frequency statistics.

6.4.3 Baselines

Simple Baselines and Bounds

Since essentially both our approaches to automatic summarization follow the task of selecting representative instances from lists of candidates, much as it was the case with keyphrase extraction, we again are able to formulate comparable baselines and bound estimates. For many datasets the first sentences of a document are the most relevant candidate for typically being part of a summary themselves. For instance, this is the case for abstracts of research papers, as well as for most news articles. Therefore, the so-called lead baseline of either selecting the first n sentences or the first n words of a document is often hard to beat, even by much more complex models. For the DIPF Eduserver dataset, we use two variations, one including meta descriptions and one that doesn't, as well as we introduce a minimum tokens parameter, since many documents, especially Web pages, begin with titles or navigation terms instead of proper sentences.

Other self-evident baseline or bound approaches are a random selection of sentences, or ranking candidate sentences based on their cosine similarities with regards to the respective document. Furthermore, we are again interested in finding realistic upper bounds for extractive summarization approaches. To do so, we first introduce a baseline method which includes any candidate sentence in the proposed summary in order to find an upper bound on recall (ROUGE) for any algorithm using our candidate sentences without automatically enhancing or modifying them. Second, we consider all sentences from the gold standard summary as candidates and for each we identify the maximum cosine similarity when comparing them with all sentences from the original document individually. The idea is that when setting a high threshold on the similarity, we will only include gold standard summary sentences in our

results if they are almost identically found in the document. Doing so, we achieve an estimate of how well algorithms would perform which would be perfectly capable of only identifying actual gold standard sentences when extracting from the text.

TextRank and LexRank

For keyphrase extraction, we used a baseline algorithm called TextRank, which created a co-occurrence graph for terms and then performed the PageRank algorithm to find the most prominent terms in the graph. The concept behind this was that it can be assumed that important terms frequently co-occur with other important terms and therefore, with reinforcing connections between prominent neighbors, the most important keyphrases would be found in high-density areas of the weighted neighborhood graph. A similar idea can be applied to summarization, where not single terms but whole sentences are vertices in a graph with edges representing similarity. This approach was published under the same name, TextRank, by Mihalcea et al. [Mih04]. Again, central sentences would gain a certain "popularity" which then is reflected on its neighbors until, after iteratively updating vertice and edge weights, the most prominent sentences in terms of centrality and being located in high-density similarity neighborhoods would achieve the highest vertice weights. In the section on related work (6.2) we also mentioned LexRank, which is closely related to TextRank but developed independently, for example, supporting multi-document summarization. For comparison with our own algorithms, we use an open-source Java implementation of LexRank, which is provided by the *Dragon Toolkit* [1] [ZZH07].

[1] `http://dragon.ischool.drexel.edu/`

6.4.4 DIPF Eduserver Dataset

In Table 6.1, statistics of the DIPF Eduserver dataset for summarization are presented:

#	Variable	Value
1	Summaries	14379
2	Mean Length	444.6 ±268.4 [100, 5007]
3	Mean Sentences	3.2 ±2 [1, 38]
4	Mean Sentence Length	139.4 ±85.7 [3, 2039]
5	Mean Sentence Tokens	17 ±10.2 [1, 236]
6	Evaluation	50/50 Split
7	Min/Max Length	100 / 560
8	Min/Max Tokens	7 / 80
9	Min. Stopwords Ratio	.15

Table 6.1.: Statistics and Variables for DIPF Eduserver Summaries

Again, the total number of evaluation summaries is far below the original size of 27,000. As it was also the case with the dataset for keyphrase extraction in Section 5.4, this is mainly due to Web resources not being available anymore, as well as having database entries for which either the linked document or the gold standard summary is not suited for our evaluation.

First, we filter out any gold standard summaries which are not written for German documents or which in total do have less characters than our lower threshold of 100 on sentence length allows. By doing so, we reduce the amount of gold standard summaries for which it can be assumed that they do not represent enough depth and quality and thus would send false signals to model building as well as to our quantitative evaluation. Second, we extract summary candidate sentences from the respective documents using the filtering parameters given in the second half of Table 6.1, as well as using further filtering steps described in Section 6.3.3. If the number of candidate sentences is below 6, we also remove a database entry from the training and evaluation set since the reference document appears to not provide enough natural language information which would be required for making a quality automatic summary realistic.

After filtering, there are 14,379 entries left in the dataset, which we then split into equal-sized parts for training our models on one half and evaluating the results on the other. For model building, from the statistics, we can see that the average gold standard summary is about 450 characters long and made of about 3 sentences. There appear to exist some outliers, such as “sentences” which are only 3 characters long or are only made of one token. This most likely stems from parsing errors such as dots in abbreviations being misinterpreted as sentence delimiters.

The results of our 50/50 split simulation are presented in Table 6.2. For the sake of a clean presentation, we abbreviated the names of the metrics used: BL-N represents the BLEU score for N-grams, while RO-N represents ROUGE-N; F1-N is the harmonic mean between BL-N and RO-N; and TfIdf and TF measure cosine similarities between extracted summaries and gold summaries using their tf*idf and term frequency vectors for 2-grams, respectively.

In the table, we find three indicators named “InText” (#1, #4, and #13), which are obtained by considering all sentences of the actual gold standard summaries as candidates for an automatic summary. Those candidates were then filtered by computing pairwise cosine similarity values between them and sentences from the actual document and applying a minimum threshold on the maximum pairwise

#	Model	TfIdf	TF	BL-1	BL-2	F1-1	F1-2	RO-1	RO-2	#Sel.
1	*InText{0.8}*	.3301	.3427	.3924	.3866	.3167	.3172	.2869	.2891	1
2	SRL{GBRT{1000,0.01,0.5,8,30}, 0.2, 0.99^{t-1}-Greedy}{10000}	**.2767**	.4478	**.2619**	**.1799**	**.2679**	**.183**	.3173	.2118	3.1
3	F1-1{GBRT{1000,0.01,0.5,8,30}}	**.2774**	**.4707**	.2396	.1675	**.2629**	**.1804**	**.3536**	**.2312**	3.9
4	*InText{0.9}*	**.2741**	.287	.3351	.3298	**.2616**	.2617	.2334	**.235**	.9
5	F1-1{RF{20,0.8,10,10}}	**.2706**	.4467	.2487	**.1716**	**.2604**	**.1784**	.3207	.2133	3.3
6	F1-1{RegTree{7, minObs=30}}	.2611	.4427	.2367	.1626	.2519	.1714	.3173	.2089	3.5
7	Lead4	.2576	.4449	.2203	.1548	.2468	.1709	.3309	.2202	4
8	Lead4 (unfiltered)	.2356	.3682	.2255	.1431	.2281	.1431	.2744	.1674	4
9	Top3{CosSim{Tf}}	.1987	.3889	.1768	.0975	.1882	.1021	.2412	.1265	3
10	Top4{CosSim{Tf*Idf}}	.2135	.4131	.1571	.0907	.1866	.1061	.2784	.1518	4
11	LexRank{4, 2000}	.1899	.4144	.1523	.0854	.1794	.099	.2647	.1399	4
12	Random6	.1841	.4226	.131	.0736	.1711	.0943	.3009	.1572	**6**
13	*InText{1.0}*	.1707	.185	.242	.2362	.1556	.1553	.1262	.1271	.5
14	All	.2465	.4948	.0887	.0597	.1394	.093	.5356	.3371	20.4

Table 6.2.: Simulation Results for DIPF Eduserver Summaries

similarities to only keep gold standard sentences which appear to be quoted from the document. Our intuition is that this indicator gives us an insight into the ratio of extractive and abstractive gold standard sentences, and thus an idea of how high overlap metrics between extractive summaries and those gold standard summaries can possibly be. And indeed, as expected before, the amount of gold standard sentences directly copied from the texts is very low. On average, only about half of all summaries do contain a sentence which was exactly quoted from the text and thus the average recall of gold standard summaries by those quotes is only at about 12% (#13), as indicated by the ROUGE metrics. Gold standard sentences which find corresponding document sentences with cosine similarities of at least 0.8 or 0.9 (#1 and #4) yield much higher recall, but still on average are only found once in gold standard summaries — which typically have three sentences in total. Instead of having one extracted sentence in each gold standard, it should furthermore be save to assume that we rather have one third of summaries entirely quoted from the text while the rest cannot be matched exactly at all. A possible explanation for this is that the DIPF Eduserver authors prefer writing abstractive summaries or are even forced to do so if the source documents do not contain summaries on their own which can be quoted. Another possible reason for this observations is that the Eduserver database is being expanded for over 15 years now, and thus contains a significant amount of entries which are up to several years old. While manually inspecting entries, we often found that custom error messages (instead of better identifiable 404 http status codes) were presented for older links, or that we were redirected to the front page of the Website. Furthermore, Websites often change their contents over time, which is especially true for front pages, and therefore it may be the case that a summary once was quoted from a document which has then been changed afterwards. Altered contents especially effect the quantitative measures of the results when the Web pages are changed entirely, even in their content, for instance when the owner of a domain changes.

