Information Retrieval for Language Learning

An Exploration of Text Difficulty Measures

Niels Ott

Tübingen, April 14, 2009

Master's Thesis in Computational Linguistics
submitted to the
Seminar für Sprachwissenschaft, Universität Tübingen

Supervisor and first examiner: Prof. Dr. W. Detmar Meurers
Second examiner: Dr. Dale Gerdemann

Erklärung

Hiermit versichere ich, dass ich die vorgelegte Arbeit selbstständig und nur mit den angegebenen Quellen und Hilfsmitteln (einschließlich des WWW und anderer elektronischer Quellen) angefertigt habe. Alle Stellen der Arbeit, die ich anderen Werken dem Wortlaut oder dem Sinne nach entnommen habe, sind kenntlich gemacht.

(Niels Ott)

Abstract

This thesis explores a variety of text difficulty measures in the context of language learning and Information Retrieval. The possibilities of fast and straightforward retrieval of general information as well as of reading material at the language proficiency level of a learner are examined on the basis of a prototypical search engine implementation. In a preliminary evaluation experiment we found two of nine traditional readability formulas to be promising candidates for classifying texts gained from the Web into levels of text difficulty. In addition, the use of Lexical Frequency Profiles as indicators for vocabulary load appears to be promising as well. Having shown the general track to follow in order to retrieve information and reading at the learner's level, we suggest future work to investigate the mapping of the discussed difficulty measures to a well-established system of representing language proficiency, such as the system of Common European Framework.

Zusammenfassung

Die vorgestellte Arbeit untersucht eine Auswahl an Messinstrumenten für die Schwierigkeit von Texten im Kontext des Sprachenlernens und des Information Retrieval. Die Möglichkeiten des schnellen und einfachen Retrieval sowohl von generellen Informationen als auch von Lesematerialien auf dem sprachlichen Niveau einer Lernerin oder eines Lerners werden auf der Basis eines Suchmaschinen-Prototyps untersucht. In einem vorläufigen Experiment zur Evaluierung stellten sich zwei von neun traditionellen Lesbarkeitsindicies als vielversprechende Kandidaten für die Klassifizierung von Texten aus dem World Wide Web in Schwierigkeitsstufen heraus. Des Weiteren zeigten sich als Indikatoren für den verlangten Wortschatz benutzte Lexical Frequency Profiles als ein vielversprechendes Instrument. Der Weg zum Retrieval von Information und Lesematerial mit einem dem Lernenden angepassten Sprachniveau wird aufgezeigt. Als Thema zukünftiger Untersuchungen schlagen wir die Beziehung zwischen den vorgestellten Messinstrumenten und den Stufen eines etablierten Systems zur Repräsentation sprachlichen Niveaus wie dem System des Gemeinsamen Europäischen Referenzrahmens vor.

"I'm sorry Dave, I'm afraid I can't do that."

—HAL 9000 in *2001: A Space Odyssey*

Contents

List of Tables

List of Figures

1 Introduction

In a world of globalization where borders are slowly crumbling and where information has become available at the click of a button, the ability to communicate in an international environment becomes more and more important. As physical barriers disappear, language barriers are becoming more apparent. Therefore language learning and teaching is likely to play an even more prominent role in the future, both as a market and as an element of culture.

People learning a language naturally exhibit a wide variety in their abilities. For those learning a language by instruction, their proficiency is known in terms of their level acquired in language courses. It takes long to study enough to be able to understand every text in a foreign language. There are two issues resulting from this: 1) language learners have only limited access to information from modern sources such as the World Wide Web since the comprehensibility of the materials is not guaranteed. 2) There is an increasing need of exciting, appropriate, and up-to-date reading materials in language teaching.

Information at the learner's level is required. For retrieving information, the scientific field of Information Retrieval (IR) has led to the development of successful search engines such as Google[1]. Let aside the issues of information monopoly, the way of retrieving information and extending knowledge has changed dramatically since the rise of AltaVista[2], the first popular Web search engine. Yet there is still no guarantee that the documents retrieved are actually understandable to the users. From all those documents available on a topic, only those fitting the proficiency of the user or learner are of interest.

In the presented thesis, we will explore possible means to identify the reading difficulty of English texts in general, yet with a strict focus on language learning and Information Retrieval. The results of the presented work are based on insights gained from the implementation of a prototype search engine which we developed. This prototype system uses various methods for assessing text difficulty. The users can specify a level of language proficiency in the query module, allowing them not only to satisfy their information needs but also to find material that is readable to them.

For assessing text difficulty, we will discuss and explore the most prominent traditional readability formulas. These formulas usually are based on surface indicators such as average word and sentence length as well as the amount of long or complex words. For each text, they compute a single number putting the difficulty on a scale. Furthermore, we will investigate Lexical Frequency Profiles,

[1] `http://www.google.com`
[2] `http://www.altavista.com`

a statistical means of judging on learner vocabulary introduced by Laufer and Nation (1995), and their application as indicators for readability. An outlook on other promising measures, particularly those of syntactic complexity, will be given.

After an introductory chapter on the mechanics of modern Information Retrieval, we will present an approach for combining text difficulty assessment with search engine technology. Subsequently we will show that this approach can be implemented using state of the art technology from the fields of Natural Language Processing and Information Retrieval. We will furthermore discuss the possibilities of evaluating such a system as a whole. In a first statistical glance at a test collection of almost 200,000 unique documents, we will present a comparison of the various techniques used for automatically assessing text difficulty.

Having placed the presented thesis in the landscape of related work, we will attend to possible routes of further research, before we will eventually reach the summarizing conclusions.

2 Text Difficulty and Readability Measures

In order to deliver information at the learner's level, one must first of all be able to judge the reading difficulty of the information that is available. In this chapter, we introduce the concept of readability*. Furthermore we present and discuss traditional readability formulas as well as the application of Lexical Frequency Profiles for assessing readability.*

2.1 About Readability

Readability is a term commonly used to denote legibility of handwriting or typography, ease or pleasantness of reading, or to refer to the understandability and comprehensibility of a text. In readability research, only the latter meaning of the word is dealt with: the criteria making texts easy or hard to grasp (Klare, 1963, p. 1). *Text difficulty* can be seen as a synonym to readability.

From the reader's perspective, *reading proficiency* is the corresponding concept. The more proficient in reading somebody is, the less readable texts need to be in order to be understood. Unfortunately, the properties of readers and texts that define their position on the scales of proficiency and readability are not very clear-cut.

Klare (1963, ch. 1)[1] states that writers should fit their style of writing to the reader in order to produce readable text. Gunning (1968, p. 10) advises: "Don't write up. Don't write down. Write *to*." What writers can find out about their readers are according to Klare three pieces of information. If the writer adjusts to these three criteria, the text will be readable: 1) the text fits the reader's educational level 2) the reader has a strong motivation to 'conquer' the text, e.g. because he or she has a strong interest in the conveyed information, and 3) the reader has a strong background in the topic of the text. Two of these three points put the reader in the first place. Readability is defined via understanding and comprehension, two activities taking place in the reader's mind, not in the text itself.

The Common European Framework of Reference for Languages (CEF) uses so called *illustrative descriptors* to define language proficiency in general (Council of Europe, 2001; see section 4.2.2). These descriptors define levels from A to C, each divided into two sub-levels (A1, A2). All descriptors are given in task-dependent 'can-do lists'. These definitions mix social skills with purely linguistic

[1] Reprinted in Klare (2000a)

ones. Canada participated in the International Adult Literacy Survey (IALS). This survey specifies *Literacy Levels* from 1 to 4 that are defined by what people at a certain level can do or cannot do in order to "function in our society" (Evetts and Gauthier, 2005, p. 13). Turned upside down, these task-based definitions describe what skills a text is allowed to demand from the addressed audience in order to be still readable.

To conclude: readability describes how well a text fits the reader's abilities.

2.2 Readability Measures

2.2.1 Introduction

Readability measures, also referred to as *readability formulas* or *readability yardsticks*, aim to assign a single index of difficulty to a given text. Many of these measures use the U.S. grade level scale. DuBay (2004, p. 7) claims that "the grade of completed education is no indication of one's reading level." It is likely that in the early days of readability measures, scholars thought of grade levels as indicators for the reading proficiency school students had at a certain grade in school. Meanwhile, grade levels seem to exist as a readability scale irrespective of a given educational system.

As previously stated in section 2.1, the properties of readable text are not clear-cut. How readable a text is depends on both the reader and the text. However, if one is to measure the readability of a given text, no readers are available. This leaves us with the properties of the text. Which of those to look at is a re-occurring question to be answered in the following sections. There is no such thing as a comprehensive measure of readability.

Methods of measuring readability range from scratching the text's surface down to a deeper analysis. Most formulas rely on the relatively simple counting of sentences, words, and syllables. The ratios of these variables serve as proxies for an underlying complexity that cannot be touched directly. The vocabulary used in a text is another surface property that can be mastered by simple means. More complex analyses are inferred from the surface and include sentence structure such as counting prepositional or coordination phrases. The diversity of supposedly good indicators led to the development of a very large number of formulas, each putting the focus on a different spot, some being updates to earlier versions. Klare (1963) described 31 formulas, aiming to provide a complete overview. In 1981, over 200 formulas had been invented already (DuBay, 2004, p. 19).

Readability measures are designed with three main principles: 1) they are to yield results as accurate as possible 2) they are to fit a given usage scenario, such as a specific audience or text type 3) they must be easy and comfortable to use. The first principle is approached by conducting reading comprehension tests after subjects have read different texts. The formula then emerges from a regression equation matching the variables (such as sentence length or syllable count) to the text difficulty assessed via the human performance (Klare, 1963, ch. 3). The second

principle can be adjusted to by the choice of texts and subjects. The third principle was of major importance in the pre-computer age. Nowadays, computers perform cumbersome repetitive tasks, however with less linguistic competence. The necessary 'modern interpretation' of the old formulas is discussed in section 5.4.