A direct upper bound on recall for our selections of candidate sentences is given by approach #14. Here we can see that even when producing summaries out of the first 30 proper sentences of a document, only half of all non-stopword, stemmed terms of the gold standard summaries are recalled (as measured by ROUGE-1). Those results yield significant implications for any automatic summary extrac-

tion approach applied on this dataset. While, with several thousands of training examples, still enough positive samples of sentences which also can be found in the respective gold summaries exist to extract characteristics from, even more than with keyphrase extraction, it is difficult to correctly evaluate negative examples in modeling as well as in our evaluation. Especially since a clear majority of summaries are abstractive with only weak correspondence with the original document's terminology, it is near to impossible to establish a precise classification into relevant and irrelevant example sentences for training. Even if a candidate sentence would serve perfectly as a summary sentence, missing an overlap with the gold standard summary means a negative training label if we want to optimize our results for BLEU and ROUGE. As a result, not only are statistical models difficult to derive from conflicting signals and false negatives; also the results of our quantitative evaluation must rather be seen relatively in the comparison between different approaches, than as an absolute indicator of the results' quality.

The most promising baseline approach is given by the lead baseline (#7), which just considers the first n candidates for each document. This is a typical observation for many datasets, as for most types of publications, the authors use to provide introductory sentences at the beginning of the documents which thus serve best as candidates for extractive summaries. However, it is noteworthy that those relatively good results are a direct result of our prefiltering techniques for sentence candidates, which also factors into the other baseline results. Therefore, we provide additional lead baseline results in line #8, where we turned off our filtering steps and select the first n sentences of a document which only have to surpass a minimum length of 50 characters to prevent the selection of entirely irrelevant sentences. As a result, the quality in terms of all quality metrics except for BLEU-1 decreases significantly.

Our learning-to-rank results, which are prefixed with "F1-1" to indicate that their thresholds were selected to optimize this metric, all are able to outperform the lead baseline on filtered results. For our implementation of bagged decision trees (#5) and all other models performing better than this model, the significance of the difference to the lead baseline is significant with a confidence level of over 99%. The best-performing learning-to-rank model again is driven by a gradient boosted regression trees base regression model (#3). With this model, we even obtain a slightly higher F1-1 score than the "InText{0.9}" baseline as we are able to achieve a much higher recall of single terms. This indicates that while most gold standard sentences cannot be retrieved exactly, our algorithms are still able to find sentences which on average still cover most important topics with a relatively high precision. This assumption is further supported by the relatively high cosine similarity values between retrieved and gold standard summaries' tf and tf*idf vectors (after stopword removal).

Our second approach to automatic summarization is reinforcement learning, for which we report our results as well. For most weaker supervised learning base models used to predict future rewards for sentences to be added to a summary, using reinforcement learning yielded insignificant improvements over using the same base models in our learning-to-rank framework. Therefore, for the sake of a clean presentation, we excluded those results from Table 6.2. When using a gradient boosted regression trees base model, however, we could achieve the best overall performance of an F1-1 score of .2679 (#2), as well as the significantly highest precision, as expressed through a BLEU-1 score of .2619. The particular reinforcement setup used for this model was a rewards discount factor of 0.2, an $\epsilon^{episode-1}$-greedy exploration policy for training, as well as a total of 10,000 training episodes, i.e., summaries that were constructed and observed rewards for.

Comparable to the previous sections on quantitative evaluations of our algorithms, we also report on the relative feature importance as determined by our highest-scoring models. Please remind that this distribution is determined by the reduction of squared errors that each feature caused when fitting the model to the training data. As feature selection is greedy for most implementations of decision tree-based algorithms, Table 6.3 is biased towards those features which provide the best splits at each level

of a tree. Other features may reduce a comparable amount of squared errors, but are not attributed with this information in the calculations underlying this table, since here only effective error reductions are considered. Still this information is valuable as it indicates which features discriminate very well and which features could be omitted for obtaining comparable results using the same models.

#	Feature	Importance
1	nthSentence2	.1319
2	nthSentence	.0981
3	textLength	.0677
4	tfCosine	.0637
5	textSentences	.06
6	textTfStemmedDistMean	.0542
7	textTfStemmedDistMax	.0498
8	tfidfCosine	.047
9	tfidfMean	.0408
10	lengthToTokens	.0384
11	icMean	.0364
12	fleschKincaid	.0305
13	uppercaseToTokens	.0277
14	stopwordsRel	.0275
15	tfCosineStemmed	.0264
16	textTfStemmedDistSum	.0261
17	icSum	.0211
18	tfidfMax	.021
19	tfMean	.0205
20	textTfStemmedMean	.0168
21	isMeta	.0159
22	length	.0141
23	uppercase	.0087
24	tfidfDistMean	.0084
25	contains_:	.0079
26	numToTokens	.0078
27	stopwords	.0064
28	textTfStemmedSum	.006
29	tfDistMean	.0051
30	numTokens	.0049

#	Feature	Importance
1	bleuDiff	.1712
2	cosineDiff	.0845
3	minDist	.0812
4	selectionSize	.0567
5	nthSentence2	.0484
6	textLength	.0351
7	textSentences	.0303
8	tfCosine	.027
9	tfidfCosine	.0263
10	icMean	.0238
11	tfCosineStemmed	.0238
12	rougeDiff	.0233
13	icSum	.0231
14	length	.023
15	tfidfMean	.0227
16	textTfStemmedDistSum	.0224
17	textTfStemmedDistMax	.0224
18	fleschKincaid	.0219
19	lengthToTokens	.0216
20	newTotalLength	.0195
21	textTfStemmedDistMean	.0166
22	uppercaseToTokens	.0158
23	tfMean	.015
24	tfidfDistMean	.0148
25	stopwordsRel	.0146
26	isMeta	.0138
27	tfidfMax	.0131
28	nthSentence	.013
29	tfDistMean	.0118
30	textTfStemmedMean	.0108

Table 6.3.: Top 30 Features for Learning-to-Rank (Left) and Reinforcement Learning (Right)

On the left side, the top 30 features for our best learning-to-rank model are given, while on the right side, the top 30 features for reinforcement learning are given. Comparing both, we can see that both strategies follow fundamentally approaches. While for estimating single sentences' relevance, our learning-to-rank model primarily focuses on their position in our candidate list, as well as their relevance for the entire document, the reinforcement learning model focuses on the changes to the entire summaries in terms of representing the respective documents when adding new sentences.

6.5 Qualitative Evaluation

6.5.1 Setup

Also for our third chapter on statistical machine learning methods for Web directory construction, we conducted a qualitative evaluation study with the help of the DIPF Eduserver authors. Different from the previous chapters, however, for several reasons it was our initial intuition that it would be unlikely to achieve results which were competitive enough for a blind study including human summaries. This first is due to the several difficulties and uncertainties in identifying a document's main concepts, retrieving sentences being representative enough for them, and presenting them in a suitable fashion. Furthermore, different from single URLs or keyphrase sets, natural language summaries typically comprise several hundred characters and therefore offer to the reader a large contact surface for potential errors. Even if we were able to automatically capture a perfect distribution over topics for a document, and representative sentences would be available for expressing the most relevant topics, being presented as a continuous summary, inconsistencies in grammar or semantical coherence between those sentences would easily give away the origin of such a summary.

Therefore, instead of comparing manually and automatically written summaries in a blind study, we rather performed a survey in which we asked the DIPF authors for their honest perception of how well relevant concepts were captured in the extracted summaries and if they could form a valuable basis for writing new summaries semi-automatically. Again, a random link from the Eduserver database is selected each time the evaluation website is opened. This link is then presented with its document title, its URL, and the gold standard summary, as depicted in Figure 6.1, to allow a fast and easy comprehension of the document contents for making it possible to rate the automatically extracted summary. Then, each sentence from our extracted summary is presented individually, along with the request to rate the sentence's information content on a 4-level scale from high to non-existent. Being related to our interpretation of "precision" as a quantitative measure, this question shall allow us to qualitatively evaluate how well the selection of sentences works on the content level. Furthermore, we were interested in the quality of the selected sentences as a whole and formulated three additional questions related to their presentation and correctness in terms of content and format, as well as their utility for manually writing a summary:

- All sentences together sufficiently characterize the contents of the document.
- The summary would be publishable with minimal corrections to grammar and spelling.
- The extracted summaries would make it easier to write a summary for this document.

Again, the agreement to each question is rated on a 4-level scale, ranging from a high agreement to rejecting the hypothesis.