Readability measures are a statistical means to infer text difficulty from clues mostly found on the surface of the text.

As noted above, a vast number of measures has been developed so far. This raises the question of which one might be the best one to use. Commenting on his 1963 book, Klare (2000b) presents a detailed list on the features of "the better readability formulas." A brief summary: good formulas combine counts of several indices such as word and sentence variables. They apply to connected discourse and they disregard format or layout of text.[2] They produce an indicator for reading difficulty, usually expressed in grade levels. However, they do not express anything about good or bad *style* of writing. Furthermore, good formulas require large samples or the analysis of the entire text in question.

In the pre-computer age, formulas imposing few workload on users were preferred. Nowadays, it appears natural to prefer formulas that yield good results when applied by diligent but linguistically less competent computer programs. The choice of measures in the presented work focuses on these three properties: 1) the formulas must be implementable in a computer program using well-established NLP technology. 2) they must be regarded as well-behaving in the field of readability. 3) there must be a diversity in the variables being used in the formulas. The latter point is important because different variables require different linguistic analyses. The diversity of variables gives us the chance to level out weaknesses of the different algorithms in use. (A systematic overview of measures and analyses is given in table A.1 on page 81.)

2.2.2 The Original Dale-Chall Score

"Over the years since it was published, the original Dale-Chall formula proved the most predictive of the wide-range readability formulas." (Klare, 2000b). Apart from being widely used and regarded well-behaved, this formula introduced by Dale and Chall (1948a) is also a good example for a word list-based readability measure. Their work strengthens the notion and use of *hard words* versus *easy words*. This distinction is a re-occurring pattern in later measures such as the Gunning Fog Index (Gunning, 1968; discussed in section 2.2.5).

Dale and Chall combine a list of easy words and the average sentence length into the following formula:

[2] We excluded the format or layout of text from the definition of *readability* in section 2.1. These criteria rather reflect *legibility*. However, a few formulas do not make this clear distinction.

$$\text{Dale-Chall} = 0.1579 \cdot DS + 0.0496 \cdot ASL + 3.6365$$

Where

$DS =$ Dale Score	The percentage of words outside the Dale list of 3000 words.
$ASL = \frac{\text{Number of Words}}{\text{Number of Sentences}}$	Average sentence length.

The word list of (almost) 3000 easy or *familiar* words was produced "by testing fourth-graders on their knowledge in reading of a list of approximately ten thousand words" (Dale and Chall, 1948a). The intended use of the word list and the list itself are given in Dale and Chall (1948b). The latter paper also shows that the list is a mixture of a lemma list and a full-form list: it simply contains both. A concise set of rules covers almost four pages of the publication. They cover all linguistic interpretations that must be done in applying the formula. Contractions and hyphenated words are counted as one word. For example, all words found directly on the list are considered familiar. The same holds for all numbers. Comparatives and superlatives are considered familiar, if the adjective they are derived from is found on the list. Irregular forms in general are unfamiliar unless they are present as full forms on the list.

The motivation for using a word list is the *vocabulary load* imposed on readers. Flesch (1943) finds the use of a word list undesirable, arguing that it does not only represent reading proficiency but also the living experience of the reader. He writes that a measure of abstractness should be the indicator of choice instead of vocabulary load. Hence he prefers the count of affixes in the words as an indicator of abstractness, claiming that words with more affixes are more abstract. Dale and Chall (1948a) respond to the argument by presenting the large list of 3000 words, accompanied with the claim that vocabulary load, abstract words, affix morphemes, or the number of uncommon words are all interrelated. Their conclusion is that the lookup of words on the list of 3000 is less cumbersome than counting affixes.

The values computed by this formula can be mapped to the grade level scale of grade four (values below 4.9) to grade 16 and above (values of ten and above).

2.2.3 Flesch Reading Ease

The Flesch Reading Ease perhaps is the most popular measure. "[...] it has become the most frequently used of all readability formulas." (Klare, 1963, p. 59). It is based on previous work conducted in Flesch's dissertation which is based on counting affixes (as an indicator of abstract words), sentence length, and personal references (Flesch, 1943). Updating his old formula, Flesch (1948) moves the personal references issue to a separate formula which he calls *human interest*. The nowadays-popular part however is what he calls *reading ease*. This formula also abandons the use of affixes and replaces them by a syllable count. Flesch

claims that "the measurement of word length is indirectly a measurement of word complexity", giving a correlation of complexity and word length measured in syllables of $r = 0.87$.

$$\text{Reading Ease} = 206.356 - 84.6 \cdot AWL_s - 1.015 \cdot ASL$$

Where

$AWL_s = \frac{\text{Number of Syllables}}{\text{Number of Words}}$	Average word length counted in syllables.[3]
$ASL = \frac{\text{Number of Words}}{\text{Number of Sentences}}$	Average sentence length.

The formula is based on a reading test conducted with children. The same data had been used for Flesch's old formula. He reports that "the grade level of children answering test question is not the best criterion for general readability". However, no better data were available back then, so he had to use them regardless of the concerns. The Flesch Reading Ease yields numbers from 0 to 100, expressing the range from "very difficult" to "very easy". It is meant to be used for measuring the readability of texts addressed to adult language users.

Flesch defeats the use of word lists (such as in Dale-Chall Score, section 2.2.2) in his dissertation, arguing that they would predict the words unfamiliar to the individual reader rather instead of the abstractness or complexity of words (Flesch, 1943, p. 15). However, Zipf (1936, ch. 2) had already observed the relation of word length and word frequency in a corpus before. One can say that the word length variable in the Flesch Reading Ease is also an indicator for word frequency. It is unclear to what extend Flesch was aware of this fact.[4]

2.2.4 Flesch-Kincaid

The Flesch-Kincaid measure is described by Kincaid et al. (1975). It uses the same analyses as the Flesch Reading Ease. The achievement of the Flesch-Kincaid measure is that it maps the Reading Ease to the U.S. grade level scale. The formula is stated as follows:

$$\text{Flesch-Kincaid} = -15.59 + 11.8 \cdot AWL_s + 0.39 \cdot ASL$$

Where

$AWL_s = \frac{\text{Number of Syllables}}{\text{Number of Words}}$	Average word length counted in syllables.
$ASL = \frac{\text{Number of Words}}{\text{Number of Sentences}}$	Average sentence length.

[3] Flesch (1948) uses "syllables per 100 words" instead of average word length. Consequently, his factor is 0.846 instead of 84.6

[4] Flesch (1943, p. 23) does cite Zipf (1936), but only for his hypothesis that language development tends to shorten often-used words, not for the relation of word length and word frequency. Flesch (1948) makes no reference to Zipf.

2.2.5 Gunning Fog Index

Gunning (1968) claims that the average sentence length and the portion of familiar words are the "most helpful" factors for computing a readability index. He finds the formula by Dale and Chall useful but too tedious to apply as it is based on a word list. Concerning the Reading Ease formula by Flesch, he finds the counting of all syllables of all words tedious as well. His solution is to count the *hard words*, which he defines as those words having three or more syllables. The argument is that those are easy to spot. The formula itself is given in natural language. We present a version translated to a mathematical representation:

$$\text{Fog Index} = 0.4 \cdot (ASL + 100 \cdot RHW)$$

Where

$$RHW = \frac{\text{Number of Hard Words}}{\text{Number of Words}}$$ Ratio of hard words to all words.

$$ASL = \frac{\text{Number of Words}}{\text{Number of Sentences}}$$ Average sentence length.

There are several exceptions for words with three or more syllables that are not hard words: proper names are always easy words. Compounds written together such as *manpower* must be split and their single components must be analyzed instead. Inflectional endings of verbs are not counted as syllables (Gunning, 1968, p. 38).

The Gunning Fog Index yields numbers that are supposed to correspond to the grade-level scale directly. Gunning draws a "danger line" between level 12 and 13. Beyond this line, any writing "runs the danger of being ignored or misunderstood". The formula is based on a regression equation that fits sentence length and the proportion of hard words to grade levels assessed in a reading test similar to the one used in development of the Flesch formulas.

2.2.6 Simple Measure of Gobbledygook

The Simple Measure of Gobbledygook (SMOG) is described in McLaughlin (1969).[5] He agrees with previous works such as Flesch (1948) that sentence length and word length have great predictive power for measuring readability. While the traditional formulas add a variable of word length to a variable of sentence length, McLaughlin is of the opinion that the two should be multiplied as they interact linguistically. For the counting of word length, he refers to the technique of counting polysyllabic words that had been introduced by Gunning for the Gunning Fog Index (section 2.2.5), arguing that it makes the analysis less

[5] McLaughlin (1969) refers to the name *SMOG* as being a tribute to Gunning's Fog Index (see section 2.2.5) and a reference to the weather phenomenon first spotted in London. The long version *Simple Measure of Gobbledygook* appears on McLaughlin's web page: `http://www.harrymclaughlin.com/SMOG.htm`)

laborious than the full syllable counting that is mandatory for the Reading Ease by Flesch (section 2.2.3).

McLaughlin states that the sampling of 100 words from a text as required by earlier readability formulas is not accurate enough. Instead, he suggests to use a sample of 30 sentences, which "typically cover 600 words". When transformed to a formula that operates on the number of polysyllabic words per sentence, his formula can be expressed as follows:

$$\text{Grade Level} = 3.1291 + 1.0430 \cdot \sqrt{30 \cdot RPS}$$

Where

$$RPS = \frac{\text{Number of Polysyllabic Words}}{\text{Number of Sentences}}$$

Similarly to the other works discussed in previous sections, McLaughlin used data from a reading test to adjust his regression equation in order to produce numbers of the U.S. grade level scale.

2.2.7 Läsbarhetsindex

While originally introduced by Björnsson (1968a), the following description of Läsbarhetsindex (LIX) is based on Jakobsen (1971), since we were unable to get hold of the original publication. The readability measures is given in natural language. Translated to a formula, it can be given as follows:

$$\text{LIX} = 100 \cdot RLW + ASL$$

Where

$RLW = \frac{\text{Number of Long Words}}{\text{Number of Words}}$	Ratio of long words to all words.
$ASL = \frac{\text{Number of Words}}{\text{Number of Sentences}}$	Average sentence length.[6]

Long words in the LIX sense are words with a length of more than six characters. Jakobsen (1971) describes many exceptions to be taken care of in the application of the formula. A number of text elements such as headings, the table of contents, phrases introducing direct speech, long citations—just to name a few—should be simply ignored. The rules for numbers seem to be even more sophisticated. Numbers being short or long words basically depends on the pronunciation which of course widely differs from phone numbers over dates to regular numbers.

[6] Jakobsen (1971, p. 16) advises to use "den gennemsnitlige meningslængde, d.v.s. punktumlængde" ('the average meaning length, resp. length of text between punctuation'). The German version of Björnsson's original publication (Björnsson, 1968b) refers to the very same concept as average sentence count.

Abbreviations are to be treated likewise, for example *USA* is supposed to be spoken and counted like three short words.

Even though LIX had originally been developed for Swedish, it is used in the very same form for English, too. Björnsson (1968b) discusses the application of the formula to Swedish and German. His finding is that scores computed for German are generally higher, but comparable. Hence the formula is language independent, but the interpretation of the scores varies. For Swedish, LIX values range from around 20 (very easy) to above 70 (very difficult).

Scores for English should be different but correlating.

2.2.8 Automated Readability Index

The Automated Readability Index (ARI) introduced by Smith and Senter (1967) represents an early approach of computing a readability score automatically. Smith and Senter conducted an experiment in which college students were to count syllables in textbook passages. They report a 10% variation in syllable counts among the 65 subjects. Their conclusion from the experiment is that readability score computed by a machine is more reliable. They constructed a 'readability measurement device', which was based on an electric typewriter that had been equipped with electro-mechanical counters for counting the number of key strokes, the number of words (space bar hits) and the number of sentences typed in. The latter had to be triggered manually by the typist by typing the equal sign.

From a sample of words with one to five syllables, Smith and Senter computed the average length of the words in characters. Their device could of course count only key strokes, a linguistic analysis such as counting syllables was not feasible. They present the following formula as the result of their studies:

$$\text{Grade Level} = 0.50 \cdot ASL + 4.71 \cdot AWL_c - 21.43$$

Where

$AWL_c = \frac{\text{Number of Characters}}{\text{Number of Words}}$	Average word length counted in characters.[7]
$ASL = \frac{\text{Number of Words}}{\text{Number of Sentences}}$	Average sentence length.

The formula was created by analyzing data obtained from 24 books. The grade levels of the books had been specified either by the Cincinnati School System or the publishers. The numbers computed with the formula again are supposed to reflect the U.S. grade level scale. Since the books used in development ranged from grade levels 1 to 7 only, it is questionable how reliable the results of the formula may be for materials of higher levels.

[7] Smith and Senter (1967) use the expression "strokes per word", referring to key strokes on a type writer. In deed, they count all strokes, including punctuation and numbers.

2.2.9 Coleman-Liau Index

This Coleman-Liau Index is described in Coleman and Liau (1975). Like the Automated Readability Index, it has been designed for automated use. Coleman and Liau state that the typing of punch cards is too tedious and more expensive than manual counting. Consequently, they suggest the use of an optical scanner that examines the printed text for periods, resulting in everything between periods being a sentence. Luckily, the situation has changed with the evolution of technology.

The measure is based on an estimate of correctly filled-in blanks in a cloze test. Coleman and Liau base their work on the analysis of 2,400 cloze responses by subjects for 36 150-word text passages. The cloze responses can be mapped to the grade level of the text. A first formula estimates the percentage of correct cloze answers from sentence length and word length. A second formula relates these "estimated cloze %" to U.S. grade levels. Coleman and Liau give the two formulas as follows:

$$\text{Estimated cloze \%} = 141.8401 - 0.214590 \cdot L + 1.079812 \cdot S$$

$$\text{Grade level} = -27.4004 \cdot \text{Estimated cloze \%} + 23.06395$$

Where

$L =$ The number of letters per 100 words.
$S =$ The number of sentences per 100 words.

It turns out that if one inserts the formula for the "estimated cloze %" in the grade level formula, the result is not plausible because it yields numbers below -3800. We found that in the grade level formula, what Coleman and Liau must have had in mind is the ratio of correct answers, that is the percentage divided by 100. After this little correction, the formula can be represented as follows:

$$\text{Grade level} = -0.2959 \cdot S + 0.0588 \cdot L - 15.8007$$

Or, with S and L adjusted to samples of arbitrary size:

$$\text{Grade level} = -29.5873 \cdot SPW + 5.8799 \cdot AWL_c - 15.8007$$

Where

$AWL_c = \frac{\text{Number of Characters}}{\text{Number of Words}}$ Average word length counted in characters.[8]

$SPW = \frac{\text{Number of Sentences}}{\text{Number of Words}}$ Number of sentences per word.

The specification of the formula as given by Coleman and Liau has led to some confusion in the past. For example, McCallum and Peterson (1982) confused the number of sentences per word with the average sentence length in words. The formula used in older versions of the *GNU style* program for computing readability scores is wrong in a different way.[9] Other publications may be wrong in other ways. Therefore we use the formula as given above throughout the presented thesis.

2.2.10 FORCAST

The FORCAST[10] formula is described by Caylor et al. (1973). Its primary design goal is to predict the readability of job reading material as used by the U.S. Army. The targeted reading audience consists of young adult males. Using regression analysis on the values of cloze tests, Caylor et al. came up with a simple formula that includes the number of monosyllabic words in a 150 words sample as the only variable. Adjusted to arbitrary sample size, the formula is the following:

$$\text{FORCAST} = 20 - 15 \cdot RMW$$

Where

$$RMW = \frac{\text{Number of Monosyllabic Words}}{\text{Number of Words}}$$

Like previously presented formulas, FORCAST is supposed to yield numbers on the U.S. grade level scale. This formula apparently is not very wide-spread in use. We chose to include it since its results are independent of sentence length.

2.3 Lexical Frequency Profiles

2.3.1 Properties of Lexical Frequency Profiles

Lexical Frequency Profiles (LFPs) were introduced by Laufer and Nation (1995). Since they do not boil down a piece of text to a single number, they cannot be counted towards the pure readability measures discussed in the preceding sections. Furthermore, they are originally not applied to text for the reader but to text by the language learner. Laufer and Nation define LFPs as follows:

[8] Coleman and Liau (1975) use the term *letters* instead of characters. Even though they do not specify it more precisely, it can be assumed that the refer to the characters contained in words only.

[9] At least version 0.7 of *style* is affected. Version 1.11 use the correct formula. The tool is available from `http://ftp.gnu.org/gnu/diction/`

[10] FORCAST is named after three of the authors that introduced it: *Fo*rd, *Ca*ylor, and *St*icht.

> "The LFP shows the percentage of words a learner uses at different vocabulary frequency levels in her writing—or, put differently, the relative proportion of words from different frequency levels."

This definition clearly states that LFPs are not a readability measure but a measure of the active vocabulary of a language learner. Laufer and Nation claim that their measure is superior to many others, the probably most popular among those being lexical variation, also known as type/token ratio. Tokens in this notion refer to the occurrences of words (punctuation excluded) in a text while types refer to unique words in a text. Hence the type/token ratio is high if a writer uses many different words and it is low if a writer uses only a small set of words. Laufer and Nation suggest to look up all words in the learners' writings in the list of the first 1,000 most frequent words as well as in the next 2,000 most frequent words, and the University Word List (UWL; Guoyi and Nation, 1984). Apart from these three categories, there is a category for all remaining words.[11] A resulting example profile is shown in table 2.1.

Word List	Tokens		Types		Families
GSL 1	2202	75.39%	542	54.25%	384
GSL 2	121	4.14%	94	9.41%	78
AWL	245	8.39%	136	13.61%	109
Others	353	12.08%	227	22.72%	n.a.
Total	2921	100%	999	100%	n.a.

Table 2.1: Example of a Lexical Frequency Profile.

The word lists used for a LFP are not just simple lemma lists or fullform lists. They consist of words that are grouped into families. The concept of word families is described in Bauer and Nation (1993). They define six cumulative levels of inflectional and derivational morphology that can be applied to the word stem of the given lemma. The difficulty for language learners to handle these word forms is supposed to rise with the levels.

Laufer and Nation (1995) used a program called *VocabProfile*. This program works on lists as described above. A cutoff was made so that words beyond level three of each family are treated as separate words. The example in table 2.1 has been created using a program called *Range* by Paul Nation.[12] Range uses the first 1,000 and the second 1,000 words from the General Service List (GSL; West, 1953) as basis for identifying the most frequent words. Instead of the University Word List, it uses the New Academic Word List (AWL) by Coxhead (2000). The

[11] This suggestion entails that the word lists must not overlap.

[12] *Range* is available from `http://www.victoria.ac.nz/lals/staff/paul-nation/nation.aspx`. While the program computes a total number of word families, we chose to omit this number in the example. Such a number cannot be computed without a complete list of all word families in the input text. The total number of families in the output of *Range* therefore is simply the sum of the families found on the three word lists, which is slightly misleading.

word family lists used by the program are not part of the original publications of the GSL or the AWL, they were derived later on. Our implementation of LFPs described in section 5.4 make use of the very same data set.