6.5.2 Discussion

The qualitative evaluation of our summaries started on the 14th of March, 2013, and led to over 50 responses to single summaries within the first week. Unfortunately, the results quickly indicated that there still were severe problems with the selection of sentences, both in terms of content and presentation. Therefore, we disabled the evaluation of this specific task and started working on solutions which would hopefully yield significantly better results. And indeed, within the next two weeks we were able

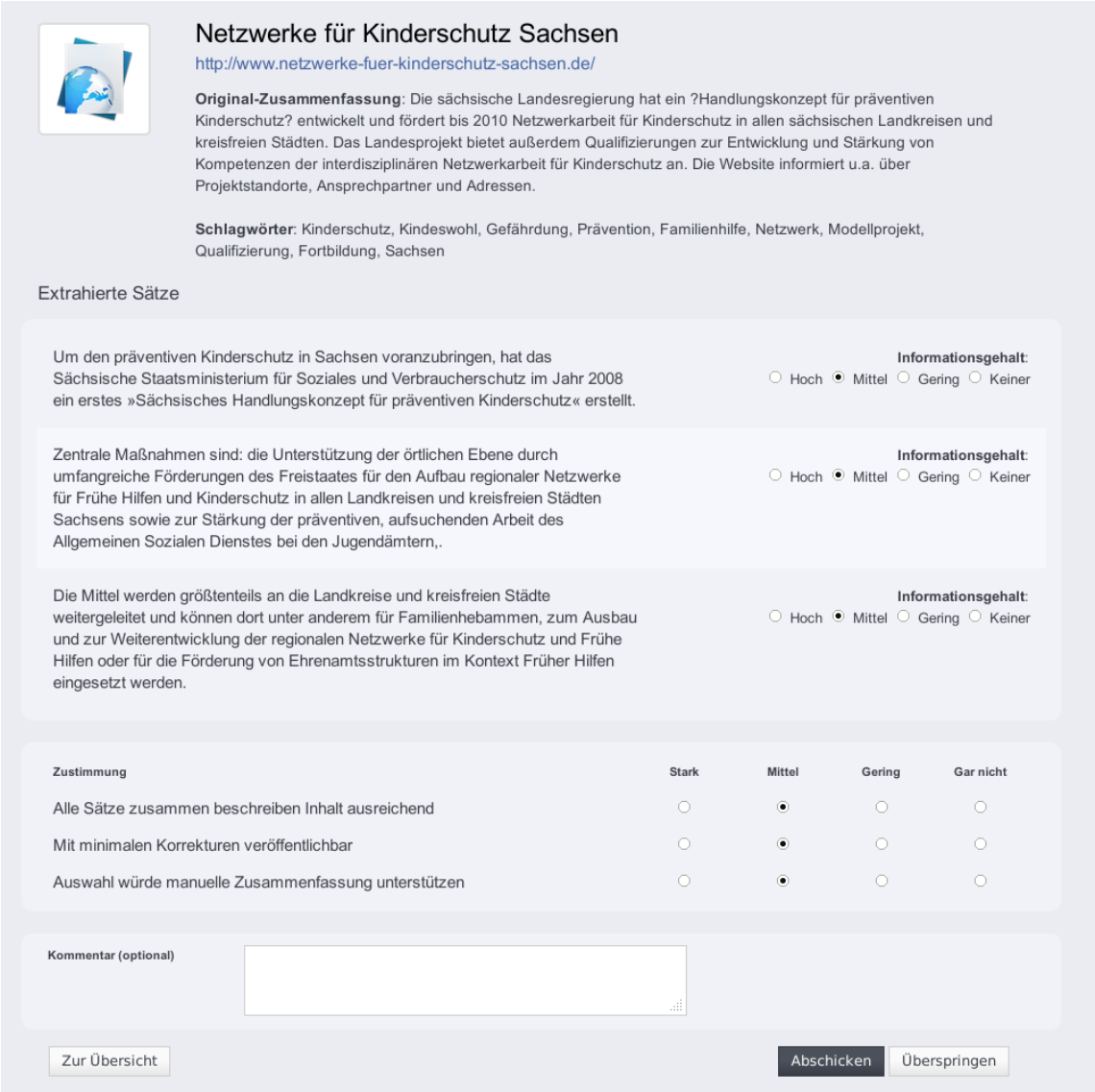

Figure 6.1.: Screenshot of a Random Summary In Our Evaluation Tool

to resolve multiple issues with open-source HTML and PDF parsers, which lead to an increase of the F1-1 metric combining BLEU-1 and ROUGE-1 from .22 to over .32, as also described in Section 6.3.2. Please note that this cannot be compared to the final results of our quantitative evaluation, as presented in Section 6.4, which only yielded a F1-1 score of .2679. This lower score for our most recent models is because in earlier stages of the development we applied significantly stronger filters on our evaluation data, such as a higher minimum-length threshold on gold standard summaries, which caused the gold standard summaries to be easier to match. Still, those numbers illustrate how much the results have improved as a result of those changes and how earlier evaluation feedback is not representative anymore. Because of the due date of this thesis in mid April, unfortunately there was not enough time left after re-opening the evaluation tool for summaries to gather enough data which would allow to present significant results for the improved version of our algorithm. Altogether, we were only able to collect feedback on 11 new summaries. Therefore, comparable to our evaluation of topical crawling, this section will rather serve as an illustration of a study design and as a basis for discussing potential threats to users' perception of the results, than it will serve as a definite judgment on our algorithms' quality — which will have to be postponed to potential further publications.