Laufer and Nation consider the count of word families "more revealing as an indication of lexical richness" than tokens or types. Their argument is that simple inflected or derived forms such as *happy*, *happiness*, *happily*, etc. are of no concern to the learners. However, beyond a given word family level, the words become difficult even though one knows the lemma. For example, *government* would be counted as a different word family than *govern*, because -ment is a level four affix.

2.3.2 Lexical Frequency Profiles as an Indicator of Text Difficulty

LFPs are used to measure the language proficiency of learners with respect to vocabulary. Their use originally is restricted to active vocabulary. Therefore an LFP should be computed for written text by language learners. In order to be used as a readability indicator as required for the purposes of the presented thesis, LFPs should measure the passive vocabulary of learners as this is what they need in reading. However, a study by Laufer and Goldstein (2004) hints to the direction that a small increase in the learners' active vocabulary implies a large increase of the passive vocabulary. Hence LFPs can be used as readability scores indirectly. This type of score is clearly limited to the measurement of the vocabulary load that is imposed on the reader by a given text.

Absolute values such as the number of tokens or types cannot be used as measures since they depend on the text length. Hence they must be normalized, which is similar to the computing percentage values such as those shown in table 2.1. It is imaginable that the percentage of types found on the GSL 1 list could serve as an indicator for reading ease: the more common words a writer uses, the easier should the text be. However, if the text is about an abstract topic, avoiding technical terms may as well make it harder to read. As Laufer and Nation (1995) point out, the count of word families is an interesting indicator. Unfortunately we cannot have a total number of word families since this would require a word family list that covers the entire input text. For arbitrary input, this kind of coverage cannot be guaranteed with any list.

A normalization of word family counts by the overall token count must be rejected. The same is the case for normalizing type counts by the overall token count (equivalent to the type/token ratio). Inspecting a large text, one will encounter fewer and fewer 'new' words the more of the text one is looking at (cf. Zipf's Law; Zipf, 1949). In other words, the type count grows much slower than the token count. Long texts will therefore always have a smaller type/token ratio than short texts. Consequently, a family/token ratio is affected even more because there are fewer families than types.

As a result of this discussion, we suggest to investigate two types of measures based on the LFP concept:

1. The proportion of tokens from each of the three word lists in all tokens (yields four measures, including the words not found on any list).

2. The proportion of types from each of the three word lists in all types (yields four measures, including the words not found on any list).

The investigation of the proportion of word families from each of the three word lists cannot be conducted. As explained above, there are no absolute counts of word families available for the entire text, hence no ratio can be computed. Absolute numbers for word families could be created by switching from the lookup in lists to a data-driven approach: the derivational and inflectional morphology discussed in Bauer and Nation (1993) can be implemented as a program generating the lemma plus the affix level. However, this lies beyond the scope of the presented thesis and must be left to further research.

3 Information Retrieval

Having discussed possible ways of assessing text difficulty, we turn to the techniques of Information Retrieval in this chapter. It is important to be aware of these rather engineering-focused methods as a further step towards gaining information at the reader's level. Information Retrieval is too complex to be used and understood as tool in a black box. Thus, we provide this introduction.

3.1 Defining Information Retrieval

Today, a single word is enough to give people an idea about the matter: *Google*. Most people know how to use a World Wide Web search engine such as Google and they have at least some kind of understanding about how such a tool works. Nevertheless, for the presented thesis we need to present a more detailed version.

Information Retrieval (IR) had appeared long before the emergence of the WWW in 1993. Many of the basic techniques introduced in this chapter of the presented thesis had already been discussed by Salton and McGill (1983). The 'huge' Web was not yet there, but computers also were 'smaller' both in processing power and in storage capacity. Salton and McGill refer to IR as a rather generic field: "In principle no restriction is placed on the type of item handled in Information Retrieval." However, they continue stating that what IR usually is concerned with is "narrative information", not the querying of databases. They contrast IR with database management systems, decision support systems, question-answering systems and management information systems (Salton and McGill, 1983, p. 7).

In the presented thesis, we employ the definition of IR as given by Manning et al. (2008, ch. 1):

> "Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)."

This definition implicitly states that IR does not mean *question answering*. Users of IR systems are bothered by an *information need*. They formulate a query to the system according to this need. However, they do not expect the system to answer a particular question. Instead, the system returns a set of documents in which the answer to the implicit question behind the query is given. Users expect the order of the results to represent the relevance of the documents behind them to the query. This interaction of the user with an IR system is illustrated in figure 3.1.

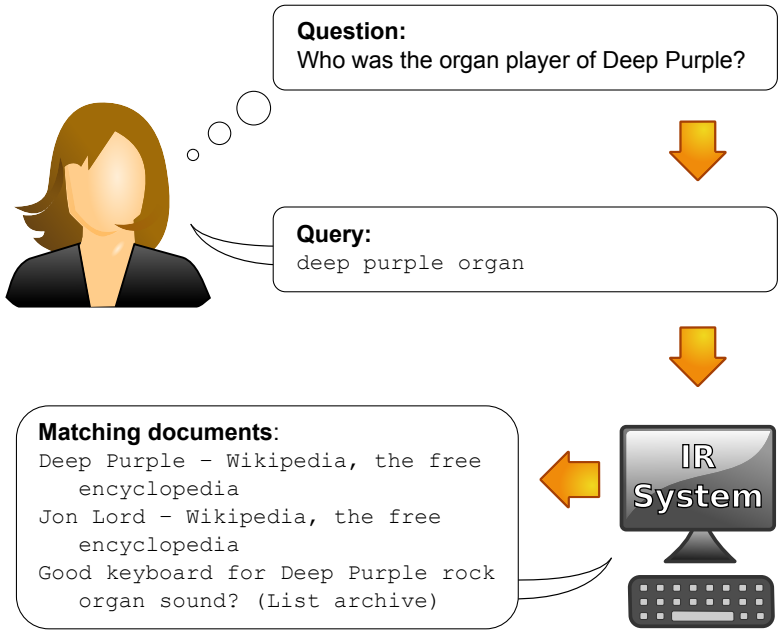

Figure 3.1: User interacting with an Information Retrieval system.

Manning et al. (2008, ch. 19) present a more fine-grained view on the users' needs. They distinguish three broad categories of queries: 1) *Informational queries* are those used for obtaining general information about a topic. Users want to learn more about something and they usually visit a number of web pages from the result list. 2) *Navigational queries* are supposed to take the user to exactly one spot on the Web, e.g. the site of a certain company. 3) *Transactional queries* are formulated when users want to find places on the Web where they can perform certain transactions such as buying a product, downloading music, or making a reservation.

3.2 Indexing

A simple form of IR is the use of a file search tool such as *grep* on the Unix command line (Manning et al., 2008, ch. 1.1). While this is an efficient solution for a limited amount of text data on a computer's hard disk, it is easy to imagine that for a text collection of larger size—such as a subset of the WWW—one must use something more advanced. The solution to the problem is *indexing*. Simply put, indexing means to sort the data in a way that allows quick access to the search terms. Normally, words are grouped into texts and texts are accessible as documents. An index puts the words in the first place. Therefore it is also called an *inverted index*.

Some more terminology: the unit in which texts usually are grouped is the *document*. However, in IR, a document need not to be a book or a paper. It is

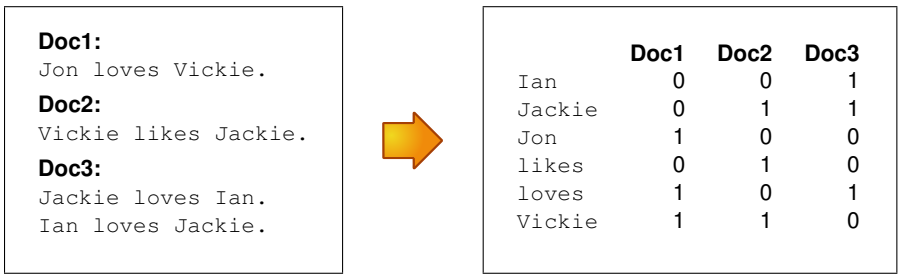

Figure 3.2: Creating a boolean index from documents.

also possible that a document could be as small as a paragraph. This is a design decision of the developer of such a system. What we generally see as a *word* is often referred to as *term* in IR. Consequentially, the expressions *document index* and *term index* are used.

A simple way of indexing is used in the *boolean retrieval model*. For each term, a vector of booleans is stored in the index. Each position in the vector corresponds to one document. The value of *true* or *false* at a given position indicates whether or not a term is present in the corresponding document. However, the boolean model does not allow for the ranking of results (Manning et al., 2008, chs. 1.1, 1.4). It only serves as an introductory example here. An illustration is given in figure 3.2.

For *ranked retrieval*, the 'importance' of each term-document relation in the index must be known. The general technique of the underlying *vector space model* was first described by Salton et al. (1975). Instead of a boolean value stating 'term is present' or 'term is absent', the vector space model specifies a weight for each term. This weight specifies how salient the term is for each document. One of the most popular formulas for computing term salience is the *TF-IDF* measure. Its definition according to Manning et al. (2008, ch. 6.2.2) is:

$$\text{TF-IDF}_{t,d} = TF_{t,d} \cdot IDF_t$$

Where

$TF_{t,d}$	Term frequency: the number of occurrences of the term t in the document d.
$IDF_t = log\frac{N}{DF_t}$	Inverse document frequency of the term t. N refers to the total number of documents in a collection, DF_t to the number of documents containing t.

The general behavior of this formula is to assign a high weight to terms that occur in a few documents only. Terms occurring in almost every document are assigned a low weight. The resulting index can be seen as a term-document matrix. This is illustrated in figure 3.3.

The vector space model is related to the *bag of words model*. In this model, the term frequency ($TF_{t,d}$) is stored in the index. Similarly to the vector space

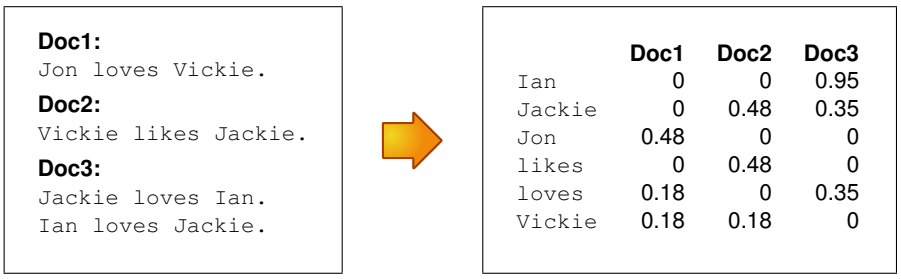

Figure 3.3: Creating a term-document matrix based on the *TF-IDF* measure.

model, the order of words in the documents is not stored (Manning et al., 2008, ch. 6.2). All three models described so far do not allow for querying phrases instead of keywords. The solution to this issue is to store a list of positions of word occurrences with each term-document pair. From this list, the original word order can be inferred, which then allows for the querying of phrases.

3.3 Searching

An index without the ability to query for documents would be of no use. For the boolean model, querying is straightforward: each term in the query is looked up in the index. The resulting boolean vectors are merged. The result is a set containing all documents that contain all search terms (Manning et al., 2008, ch. 1).

For querying in the vector space model, it is useful to see the inverted index as a term-document matrix. The rows in this matrix (figure 3.3) can be used to create *term vectors*. Since all entries of the matrix contain term-document weights, *document vector* can be constructed from the columns. The actual querying is done by representing the query as a vector as well. With this little trick, one can compute the *cosine similarity* between each document and the query. Each document is assigned a different score that yields the ranking of results. Manning et al. (2008, ch. 6.3.2) present the following formula for computing the score of a document:

$$\text{score}(q,d) = \frac{\overrightarrow{V}(q) \cdot \overrightarrow{V}(d)}{|\overrightarrow{V}(q)||\overrightarrow{V}(d)|}$$

Where

$\overrightarrow{V}(q) = [w_{q,t_1} \dots w_{q,t_n}]$	Query vector containing the weights of all terms in the query.
$\overrightarrow{V}(d) = [w_{d,t_1} \dots w_{d,t_n}]$	Document vector containing the weights of all terms in a single document.

Compared to the boolean model, this type of retrieval is rather complex to compute. Actual systems therefore take shortcuts such as including heuristics in order to work fast enough. Nevertheless, the cosine similarity does not only provide the matching but also the ranking of the results.

3.4 Natural Language Processing in Information Retrieval

In section 3.2, we use the expression *term* in the IR sense referring to words in a document. This also brings in the first Natural Language Processing (NLP) technique that is required in IR: tokenization. In general, one could use as much analyses as one can get, but in practice, IR systems use only a few strategies. The following information is provided by Manning et al. (2008, ch. 2):

Since almost all NLP strategies are language-dependent, language identification may be necessary in order to choose the right tools in the chain. The issue of Chinese vs. English in tokenization makes the dependency on language obvious. German compounds are written together. Hence compound splitting can improve the results in IR for German. Stemming or lemmatization applied to both the query and the documents noticeably improves the retrieval results. Some words occur very often but do not convey to meaning. These so called *stop words* such as *the* or *of* are oftentimes excluded from the index.

Equivalence classes can be used to map similar terms together. This strategy allows to deal with spelling variations (Manning et al., 2008, ch. 2.2.3). Unfortunately, many approaches using more advanced NLP techniques such as parsing or discourse analysis or the use of query and term expansion based on WordNet (Fellbaum, 1998) so far are not able to outperform traditional IR systems based on purely statistical methods (Tzoukermann et al., 2003).

Concluding, one can say the majority of successful NLP in IR takes place on the word level. The most important techniques in use are tokenization and stemming.

3.5 Crawling the Web

Before the WWW started rocketing up in the nineties, the information fed into IR systems was available locally in one form or the other. The web, lacking any table of contents or other centralized directory and being stored on an unknown number of computers all over the Internet, requires a strategy for obtaining the documents managed by an IR system. The technology dealing with this issue is called *crawling* or *spidering*. A *crawler* is a program that navigates through the Web by following the links it finds on the inspected pages. While doing so, it

collects all pages and passes them to the indexer of an IR system. According to Manning et al. (2008, ch. 20.2.1), a crawler consists of the following components:[1]

1. A *frontier* that holds all URLs that need to be fetched. The URLs may point to Web pages that are new or to ones that must be re-visited after a given time in order to update the index.

2. A module for fetching (downloading) the documents pointed to by the URLs managed by the frontier.

3. A module for parsing the downloaded documents in order to extract new links to follow.

4. A duplicate detector that prevents the frontier from re-visiting documents it has already seen.

There are many issues and details in crawling. For example, there are *spider traps*, web-sites that link to variations of themselves, keeping the crawler in an infinite loop. The crawler must also track pages that have vanished from the Web in order to make the indexer remove them or move them to an archive. The future advancement of the presented prototype IR system towards a real-world tool will make it unavoidable to include a crawler. For the purposes of the prototype, we assume the data fed into our indexer to be given.

[1] Manning et al. furthermore discuss the need for a Domain Name System (DNS) resolver module which we consider a technical detail to omit in our overview.

4 Combining Information Retrieval and Readability

Now that we have the assessment readability and the efficient retrieval of information at hand, the remaining question is how to bring those two together. In the following sections, we present a new approach for such a combination, eventually sketching a strategy to retrieve information at the learner's level.

4.1 Information at the Learner's Level

Language teaching or learning requires text material for reading practice. The motivation for reading increases when the text material fits the interest of the learner. The World Wide Web provides an incredibly huge amount of text and it is unlikely that there is not enough reading that fits the learner's interests. This large 'textbook' is accessible via search engines such as Google. However, texts must also fit the learner's language proficiency level in order to be useful for the purpose of practicing. This extends the traditional Information Retrieval (IR) paradigm as introduced in section 3.1 by the concept of a language level parameter. A query to such a specialized search engine consists of two parts:

1. A keyword query constructed by the language learner, derived from his or her information need.

2. A specification of the learner's level of language proficiency.

For 1) we assume that most users of the Web are able to construct queries without assistance. Constructing queries is not specific to language learning, users of search engines should be familiar with it from seeking information in their mother language. However, for 2) it is unlikely that the learner is able to give an exact specification of his or her skills in the language to learn. For example, the learner may be rather proficient in passive constructions but not proficient in gerunds. It is unlikely that the learner is aware of these details in a way that allows him or her to specify them in a query. Furthermore, the usability of a search engine would suffer dramatically from such a detailed query form. Consequently, the specification of the learner's level of language proficiency must be presented as a single parameter with simple values.

It is important to emphasize that a search engine for language learners cannot satisfy all information needs. From the query types presented in section 3.1, only the informational queries can be dealt with successfully. The index must hold

several documents for each information need, each at a different level of difficulty. This is for example the case if there are several online encyclopedias containing an article on a certain topic each.

For navigational queries, the user always wants to find the one and only web site. If the user wants to go to the official web site of the band Deep Purple, there will be no other official web site of deep purple in simple English. Perhaps for transactional queries such as seeking an online shop for buying a product, there may be several possibilities. The linguistic variety of online shops however remains questionable. Furthermore, users will most likely be interested in the cheapest offer for a product, not in the most readable one.

An issue not mentioned so far is the one of undesirable content: a search engine for language learning may be used by underage students. This calls for effective protection of the users from inappropriate content. It may be necessary to limit the search engine's scope to a (large) subset of the Web from which we suspect that it contains valuable material—or at least no offending pages.

4.2 Levels of Language Proficiency Used in Teaching

In this section, we focus on levels of language proficiency as used in education. There are great many scales on which one can measure how well a learner reads, but for the purpose of a search engine, it is important to use those scales that are known by teachers and learners.

4.2.1 U.S. Grade Level Scale and UK Key Stages

Many of the readability formulas discussed in section 2.2 yield scores which are supposed to reflect the *grade level* required from the reader in order to be able to understand the text. The grade level translates to the years of education in the U.S. system that the reader has mastered. Many designers of readability formulas see the grade levels as a continuous scale from 1 (primary school) to 17 (college graduate). This is illustrated in table 4.1 which is based on Gunning (1968, p. 40).

The grade level is to be understood as a ballpark figure allowing for quick and easy judgement of how suitable reading material is for readers. It is not an exact measure. DuBay (2004, p. 7) even mentions that the grade level is a measure of readability that has become independent from the actual number of years one has been educated: "The grade level of completed education is no indication of one's reading level. Average high-school graduates read at the 9th-grade level, which means a large number reads below that level." With respect to that statement, one can say that the grade level can serve as an indicator of appropriateness of a certain piece of reading, but the often-proclaimed direct relationship from the score of a readability formula to the actual grade level does not seem to be given.

Grade Level	Named Grade	
17	College	graduate
16		senior
15		junior
14		sophomore
13		freshman
12	High School	senior
11		junior
10		sophomore
9		freshman
8	Eight grade	
7	Seventh grade	
6	Sixth grade	

Table 4.1: U.S. grade level scale based on Gunning (1968, p. 40).

In the United Kingdom, a related system of *key stages* was devised by the government (UK Parliament, 2002, sec. 82). The key stage levels ranging from 1 to 5 reflect the stages in British state school education of students in the age range from 5 to 18. They are used by publishers to indicate the appropriateness of books for children as well as by some educational websites such as BBC Bitesize[1].