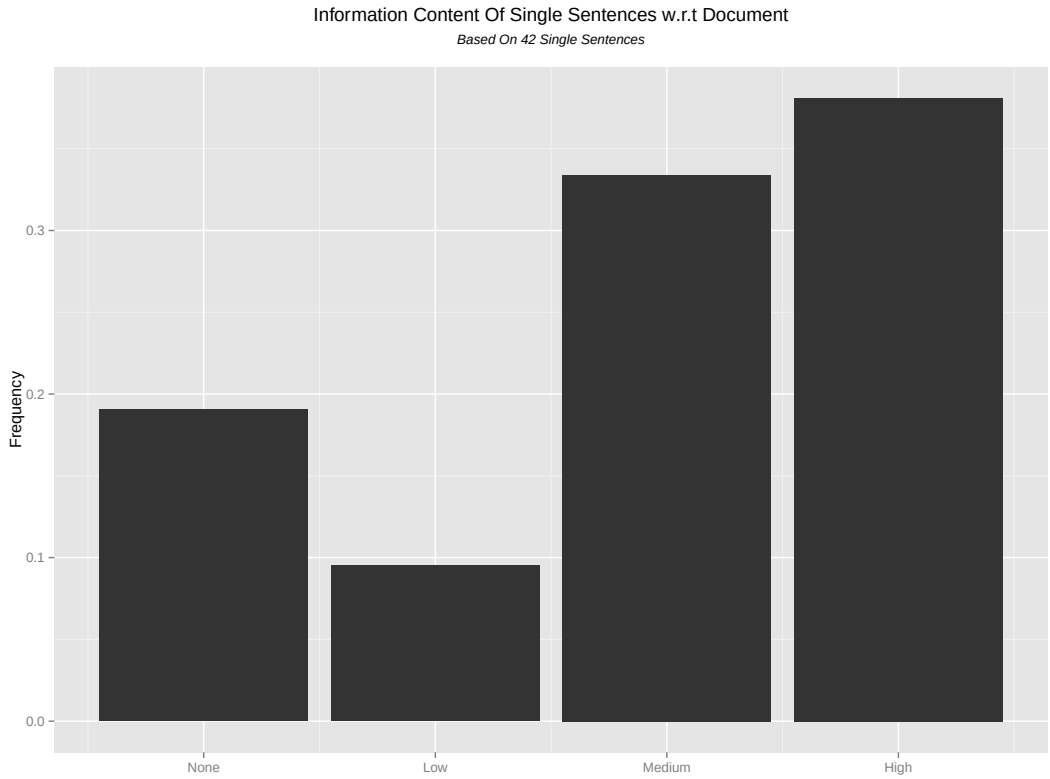

Figure 6.2.: Distribution Over Content Relevance Ratings for Single Sentences

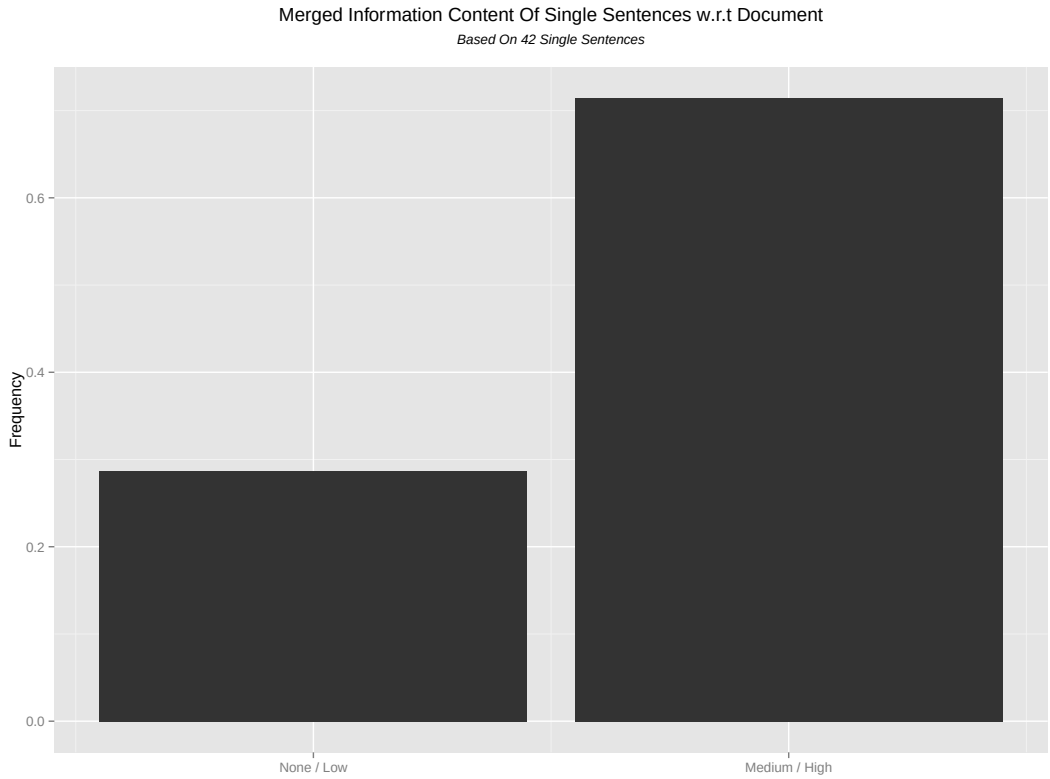

Figure 6.3.: Merged Distribution Over Content Relevance Ratings for Single Sentences

In Figure 6.2, the empirical distribution over ratings regarding the information content of single extracted summary sentences is given. In the order from left to right, the four bars represent no relevant information at all, a low information density, a medium one, as well as much information content with respect to the document. For the 42 single ratings received from DIPF authors so far, we can see that most sentences fall into the categories of having medium or high information content. It is especially noteworthy that with 38%, the majority of all sentences have a high information content, while only 9.5% of all sentences are deemed to have a low information content. However, 20% of all sentences are rated as providing no valuable information at all, which is why our interpretation of the available results is that within our current models good heuristics exist for selecting sentences, but at the same time more content-related features should be added to correctly identify when sentences, despite, for example, being located in promising areas of the text, are not as relevant as estimated.

Similar to the qualitative evaluation of keyphrase extraction, we also merged the ratings into the categories “None/Low” and “Medium/High” for a better separation into generally acceptable and generally weak sentence selections. From Figure 6.3, we can see that with over 71% — a total of 30 —, a significant majority of sentences can be regarded as semantically revelant, while less than one third of all sentences provide only little information.

In Figure 6.4, we present the DIPF authors’ agreement towards the question to what extend the selection of summary sentences was enough to sufficiently characterize the document. With the semantics of the axis remaining the same as for Figure 6.2, we can again observe a similar pattern to what we could see for single sentences. However, this time there is a fewer percentage of high quality summaries in terms of content while the majority of 45% falls into the category of a medium information recall. This observation is related to what we already discussed for keyphrase extraction in Section 5.5: For single sentences, an automatic approach just has to estimate the likelihood of the sentence containing relevant information. For entire summaries however, not only have multiple sentences to be correctly identified as relevant, but also dependencies between them as well as further requirements are introduced. In the search for an optimal summary, the minimal set of sentences has to be found which avoids redundancies while covering all major aspects of a document. With some documents, this may not even be possible, since some information might not be made explicit in single sentences, but instead, for instance, given through visual aids. As a result, even if every individual sentence of an extractive summary may be relevant, the summary as a whole can still be insufficient if it lacks some important information.

Again, a merged representation of the distribution over agreement categories is given in Figure 6.5. With 72%, despite the previous considerations, the relative amount of medium to high quality summaries in terms of content recall is practically the same as for the content precision of single sentences. However, it again has to be noted that ratings for 11 sentences is far from being enough to derive definitive interpretations. Instead, we are merely presenting arguments which would fit comparable distributions based on representative amounts of ratings.

Our second question regarding extracted summaries as a whole was concerned with the effort which would be required for transforming the provided summary into a publishable one, meeting the DIPF Eduserver standards. In Figure 6.6, we find a surprisingly homogeneous distribution for the 11 rated summaries. The agreement to the statement that the summary would be publishable after minimal corrections is found in equal proportions for low, medium, and high approval. In other words, the amount of summaries which could be published directly is equal to those which would require some modifications, as well as it is equal to the amount of summaries requiring a considerable amount of effort. Only those summaries which could not or should not be put manual efforts into for making them publishable appeared less frequently.

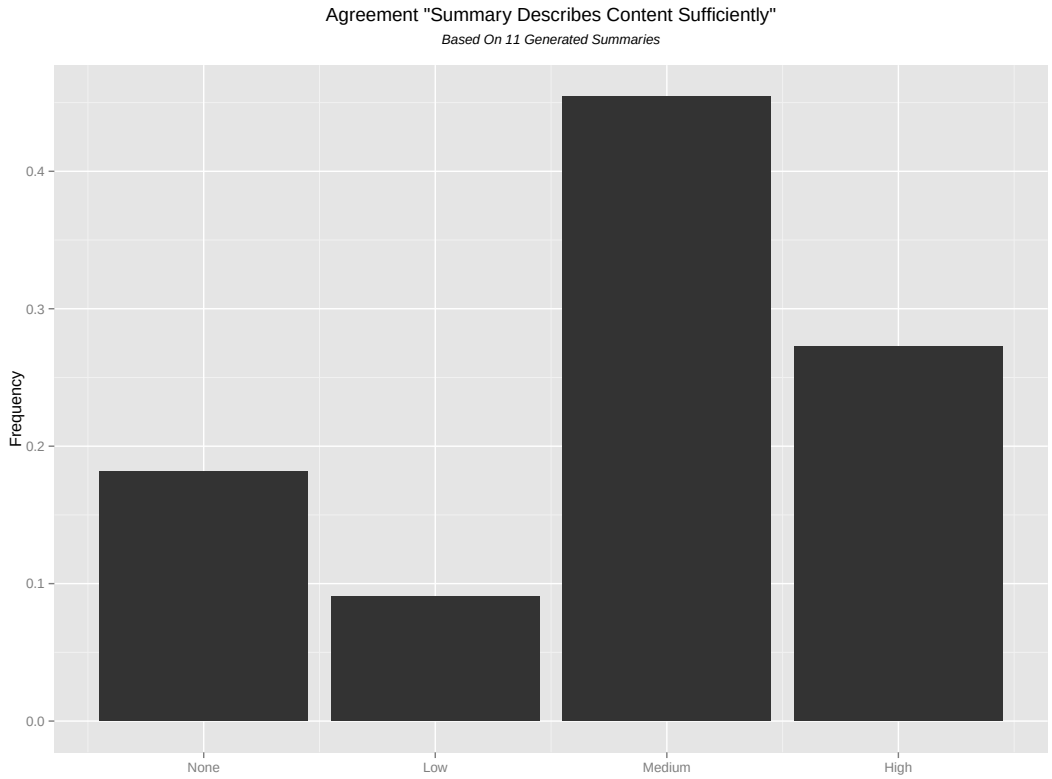

Figure 6.4.: Distribution Over Agreement for "Summary Describes Content Sufficiently"

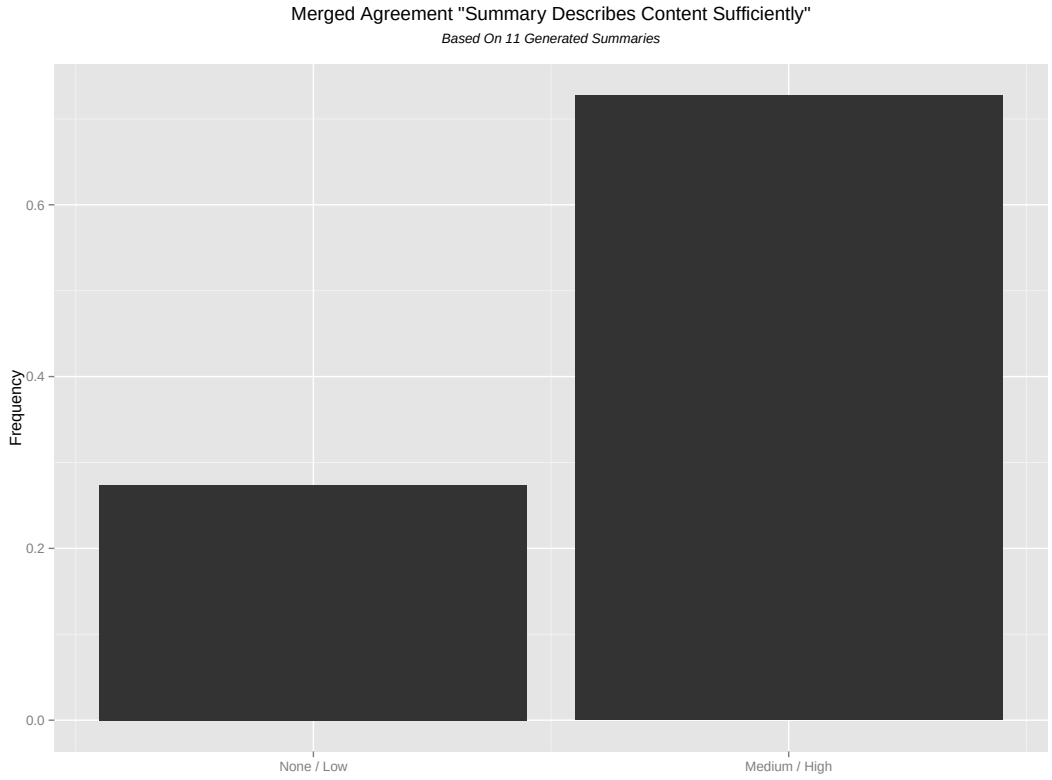

Figure 6.5.: Merged Distribution Over Agreement for "Summary Describes Content Sufficiently"

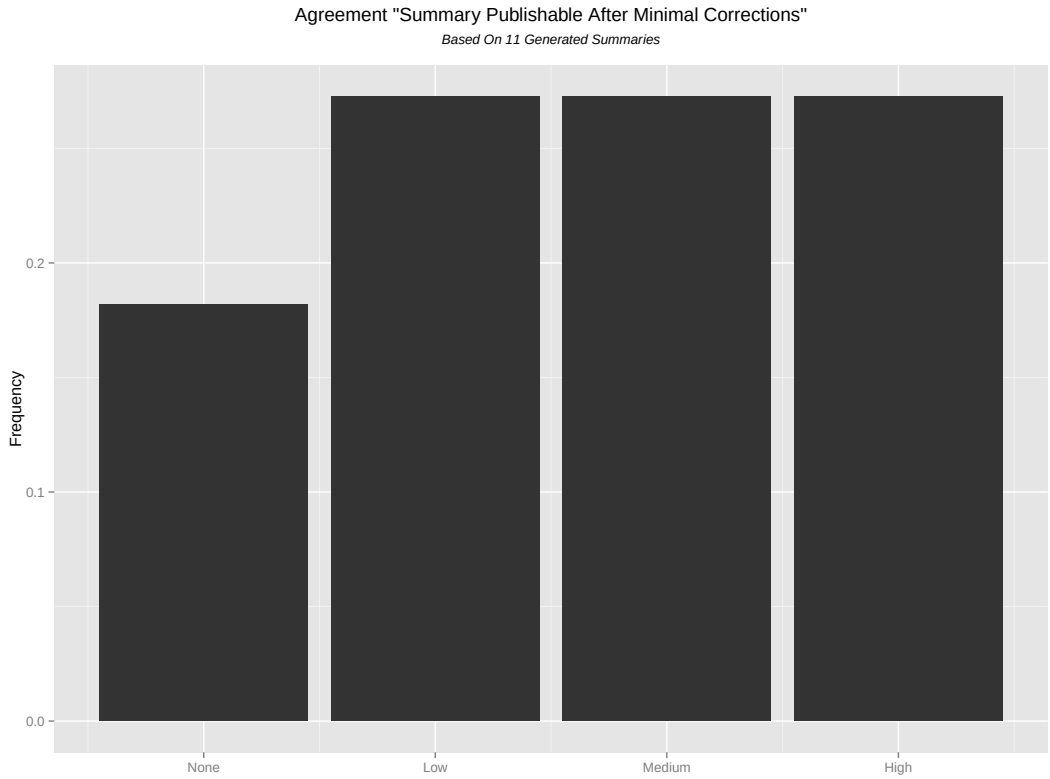

Figure 6.6.: Distribution Over Agreement for "Summary Publishable After Minimal Corrections"

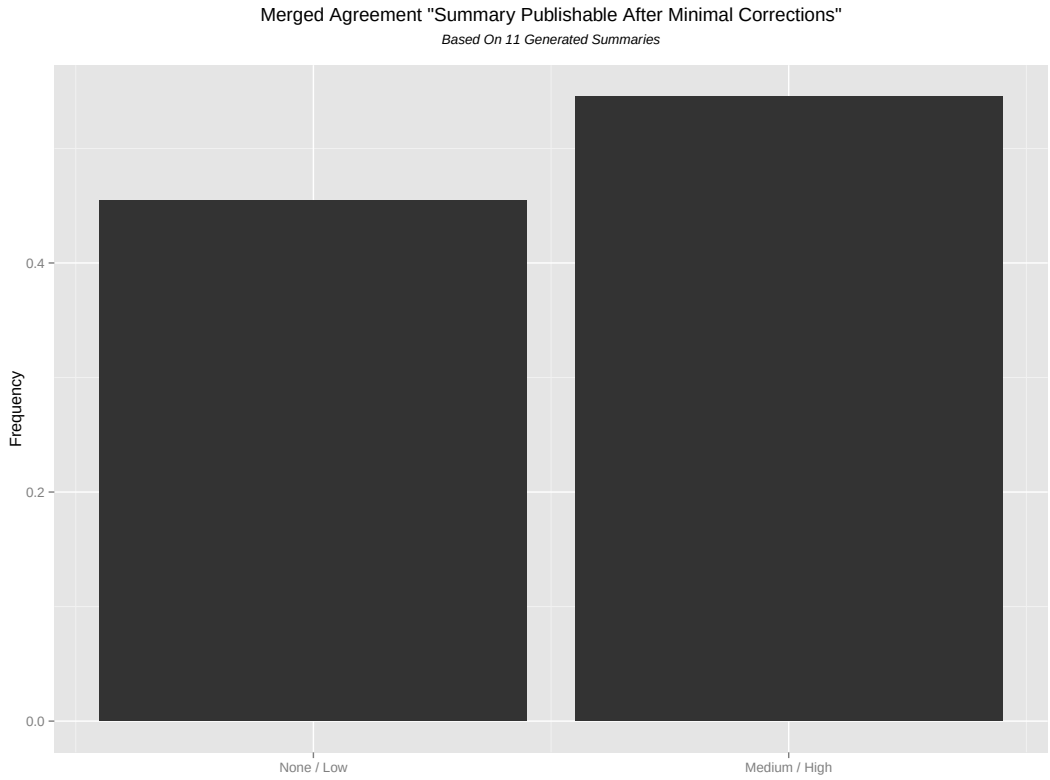

Figure 6.7.: Merged Distribution Over Agreement for "Summary Publishable After Minimal Corrections"

In the merged representation of Figure 6.7, the result that about half of all summaries could be turned into publishable summaries with little to medium effort is summarized. In our interpretation, such findings are related to those for content recall since compared to single sentences, finding whole summaries which are correct, relevant, and comprehensive in content, as well as publishable in style and other formal aspects, is much more difficult.

Tied to the previous question, in our last one we wanted to know that what extend DIPF authors can agree that the extracted summaries would be helpful for semi-automatically writing summaries. Such help could either result from suitable summaries which would only require a few modifications to be publishable, or help could result from the mere examples those sentences provide which could make understanding the document contents as the first step to manually writing summaries faster and easier. In Figure 6.8, we can see that the agreement to the helpfulness of extractive summaries is approximately normal distributed around a medium agreement as its mean.

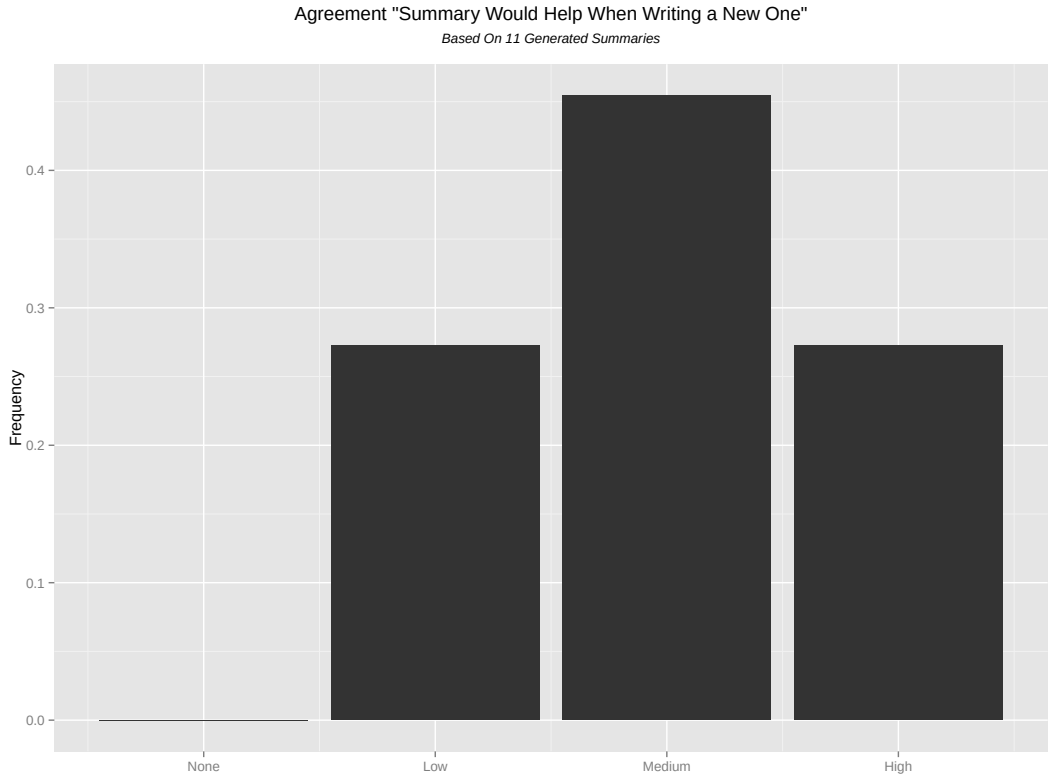

Figure 6.8.: Distribution Over Agreement for "Summary Would Help Writing a New One"

Again merging the agreement categories, in Figure 6.9, we see that a generally positive feedback of medium to high usefulness is equal in proportion to the agreement for summaries sufficiently capturing the contents of a document. From this we can assume that the main relevance of extracted summaries for semi-automatic summarization stem from their abilities to recall important information from the text. Possible corrections to content and presentation, especially when considering the efforts of writing a summary from scratch, appear to be less relevant.