4.2.2 The Common European Framework (CEF)

Both U.S. grade levels and UK key stages are used to denote general difficulty levels of reading material. However, they both were not designed with language learning or language abilities in mind. The Council of Europe has gone a different way by introducing a system exclusively designed for judging language proficiency levels. The *Common European Framework of Reference for Languages* (CEF) is introduced in Council of Europe (2001, ch. 1) as follows:

> "The Common European Framework provides a common basis for the elaboration of language syllabuses, curriculum guidelines, examinations, textbooks, etc. across Europe. It describes in a comprehensive way what language learners have to learn to do in order to use a language for communication and what knowledge and skills they have to develop so as to be able to act effectively. The description also covers the cultural context in which language is set. The Framework also defines levels of proficiency which allow learners' progress to be measured at each stage of learning and on a life-long basis."

CEF distinguishes six levels of language proficiency ranging from A to C, each consisting of two sub-levels 1 and 2. These levels are meant to describe the *com-*

[1] `http://www.bbc.co.uk/schools/gcsebitesize/`

municative language competence of a language user[2]. The proficiency as categorized by CEF is divided into several *language activities*. These activities are described in so called *illustrative descriptors* which are basically 'can-do' lists. For example, the descriptor for the activity "sustained monologue: describing experience" states for the highest level C2: "Can give clear, smoothly flowing, elaborate and often memorable descriptions" (Council of Europe, 2001, p. 59). The most general description is summarized as a "global scale" given in table 4.2.

It turns out that the illustrative descriptors for CEF levels constitute a rather complex image of social and linguistic skills. They try to cover skills such as reading, listening, spoken interaction, writing summaries, conveying information, or using the appropriate level of politeness. Council of Europe (2001, ch. 9) reports on assessment of CEF levels in learners. The re-occurring relation between production and perception in learners shifts to the relation between 'can do' and 'requires ability'. For CEF, the tests are not limited to production as in Lexical Frequency Profiles (LFPs). Still, the intrinsic question answered is what a language user 'can do', not what a text requires from the language user for comprehension. The objective classification of texts into CEF levels must be left to further research here.

Nevertheless, using CEF levels for a search engine for language learners bears a few great benefits: 1) CEF is becoming increasingly popular in language teaching in Europe. 2) Language learners are likely to know their CEF level after they have mastered a language class, so they can specify that level in a search engine query. 3) CEF has been designed exclusively for the assessment of language proficiency and is not dependent on any educational system or particular language.

The popularity stated in 1) can be demonstrated by a few examples: Cornelsen, a German publisher of course books, issued a version of their *Headway* series used in English teaching to adults which is based on CEF levels. The UNIcert levels used in language teaching at many German universities aim at a one-to-one relation to the CEF levels B1 to C2 (Arbeitskreis der Sprachenzentren, Sprachlehrinstitute und Fremdspracheninstitute, 2006). The German state Bavaria recently introduced CEF in language classes taught in secondary school education (KWMBl, 2008).[3]

4.3 Models

Computer programs can only handle formalized representations of facts or relations. Therefore we discuss several types of models holding structured information about learners, texts, and levels of language proficiency. We give reason why there is no learner model in the special scenario of Information Retrieval for language learning. Text models capture the formal measures of text difficulty. With

[2] *Speaker* is a widely accepted term in linguistics. In the CEF context, the term *language user* is used instead. Not all language users speak, they may as well only read or write.

[3] In the Federal Republic of Germany, the states are responsible of school and university education, not the republic itself.

Proficient User	C2	Can understand with ease virtually everything heard or read. Can summarize information from different spoken and written sources, reconstructing arguments and accounts in a coherent presentation. Can express him/herself spontaneously, very fluently and precisely, differentiating finer shades of meaning even in more complex situations.
	C1	Can understand a wide range of demanding, longer texts, and recognize implicit meaning. Can express him/herself fluently and spontaneously without much obvious searching for expressions. Can use language flexibly and effectively for social, academic and professional purposes. Can produce clear, well-structured, detailed text on complex subjects, showing controlled use of organizational patterns, connectors and cohesive devices.
Independent User	B2	Can understand the main ideas of complex text on both concrete and abstract topics, including technical discussions in his/her field of specialization. Can interact with a degree of fluency and spontaneity that makes regular interaction with native speakers quite possible without strain for either party. Can produce clear, detailed text on a wide range of subjects and explain a viewpoint on a topical issue giving the advantages and disadvantages of various options.
	B1	Can understand the main points of clear standard input on familiar matters regularly encountered in work, school, leisure, etc. Can deal with most situations likely to arise whilst travelling in an area where the language is spoken. Can produce simple connected text on topics which are familiar or of personal interest. Can describe experiences and events, dreams, hopes and ambitions and briefly give reasons and explanations for opinions and plans.
Basic User	A2	Can understand sentences and frequently used expressions related to areas of most immediate relevance (e.g. very basic personal and family information, shopping, local geography, employment). Can communicate in simple and routine tasks requiring a simple and direct exchange of information on familiar and routine matters. Can describe in simple terms aspects of his/her background, immediate environment and matters in areas of immediate need.
	A1	Can understand and use familiar everyday expressions and very basic phrases aimed at the satisfaction of needs of a concrete type. Can introduce him/herself and others and can ask and answer questions about personal details such as where he/she lives, people he/she knows and things he/she has. Can interact in a simple way provided the other person talks slowly and clearly and is prepared to help.

Table 4.2: Global scale of CEF levels (taken from Council of Europe, 2001, p. 24).

query models, we present an approach for combining constraints on text models into criteria for specifying text difficulty levels as used in language teaching.

4.3.1 The Absence of a Learner Model

Learner models, also called *student models*, are a special type of *user models* used in Intelligent Computer-assisted Language Learning (ICALL). They provide a formal means of storing information about a learner. The type of information varies depending on the purpose. Many versions include a profile of the learner that holds information such as the learner's native language, motivation to study, name, level, gender, and so on (Amaral and Meurers, 2008; Murphy and McTear, 1997). An important component of such a model however is the one holding information on the learner's linguistic abilities. While the profile information generally remains static, the linguistic abilities change over time, for example together with the learning process in a language class. The actual data structure used for a simple learner model can be seen as a table or a database record.

The formal expression of linguistic abilities is often times stored in the form of *error counts*. An ICALL system poses training activities such as question-answering tasks to the user and analyzes his or her responses. For example, the number or case agreement in a verb phrase (VP) can be investigated by a parser and an agreement checker. The number of agreement errors in the learner's response is then stored in the learner model in the field for VP agreement errors (see for example Heift and McFetridge, 1999). The model is constantly adapted by the system while the learner is doing activities. Error counts are decreased whenever the user shows correct use of the corresponding form. Of course, when the learner has just started using the system, there are no data about his or her abilities yet. In this case, the initial values for the model can be obtained from a stereotype library containing typical values for groups such as novices, beginners, intermediate, and advanced (Murphy and McTear, 1997).

In ICALL, learner models are used to adjust the feedback given to the learner. Feedback refers to the message produced by an ICALL system after the learner has entered an erroneous sentence. The learner's response is likely to contain more than one error. Using the learner model, the error addressed by the system can be chosen on pedagogical grounds (Heift, 2003).

In Information Retrieval for language learning, a learner model could be used to retrieve texts at the learner's level without him or her actually specifying a language proficiency level. Given a search engine that is embedded in an ICALL application, the model of the user would be readily available.

The remaining challenge would then be to infer a formal search engine query from the learner model. Two basic strategies are imaginable: 1) if the model indicates that a learner is weak in using a certain linguistic feature, the search engine tries to avoid documents containing occurrences of this feature. 2) The search engine tries to prioritize documents containing a certain linguistic feature because the model indicates that the learner needs to practise this form. While the

first option may satisfy the learner as his or her information need is satisfied by the readable documents the search engine retrieves, the latter option is pedagogically more reasonable as it may train the learner instead of just ignoring the issue.

Unfortunately, all these benefits of a learner model are irrelevant to us since the presented prototype search engine is designed as a standalone application that cannot make use of any information provided by an ICALL system. The only information that is available is a vague judgement of the learner's level, given by the learners themselves. As a result, one must consider how to represent these vague levels in terms of a model suitable for querying a search engine. This is discussed in section 4.3.3.

4.3.2 Text Models

Type	Key	Value
General	Character Count	14249
General	Sentence Count	111
General	Token Count	2542
General	Type-Token Ratio	0.3703
LFP	Academic Word List Token Ratio	0.0816
LFP	Academic Word List Type Ratio	0.1389
LFP	General Service List 1k Token Ratio	0.1389
LFP	General Service List 1k Type Ratio	0.4191
LFP	General Service List 2k Token Ratio	0.0557
LFP	General Service List 2k Type Ratio	0.0841
LFP	Off-List Token Ratio	1.3119
LFP	Off-List Type Ratio	0.1325
Readability	Automatic Readability Index	12.7182
Readability	Flesch Reading Ease	57.6363
Readability	Gunning Fog Index	19.4510
Readability	Original Dale-Chall Score	8.8971

Table 4.3: An example text model holding general information, parts of a Lexical Frequency Profile and several readability scores.

In order to find the appropriate texts for language learners, the texts indexed by such a specialized IR system must be analyzed using the methods introduced in chapter 2. The results of the analysis must be stored together with the actual text data in the index. While the readability measures yield one score each, other measures such as the LFPs yield a bunch of different scores.

A text model as we use it for the purposes of combining readability and IR is a simple key-value table. It is used to describe the linguistic properties of a document fed into the indexer. An example is given in table 4.3. The structure of such a model is flat, even though the example indicates a logical grouping of the given key-value pairs. Some entries are of a general nature, such as the count of sentences in a document. The LFPs logically form a group of their own. There

is an entry for each readability measure. The example model is incomplete and serves illustration purposes only.

The text model for a document simply holds the results of the conducted analyses. The interpretation of some or all of these numbers is done using a query model.

4.3.3 Query Models

Category or Level: Easy	
Key	**Range**
Flesch Reading Ease	[0, 8]
Original Dale-Chall Score	[0, 8]

Table 4.4: Simplistic example query model with two ranges.

Query models are the counterpart of text models as specified in section 4.3.2. A query model consists of a table of ranges for fields in the text model. Formulated in natural language, an entry in a query model could sound as follows: "The Original Dale-Chall Score is between 0 and 6.5." An illustration is given in table 4.4.

Another view on query models could be the one of a set of constraints that are applied to the document: a document is counted towards the positive search results only if the text model of the inspected document contains values in the ranges given in the query model. Only if all constraints are met, the document is considered 'good'. Each query model contains only constraints for those variables that are known to be salient for identifying the desired proficiency level. Hence query models are underspecified on purpose.

Since CEF levels are becoming increasingly popular, query models should reflect these levels. On other words: a query model for a CEF level must contain constraints on readability measures and other scores that are only met by text models of texts that actually require that level from the reader.

Since the query models are supposed to directly reflect a level of language proficiency, we include a category or level name in the models themselves. The users of an IR system for language learning then must specify this level name in the query.[4]

4.4 Specific Research Questions

Having introduced the concept of searching documents of a given CEF level in the index of an IR system, the most important question is: how can we map from the readability measures and other scores to CEF levels? Another question inherent

[4]Quite naturally, he or she will be offered a list of available levels to choose from.

to the approach is: are there enough texts of different levels on the web? Both issues are addressed as part of further research in section 8.2.

5 Implementation: A Search Engine Prototype

The strategy for a Information Retrieval system has been sketched in the previous chapter. Consequently, we present the implementation of a prototype system in this chapter. The system is based on UIMA as a framework hosting NLP analysis modules and on the Lucene search engine library. Apart from these big elements in the picture, we discuss the implementation and modern computer-based interpretation of the traditional readability measures presented earlier.

5.1 System Overview

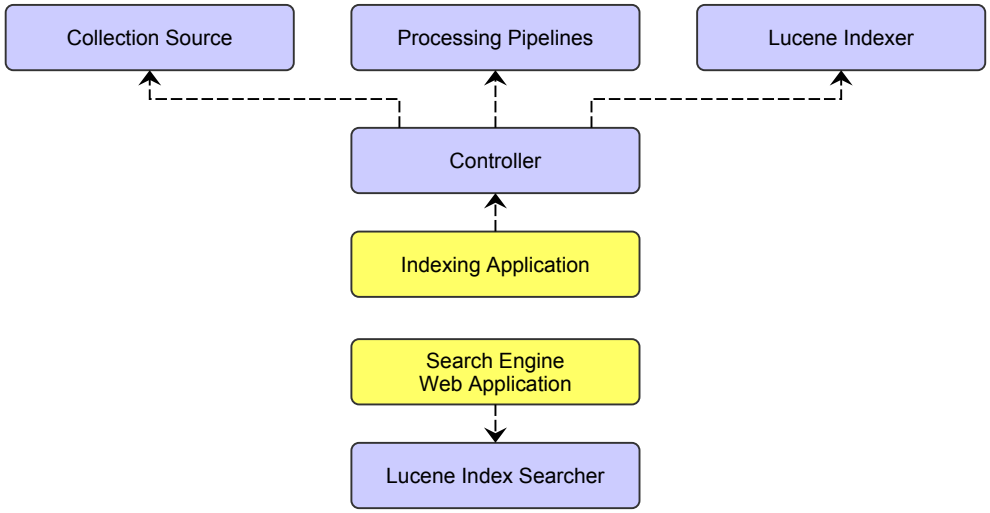

Figure 5.1: Simplified overview of the presented prototype, including indexing and querying applications.

There is a natural division into two components in an Information Retrieval (IR) system: indexing and querying. In the case of the presented prototype system, indexing is clearly more sophisticated. After an introductory overview we describe the details in the following sections. A greatly simplified view on the presented system is given in figure 5.1. The entire system is written in the Java programming language.

The indexing application takes care of reading the input data, eventually producing an index suitable for fast querying. In order to do so, the application makes use of a central controller. The controller reads the HTML data from disk and hands them to three processing pipelines. The result of these pipelines then is passed on to the indexer by the controller.

As mentioned earlier in section 3.5, the prototype system does not make use of a Web crawler. The data source is of a generic nature, specified by an interface. The current implementation of that interface reads the data from disk expecting the format of the Unix tool *GNU wget*[1]. Since wget is capable of crawling over Web sites, it is a suitable substitute of a crawler in prototype testing.

From the controller's perspective, there is only one single processing pipeline. Again, the pipeline is specified by an interface. In fact, only the actual application 'knows' that the UIMA-based processing pipeline is used. This pipeline consists of three sub-pipelines as presented in the following sections. The output of the pipeline is a data structure holding a text model as previously described in section 4.3.2. For technical reasons it proved appropriate to also include the plain text of the document in that model using a special field. This text model is then handed to an again generic indexer that is implemented as a wrapper around the Lucene search engine library.

As the actual implementations of the generic modules are only known to the application itself, they can be easily replaced by other implementations. It is unlikely that the implementation of the pipelines or the indexer will be changed in the future. However, with the escape from its prototype status, the system will certainly make use of another implementation of the collection source: a full-featured Web crawler must take over here.

Searching the index is a lot less sophisticated: a web application queries the previously constructed index. This application also takes care of loading and applying query models.

5.2 UIMA-Based Processing Pipelines

5.2.1 The Unstructured Information Management Architecture

The processing pipelines previously mentioned are subject to detailed discussion in this section. We use the Apache Unstructured Information Management Architecture (UIMA)[2] as a framework to host our analysis modules. UIMA originally was developed by IBM and introduced in Ferrucci and Lally (2004). And extensive overview is also given in Götz and Suhre (2004). Simply put, UIMA provides a skeleton for implementing any kind of data processing in pipelines. Usually these pipelines are fed with unstructured data such as text. These data are then *annotated* with deduced information until structured data can be gained. An often-used example is the one of extracting all dates from a text.

[1] `http://www.gnu.org/software/wget/`
[2] `http://incubator.apache.org/uima/`

The document's text as well as all annotations it is equipped with while passing the pipeline are stored in a Common Analysis Structure (CAS). The CAS can be seen as an instance of a platform-independent type system specifying all data structures that are used in the analyses. Such a type system must be customized for each application in question. The modules providing annotations are referred to as Analysis Engines (AEs). In addition, there is the possibility to have Aggregate Analysis Engines. These are containers holding primitive AEs. The processing order within an Aggregate AE is managed by a flow controller. In many scenarios, the flow controller simply runs the AEs in linear order. For special purposes, other orders are possible. Furthermore, it is possible do design custom flow controllers.

Collection Readers and CAS Consumers are used to read in the documents that are to be processed and to store the structured information in some form or the other. The presented prototype system does not make use of these. Instead, it instantiates a pipeline on startup and feeds it directly with CAS instances for each document.

Our system makes use of a special feature of UIMA: a CAS can be *multi-viewed*. This means that apart from the initial annotation, there are several CAS layers or views available.

5.2.2 Refactoring the New WERTi Pipeline & A Pipeline for Computing Readability

We used the NLP pipeline of the Java re-implementation of WERTi by Dimitrov (2008) as a basis. WERTi shares a few important concepts with our system: HTML pages must be fetched and analyzed. The relevant text must be extracted from the page and converted to plain text. During development, we found it useful to divide the processing in three essential steps or sub-pipelines which are illustrated in figure 5.2. The isolation of these pipelines is favored for two reasons: 1) it allows to keep apart the processing of the two formats in use, namely HTML and plain text. 2) The emerging modularity allows the sub-pipelines to be used independently, eventually allowing to use the same program code in at least two projects, namely future versions of the New WERTi and our prototype.

The UIMA type system which we use is an extension to the one of the New WERTi. Since it simply inherits from WERTi's system, it should be compatible.

The first two sub-pipelines provide the basic functionality that is used by a both our IR system and WERTi, even though only two components (asterisked in the figure) where eventually used from the original pipeline.

WERTi's way of processing HTML may seem unusual: instead of parsing the data as a browser would do, it annotates them. The *HTMLAnnotator* finds HTML tags. In addition to the original version, it also fails the pipeline if the input data are not in HTML format. Subsequently the *ParagraphSpanAnnotator* determines regions that constitute paragraphs. This means that a number of HTML tags must be mapped to the entity of a paragraph. This includes natural choices such as `<p>` or `<div>`, but also headings or list items are interpreted as paragraphs, just

to name a few. Even though they are not real paragraphs they are counted as such because the simple document representation model used later on only knows paragraphs, sentences, and tokens. The *GenericRelevanceAnnotator* is a component that is essential to WERTi and useful to our IR system. It identifies those parts of a Web page that contain the text. Advertisements and navigation menus are disregarded by this module. Eventually, the *html2plaintextMapperAnnotator* maps from the HTML code to simple plain text. The resulting text is stored in a new view of the CAS, allowing the following pipelines to be completely ignorant about HTML.