In conclusion, the current results indicate a general usefulness for the majority of automatically extracted summaries. However, with the obviously difficult challenge of retrieving entire summaries which are relevant, comprehensive, and in an error-free form, there is a need for manual corrections for most of them, which is why, again, we suggest such a system to be used in a semi-automatic work flow.

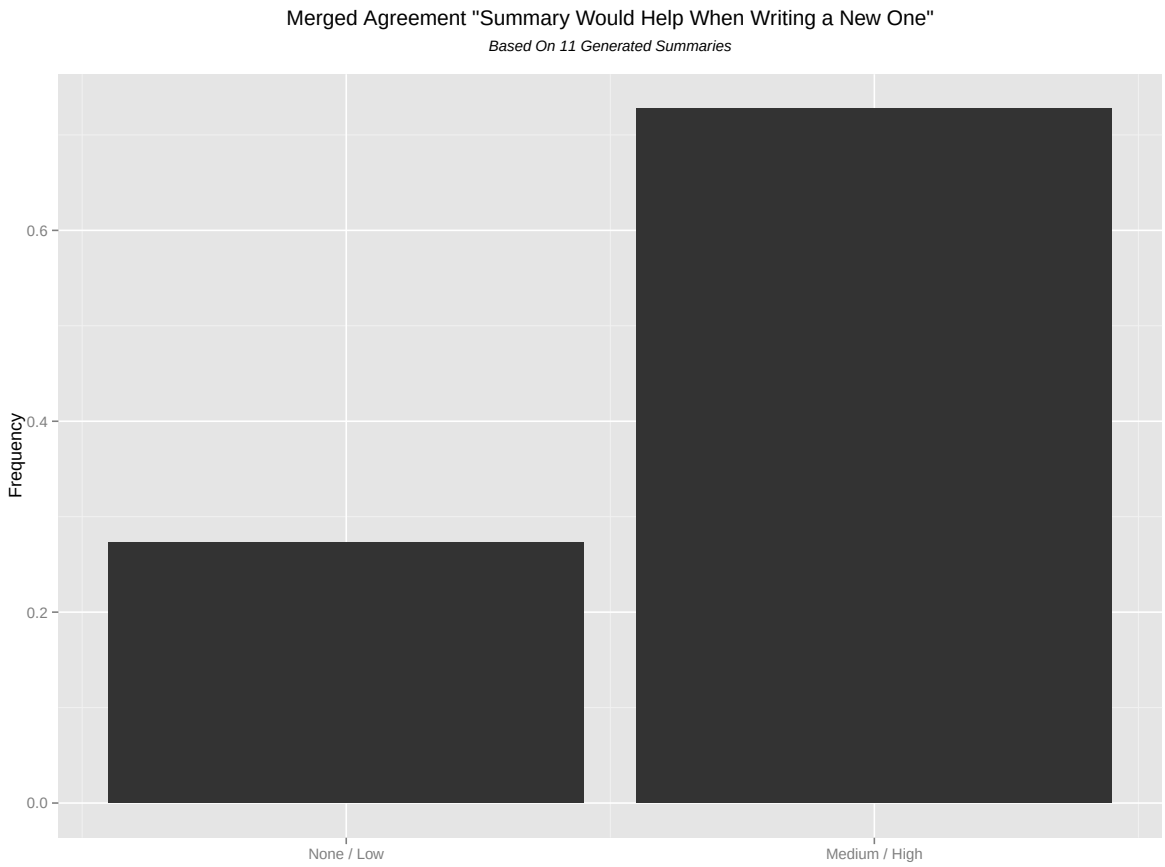

Figure 6.9.: Merged Distribution Over Agreement for "Summary Would Help Writing a New One"

6.6 Review

In this third task of our pipeline to automating the construction of Web directories, we presented two different approaches to supervised single-document summarization by sentence extraction. First, comparable to what we did in the chapter on keyphrase extraction and assignment, we used a pointwise learning-to-rank approach in which we assigned a relevance score to each candidate and then, based on training data, learned models for predicting candidate relevance scores based on a set of features. Again, this approximation of the scoring function was used for ranking candidate sentences, and thresholds on the amount of sentences and their minimum relevance were automatically derived from the training data to optimize the objective function's expected result. In our second approach to automatic summarization, we applied the reinforcement learning framework as a way to better model the human writing process and to leverage context information in the search for a best possible extractive summary. Similar to the learning-to-rank approach, we characterized sentence candidates by their intrinsic features, as well as their correspondence with key aspects of the reference document, but furthermore also introduced features which characterized the difference in quality before and after a new sentence would be added to the summary in the making. Also, since in the reinforcement learning framework, rewards are assigned to states of the search, instead of individual sentences, we were better able to also consider recall in our training example labels for supervised learning.

In our quantitative evaluation, we compared both approaches with each other, as well as with multiple baselines on the DIPF Eduserver dataset, which is primarily made of abstractive gold standard summaries. It was shown that using the better approximation to a human summarization process, as

well as leveraging the additional features, reinforcement learning models were indeed capable of outperforming our learning-to-rank approach on multiple metrics, while using almost the same features and the same function fitting approaches. Furthermore, we were able to significantly outperform all baseline approaches, which ranged from selections of the first *n* sentences after prefiltering, to unsupervised baselines trying to model sentence salience.

In our qualitative evaluation, we again asked the DIPF Eduserver authors for help in rating the extracted sentences' information content, as well as rating the quality of automatic summaries as a whole. While not yet reaching significant quantities, the results indicated that indeed a majority of sentences can be regarded as useful and while generally having smaller issues in content and presentation, many automatic sentences still provide valuable help for semi-automatic summarization.

In conclusion, our contributions are that we first described a supervised learning framework for automatic summarization which is able to handle any numeric scoring of training examples and learns to rank candidates based on those. This is different from most publications in this field which typically only assign boolean labels to training examples and just produce summaries by including all sentences which are predicted to be relevant and by then cutting the summaries off at a maximum length. Second, we studied the application of reinforcement learning to automatic summarization, which so far was only done in one other publication. Different from this competing approach, we defined different features, reward functions, and used a reinforcement learning framework which is able to use any regression model for predicting rewards, instead of just linear ones. Finally, it is noteworthy that by incorporating continuous scores into our frameworks, we are able to use any dataset for which relevance functions can be defined, e.g., by measuring the similarity with gold standard summaries. This allows to leverage larger quantities of training data compared to other approaches which typically require specific candidates to be manually labeled as relevant or not.

For further work, we suggest that quantitative evaluation metrics are devised which are better able to capture the various aspects of summary quality. When optimizing the results of statistical machine learning approaches to automatic summarization for such improved evaluation metrics, it can be assumed that a shift towards more favorable ratings in a qualitative evaluation would be observed as well. Furthermore, for an even better comparison with state-of-the-art single-document summarization, we suggest that the quantitative evaluation is extended to other established datasets, such as those of the DUC document understanding conferences , as well as obtaining executable versions of more advanced baseline approaches, or re-implementing them. Finally, this chapter lays the ground for multi-document summarization systems which operate in a similar fashion, but extract candidates from and compute features for multiple reference documents instead of just one.

7 Conclusions

In this thesis, we have researched different applications of statistical machine learning approaches to retrieval and summarization of educational document collections. Inspired by the DIPF Eduserver, an expert-curated German Web directory offering over 27,000 links to Web resources on more than 700 educational categories, we identified the main tasks and challenges involved in providing such services. First, one has to define the taxonomy of a document library, representing its various topics and subtopics. Second, for each category, the associated topics have to be researched in order to be able to efficiently search the Web for resources being most suitable for the target audience. Finally, those resources have to be presented to the users in a way which helps them to quickly navigate through the information available, and to identify those links which are most relevant to them. For this purpose, the DIPF Eduserver offers both, keyphrases and a short abstract for each directory entry to represent its content, focus, and intent. Needless to say, especially for large document collections, and Web directories designed like Eduserver in particular, a considerable amount of human effort is required in constructing them. Therefore, we began this thesis with the intention of exploring ways to automate significant portions of the Web directory design process. In particular, we chose the tasks of topical Web crawling, keyphrase assignment, and single-document summarization for confirming our hypothesis that by means of statistical machine learning, results can be produced that approximate those of human labor.

For topical Web crawling, we presented a design in which our software started to browse the Web from any given input URLs. While discovering new Web pages, we would refer to an additional input set of documents that are representative for the topic for automatically estimating the quality and relevance of those discoveries. Not only was this reference data used to determine which newly found pages were relevant enough to be added to our document collections; it would also be used as feedback for our crawler to teach him the quality of the decisions he made while accessing the Web — in particular, the quality of the links he chose to follow next. For enabling our crawler to automatically derive more efficient search strategies, we represented Web pages and links through a set of quantitative features which then could be used to automatically find a mapping between decisions' characteristics and their expected cumulated future rewards. By following this approach, our intuition was that topical crawlers trained in such a scenario would also be capable of learning to navigate through areas of the Web where much semi-relevant content would pose as red herrings to the topical crawlers — by distracting them from paths leading to more relevant results. To evaluate the capabilities of topical Web crawling, and handling those "hot zones" in particular, we proposed a new quantitative evaluation design. For categories taken from the DIPF Eduserver we used the respective entries as reference data for our algorithms and then started crawling large Web portals offering thousands of subpages of content on various educational topics. And indeed, for over 100 categories used in our simulation, we were able to retrieve a large amount of new URLs in a short time, that showed a high similarity to the reference entries from the Eduserver, thus leading to the conclusion that many of those would be relevant as supplemental resources. Finally, we conducted an online study in which we asked DIPF authors to evaluate the relevance of resources, for which they did not know if they were results of our crawler or taken from the original Eduserver database. The results indicated that a clear majority of Web pages crawled for very specific topic definitions could be regarded as relevant for the respective topics. Therefore, we have obtained strong evidence that carefully designed Web crawler are in fact capable of supporting humans in the effort of gathering material for educational document collections.

For keyphrase extraction and assignment, our approach was to first retrieve keyphrase candidates from the document itself that was going to be annotated, as well as to retrieve candidates from a collection of keyphrases that were manually assigned to other documents of our datasets before. The idea of using both sources instead of just extracting keyphrases from a text is inspired by the fact that almost half of all keyphrases found in the DIPF Eduserver dataset cannot be found in the respective documents. Instead, it appears that when annotating links in Web directories, it is often useful to also describe the contents on a meta level, using rather abstract terms. For learning to select keyphrases, we then in a second step gathered examples of human-provided keyphrase assignments from our datasets, and designated them as training data for our algorithms. To generalize over patterns found in the training data, we additionally had to characterize the relations between documents and keyphrase candidates through a fixed set of quantitative features. Comparable to our crawling approach, this information then was used for automatically deriving statistical models using machine learning techniques. In particular, we described a framework in which not only any supervised learning algorithm could be used for modeling the relevance of keyphrase candidates; furthermore, we also used algorithms for inferring optimal selection thresholds from the training data. For new documents to be annotated with keyphrases, our algorithms then would predict the relevance of all keyphrase candidates and rank them based on the estimates, cutting the selection off at the inferred thresholds. To evaluate our keyphrase extraction and assignment methodology, we used a variety of publicly available datasets in our quantitative evaluation. The results showed that not only our algorithms were able to annotate many documents the same way human annotators did for our gold standards; we were furthermore be able to significantly outperform any baseline algorithm on all datasets. In our qualitative evaluation, which was designed in the same fashion the user study for crawling was, we furthermore obtained conclusive proof that almost 80% of all single keyphrase assignments made by our algorithms can be regarded as relevant.

Finally, for automatic single-document summarization, we followed an approach similar to our solution to keyphrase extraction. As a result of learning to rank candidate sentences, we would obtain summaries that were composed of the presumably most representative sentences from the respective documents. Again, selection candidates were first extracted from the text — only this time, candidates were entire sentences instead of single terms —, and then characterized through intrinsic features, as well as through features representing their relations with the entire documents. For obtaining training examples, we then assigned relevance scores to sentence candidates based on their correspondence with gold standard summaries, and used statistical machine learning methods for deriving models for predicting those relevance scores for new data — entirely based on the candidates' features. Finally, using our learning-to-rank framework, the sentences from a document which were estimated to be most relevant for the document's summary would be retrieved, while considering automatically derived threshold on the minimum relevance estimates. Additionally, we also presented the single-document summarization task as a reinforcement learning problem, yielding further features but only small improvements on our dataset. In our quantitative evaluation on the DIPF Eduserver datasets we showed that, even though many gold standard summaries are rather abstractive than quoting from the texts, a relatively high correspondence between extracted and gold standard summaries could be measured. Preliminary results from our qualitative evaluation, again involving the feedback of DIPF authors, furthermore indicate that more than 70% of all sentences can be regarded as providing relevant information about the respective documents, and that also over 70% of such extracted summaries would be useful to the DIPF authors as a fundament for manually writing summaries.

In conclusion, in thesis we have provided quantitative as well as qualitative proof for our initial hypothesis being valid to large extends. Using statistical machine learning methods, we were able to design automatic approaches which are capable of leveraging training data for learning to produce re-

sults akin to those given by human authors. For many aspects of the results, such as the relevance of assigned keyphrases, we have shown that they can well compare to the human gold standard, for example provided by the DIPF Eduserver; for other aspects that still exhibit disadvantages in a fully automatic approach, such as summarization, it could still be shown that by laying the groundwork, they can significantly support humans in their efforts on retrieving and summarizing educational document collections.

A Manually Defined Stop Words

A.1 English Words

about all an and any are as at be become becomes been between both but by can did do does due each even first for from get given goes has have he her his how however if into is it its less let many me more most my new no not now of often or on one only other our over same see should some such than that that´s the their them then there they this those thus to two until use used using very we well were will with without when where which while would you your

A.2 German Words

ab aber abermals alle allem allen aller allerdings alles als also am an andere anderem anderen anders anhand ans auch auf aus außer außerdem ausserdem bei beide beiden beim beispielsweise bereit bereits besteht bezüglich bietet bin bis bitte bleibt bzw da dabei dafür daher damit danach dann dar daran darauf darin darum darüber das dass daß davon dazu dein deine deinem deinen deiner dem den denen denn dennoch der deren derer des deshalb dessen dich die dies diese diesem diesen dieser dieses dir doch dort du durch d.h ebenfalls egal eher eigene ein eine einem einen einer eines einmal entsprechend entweder er erst ersten erstens erster erstmal es etwa etwas euer eure etc finden folgende folgenden fraglich führt für ganz geht gibt gilt habe haben hamburg hamburger hat hatte hatten hätten häufig hier hierzu hin hinaus hinsichtlich ich ihm ihn ihnen ihr ihre ihrem ihren ihrer ihres im immer in indem innen innerhalb ins insbesondere insgesamt ist ja je jede jedem jeden jeder jedes jedoch jedweder jeweils kann kannst kommt konnte können keine kennt lassen lässt liegt machen mag man mdr mdr.de mehr meisten mich mir mit mittelbar muss müssen musst müsst musste müsste müssten nach nächsten navigation neben nehmen nicht nie noch nun nur ob oder oft ohne o.a sagt sagte sagten schon sehr sei sein seine seinem seinen seiner seines seit selbst sich sie siehe sind so sogar soll sollen sollte sollten somit sondern soweit sowie sowohl statt stehen steht stellt stets s.u um und uns unsere unter über überall überhaupt übersicht u.a u.u viel viele vielen vielfach vielleicht vielmals vielmehr vier vom von vor während war wäre waren warst was weg weil weiter weitere weiteren weiterer weiteres welche welchem welchen welcher welches wem wen wenn wer werde werden wessen wie wieder willkommen wir wird wo wobei worden wurde wurden würden zeigt zu zudem zuerst zukommen zukommt zur zum zwar zwei zwischen z.b z.t

List of Figures

List of Tables

Bibliography

[AAS03] Giordano Adami, Paolo Avesani, and Diego Sona. Clustering documents in a web directory. In *Proceedings of the 5th ACM international workshop on Web information and data management*, pages 66–73. ACM, 2003.

[AAS05] Giordano Adami, Paolo Avesani, and Diego Sona. Clustering documents into a web directory for bootstrapping a supervised classification. *Data & Knowledge Engineering*, 54(3):301–325, 2005.

[AG02] Massih-Reza Amini and Patrick Gallinari. The use of unlabeled data to improve supervised learning for text summarization. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 105–112. ACM, 2002.

[AKP07] George Almpanidis, Constantine Kotropoulos, and I Pitas. Combining text and link analysis for focused crawling—an application for vertical search engines. *Information Systems*, 32(6):886–908, 2007.

[BE+97] Regina Barzilay, Michael Elhadad, et al. Using lexical chains for text summarization. In *Proceedings of the ACL workshop on intelligent scalable text summarization*, volume 17, pages 10–17. Madrid, Spain, 1997.

[BF10] Gábor Berend and Richárd Farkas. Sztergak: Feature engineering for keyphrase extraction. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 186–189. Association for Computational Linguistics, 2010.

[BFSO84] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[BSA83] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on systems, man, and cybernetics*, 13(5):834–846, 1983.

[CD00] Hao Chen and Susan Dumais. Bringing order to the web: Automatically categorizing search results. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 145–152. ACM, 2000.

[CPS02] Soumen Chakrabarti, Kunal Punera, and Mallela Subramanyam. Accelerated focused crawling through online relevance feedback. In *Proceedings of the 11th international conference on World Wide Web*, pages 148–159. ACM, 2002.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[DC00] Susan Dumais and Hao Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM, 2000.

[DCL+00] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C Lee Giles, Marco Gori, et al. Focused crawling using context graphs. In *Proceedings of the 26th international conference on very large data bases*, pages 527–534, 2000.

[DM06] Hervé Déjean and Jean-Luc Meunier. A system for converting pdf documents into structured xml format. *Document Analysis Systems VII*, pages 129–140, 2006.

[DSY+10] Li Deng, Mike Seltzer, Dong Yu, Alex Acero, Abdel-rahman Mohamed, and Geoff Hinton. Binary coding of speech spectrograms using a deep auto-encoder. In *Proc. Interspeech*, pages 1692–1695. Citeseer, 2010.

[EC07] Gonenc Ercan and Ilyas Cicekli. Using lexical chains for keyword extraction. *Information Processing & Management*, 43(6):1705–1714, 2007.

[ER04] Günes Erkan and Dragomir R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)*, 22:457–479, 2004.

[FGG97] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2):131–163, 1997.

[Fri01] Jerome H Friedman. Greedy function approximation: a gradient boosting machine.(english summary). *Ann. Statist*, 29(5):1189–1232, 2001.

[Fri02] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[GEKH+12] Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M Meyer, and Christian Wirth. Uby–a large-scale unified lexical-semantic resource based on lmf. In *Proc. EACL*. Citeseer, 2012.

[GL01] Yihong Gong and Xin Liu. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–25. ACM, 2001.

[GP04] Alexandros Grigoriadis and Georgios Paliouras. Focused crawling using temporal difference-learning. *Methods and Applications of Artificial Intelligence*, pages 142–153, 2004.

[GTL+02] Eric J Glover, Kostas Tsioutsiouliklis, Steve Lawrence, David M Pennock, and Gary W Flake. Using web structure for classifying and describing web pages. In *Proceedings of the 11th international conference on World Wide Web*, pages 562–569. ACM, 2002.

[Hay94] Simon Haykin. Neural networks: A comprehensive foundation. 1994.

[HCC04] Chien-Chung Huang, Shui-Lung Chuang, and Lee-Feng Chien. Liveclassifier: creating hierarchical text classifiers through web corpora. In *Proceedings of the 13th international conference on World Wide Web*, pages 184–192. ACM, 2004.

[HCH+04] Zan Huang, Hsinchun Chen, Chia-Jung Hsu, Wun-Hwa Chen, and Soushan Wu. Credit rating analysis with support vector machines and neural networks: a market comparative study. *Decision support systems*, 37(4):543–558, 2004.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[HKJ+01] Anette Hulth, Jussi Karlgren, Anna Jonsson, Henrik Boström, and Lars Asker. Automatic keyword extraction using domain knowledge. In *Computational Linguistics and Intelligent Text Processing*, pages 472–482. Springer, 2001.

[HMM12] Stefan Henß, Martin Monperrus, and Mira Mezini. Semi-automatically extracting faqs to improve accessibility of software development knowledge. *arXiv preprint arXiv:1203.5188*, 2012.

[Hul03] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223. Association for Computational Linguistics, 2003.

[Jol05] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[JWF+10] Lu Jiang, Zhaohui Wu, Qian Feng, Jun Liu, and Qinghua Zheng. Efficient deep web crawling using reinforcement learning. *Advances in Knowledge Discovery and Data Mining*, pages 428–439, 2010.

[KFN10] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 441–450, New York, NY, USA, 2010. ACM.

[KM12] Pallika H Kanani and Andrew K McCallum. Selecting actions for resource-bounded information extraction using reinforcement learning. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 253–262. ACM, 2012.

[KMKB10] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26. Association for Computational Linguistics, 2010.

[Kü12] Renate; Ophoven Barbara; Bambey Doris Kühnlenz, Axel; Martini. Der deutsche bildungsserver - internet-ressourcen für bildungspraxis, bildungsverwaltung und bildungsforschung. 44:23–32, 2012.

[Lcv66] VI Lcvenshtcin. Binary coors capable or 'correcting deletions, insertions, and reversals. In *Soviet Physics-Doklady*, volume 10, 1966.

[LH03] Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics, 2003.

[Lin04] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, 2004.

[LL08] Marina Litvak and Mark Last. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24. Association for Computational Linguistics, 2008.

[LO04] Chin-Yew Lin and FJ Och. Looking for a few good metrics: Rouge and its evaluation. In *NTCIR Workshop*, 2004.

[LR10a] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[LR10b] Patrice Lopez and Laurent Romary. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 248–251. Association for Computational Linguistics, 2010.

[M+95] George A. Miller et al. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[Med09] Olena Medelyan. *Human-competitive automatic topic indexing*. PhD thesis, The University of Waikato, 2009.

[MI04] Yutaka Matsuo and Mitsuru Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(01):157–169, 2004.

[Mih04] Rada Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 20. Association for Computational Linguistics, 2004.

[MN+98] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[MNRS99] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. A machine learning approach to building domain-specific search engines. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 662–667. Citeseer, 1999.

[MT04] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, volume 4, pages 404–411. Barcelona, Spain, 2004.

[MW08] Olena Medelyan and Ian H Witten. Domain-independent automatic keyphrase indexing with small training sets. *Journal of the American Society for Information Science and Technology*, 59(7):1026–1040, 2008.

[Nav09] Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10, 2009.

[NL10] Thuy Dung Nguyen and Minh-Thang Luong. Wingnus: Keyphrase extraction utilizing document logical structure. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 166–169. Association for Computational Linguistics, 2010.

[NM65] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[NP73] Marc Nerlove and S James Press. *Univariate and multivariate log-linear and logistic models*, volume 1306. Rand Santa Monica, California, 1973.

[Osb02] Miles Osborne. Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization-Volume 4*, pages 1–8. Association for Computational Linguistics, 2002.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[Qui93] John Ross Quinlan. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[RA] Seonggi Ryang and Takeshi Abekawa. Framework of automatic text summarization using reinforcement learning.

[Res95] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

[Rid] Greg Ridgeway. Generalized boosted regression models.

[RM+99] Jason Rennie, Andrew Kachites McCallum, et al. Using reinforcement learning to spider the web efficiently. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 335–343. MORGAN KAUFMANN PUBLISHERS, INC., 1999.

[SB98] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

[SBHZ10] Mark D Shermis, Jill Burstein, Derrick Higgins, and Klaus Zechner. Automated essay scoring: Writing assessment and instruction. *International encyclopedia of education*, 4:20–26, 2010.

[She00] Chris Sherman. Humans do it better: Inside the open directory project. *Online*, 24(4):43–44, 2000.

[SKK+05] Sofia Stamou, Vlassis Krikos, Pavlos Kokosis, Alexandros Ntoulas, and Dimitris Christodoulakis. Web directory construction using lexical chains. *Natural Language Processing and Information Systems*, pages 87–103, 2005.

[SNK+06] Sofia Stamou, Alexandros Ntoulas, Vlassis Krikos, Pavlos Kokosis, and Dimitris Christodoulakis. Classifying web data in directory structures. *Frontiers of WWW Research and Development-APWeb 2006*, pages 238–249, 2006.

[SSL+07] Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. Document summarization using conditional random fields. In *Proceedings of the 20th international joint conference on Artifical intelligence*, volume 7, pages 2862–2867, 2007.

[SVB07] Krysta Svore, Lucy Vanderwende, and Christopher Burges. Enhancing single-document summarization by combining ranknet and third-party sources. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 448–457, 2007.

[TK07] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[WPF+99] Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255. ACM, 1999.

[WW99] Yong Wang and Ian H Witten. Pace regression. 1999.

[WWL08] Kam-Fai Wong, Mingli Wu, and Wenjie Li. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 985–992. Association for Computational Linguistics, 2008.

[WYX07] Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 552, 2007.

[Zha02] Hongyuan Zha. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 113–120. ACM, 2002.

[ZZH07] Xiaohua Zhou, Xiaodan Zhang, and Xiaohua Hu. Dragon toolkit: Incorporating auto-learned semantic knowledge into large-scale text retrieval and mining. In *Proceedings of the 19 th IEEE International Conference on Tools with Artificial Intelligence (ICTAI*, 2007.