The generic NLP sub-pipeline starts out with the *LanguageChecker* module. This module is a wrapper around the Java Text Categorization Library[3], an implementation of the n-gram based language classification algorithm introduced by Cavnar and Trenkle (1994). The language checker fails the entire pipeline if the detected language is not English. The subsequent modules all are language-specific, so it would be useless to continue processing if the input language cannot be analyzed.[4] The *SpWrapperSentenceAnnotator* is a wrapper around SentParBreaker[5] by Scott Piao. As the name suggests, it annotates sentences. The following two components, *OpenNlpTokenizer* and *OpenNlpTagger*, are wrappers around the tokenizer and the POS tagger of the OpenNLP[6] project. They are based on a maximum entropy strategy. The statistical models used are the ones provided for English by the OpenNLP project. The *morphaLemmatizer* module is a wrapper around the Unix tool *morpha*, a lemmatizer for English introduced by Minnen et al. (2001). Since the analyses that are currently used do not require lemmatization, this module is deactivated.[7]

The readability sub-pipeline finally conducts all the analyses introduced in section 2. A detailed discussion of the implementation is given below in section 5.4. The *SimpleReadabilityMeasures* module computes all readability measures[8] but the Original Dale-Chall measure which is computed by the *oldDaleChall* module. The *LexicalFrequencyProfiler* computes Lexical Frequency Profiles (LFPs). All modules in the readability pipeline store their results in an UIMA-based representation of the text model in the CAS. As a final step of the pipeline, the *RelevenatText2Model* adds the text of the document to the model.

After all three sub-pipelines are run, an UIMA-independent implementation of the text model is constructed from the model in the CAS. All further processing then uses this independent implementation. For quick testing of the pipeline

[3] `http://textcat.sourceforge.net`

[4] A multi-language system to be developed in a distant future could in this step devise a sub-pipeline appropriate for the detected language.

[5] `http://text0.mib.man.ac.uk:8080/scottpiao/sent_detector`

[6] `http://opennlp.sourceforge.net`

[7] In order to fulfill the previously proclaimed synergetic effects between future versions of WERTi and our system, we left the lemmatizer module in the pipeline. It is likely that WERTi will use it.

[8] In fact, these measures were developed in a separable library in order to enable the convenient use in other projects. *SimpleReadabilityMeasures* simply wraps around this library.

during development, we created a small graphical debugging tool which includes a text model viewer (depicted in figure A.1 on page 82).

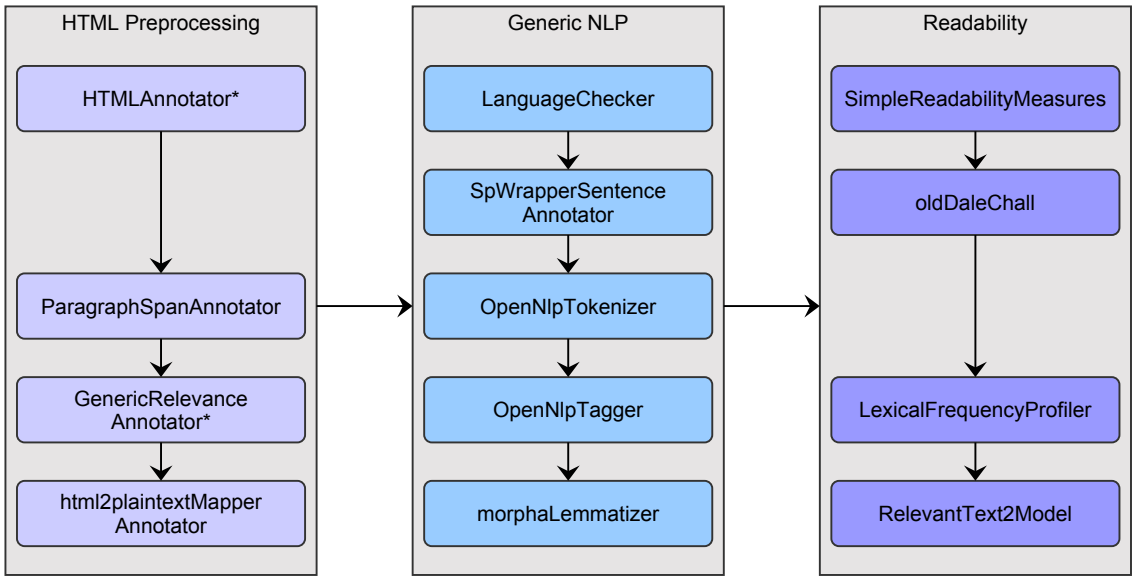

Figure 5.2: UIMA-based processing pipelines used in the prototype system (asterisked components taken from Dimitrov, 2008).

5.3 Using the Lucene Search Engine Library

5.3.1 Indexing

Apache Lucene[9] is a search engine library written in Java. An extensive introduction and programming manual is given in Gospodnetić and Hatcher (2005). Lucene implements all relevant bits and pieces of indexing and querying as described in section 3 and is convenient to use for programmers. The indexing of Lucene yields an index that can be searched in many ways, including phrase queries. The performance is state of the art, meaning that it is as fast in querying as users would expect any modern Web search engine to be.

The basic units of information in Lucene are documents and tokens. The library comes with all NLP components required for constructing a search engine index (cf. section 3.4). The analyses take place in so called *analyzers* which can also be stacked upon each other yielding a filter chain. In comparison to UIMA, Lucene constructs a true pipeline in which the output of each module replaces its input. In contrast UIMA AEs aggregate annotations and allow the access to any result deduced at any earlier step in processing.

[9] `http://lucene.apache.org`

A document in a Lucene index consists of fields containing tokens. The field containing the document text therefore must be processed by a tokenizer. Other fields such as URLs are indexed as single tokens. A common way to implement this 'knowledge' about the fields is to inherit from the default analyzer class. The resulting customized analyzer then can use other analyzers as filters, depending on the field that currently is processed.

For the presented prototype, we leave all IR-specific NLP to Lucene's *StandardAnalyzer* for English. This essentially means that some processing such as tokenization takes place twice. This is necessary for two reasons: 1) in order to find the terms in querying, the very same analysis must be conducted, however there is no UIMA-based pipeline available at that time. 2) There are different styles of tokenization and Lucene implements the style best suitable for IR. In section 5.4.1 below, we discuss different flavors of tokenization. The standard analyzer also takes care of stop words such as *the* or *of* which are usually not put into the index, because they do not constitute to the meaning or contents of a document.[10]

In addition to storing a tokenized version of the document text as found in the text model, our system also stores a version of the same text that has undergone stemming. Lucene provides an implementation of the Porter stemmer (Porter, 1997). Other fields stored in the index are technical meta-data such as the URL of the document, the document title as found in the HTML code, and the date of the last modification as given by the Web server where the document was downloaded from.

The readability scores from the text model must be stored as fields, too. Since fields can only contain string values, the numbers are converted into strings. This is implemented using the Trie encoding provided by recent development versions of Lucene (Schindler and Diepenbroek, 2008). The Trie encoding provides a string representation of numbers which is not human-readable. However, this representation allows for very fast querying for ranges of numbers. The traditional implementation of range queries in Lucene runs in a time depending on the size of the index. For a test index consisting of about 74.000 documents, a query took several seconds. This rather long processing time motivated our switch to the Trie encoding.

5.3.2 Querying

Querying the index for information at the learner's level consists of two parts: the information and the learner's level. As previously discussed, the latter is represented in query models. In our implementation, query models are defined in a simple XML-based format. An example is given in figure 5.3. The query model files contain ranges for readability measures and other scores as well as

[10]The case of the rock band *The Who* which cannot be found in the index due to stop word removal is left unaddressed in our prototype.

```
<?xml version="1.0" ?>
<querymodel version="0.9">
<meta>
  <name>Hard</name>
  <sortkey>2</sortkey>
</meta>
<entries>
  <range field="R_ARI" from="8.0" to="100"/>
  <range field="R_oldDaleChall" from="8.0" to="100"/>
</entries>
</querymodel>
```

Figure 5.3: A simple query model specified in an XML-based format.

a name presented to the user. An additional sort key is used for specifying the order in which the model names are presented in the user interface.

The query entered by the user is interpreted using Lucene's default query parser. It constructs a Java representation of the text query which is then added into a boolean query with the text model ranges. All ranges and the text query are connected with the *AND* operator. Similar to Lucene's indexing, the query parser uses an analyzer. As the analyzer produces the tokens stored in the index, the same analyzer must be used for the query in order to be able to find the tokens. Hence we use an analyzer that takes care of the text field, the stemmed text field, and the title field. Consequently, this analyzer applies the Porter stemming algorithm to the query as well. The possibility for phrase matches and direct matches must be maintained, so the stemmed query text is put into the final query in addition to the original text. The resulting query as produced by Lucene's debug output is shown in figure 5.4. The numbers in that query seem to be nonsensical. However, they were converted into the previously mentioned Trie encoding which is not human-readable.

The results are ranked by Lucene's built-in implementation of TF-IDF. Each field is equipped with a boost which is normally set to 1.0. The boost value strengthens or weakens the importance of a field for the ranking of the corresponding

```
Query: +(+(text:life title:life text.stem:life)
   +(text:sciences title:sciences text.stem:scienc))
   +trie.tm.R_ARI:[4620693217682128896 TO
       4636737291354636288]
   +trie.tm.R_oldDaleChall:[4620693217682128896 TO
       4636737291354636288]
```

Figure 5.4: Textual representation of a Lucene query for *life sciences* using the model given in figure 5.3.

document. Short fields are by default more important in Lucene. This fact makes a special boost value for the title field superfluous. For the field containing the stemmed text, we set the boost to 0.8 in our prototype. This way, a direct match of the form as entered in the query is ranked higher than an indirect match of the stemmed query.

Lucene also provides a *highlighter* as an additional component. The highlighter basically searches for prominent occurrences of the query words (or word stems) in each search hit and searches for the best text snippet representing the result. Furthermore, it is capable of inserting HTML markup that highlights the query words in that snippet.

The result to the query that served as an example in this section is depicted in figure 5.5. It is worth mentioning that our document collection of about 74.000 texts used for initial testing does not produce results that allow to draw conclusions about the system's performance. After all, the document collection is anything but well-balanced. The links labelled "debug" in the figure make the text model of the corresponding document appear on the screen. This link of course will be removed in a version for real users. The interface, although currently not overwhelmingly well-designed, is simple to use. It is only one parameter away from search engines users are familiar with: the drop-down box specifying the desired difficulty level of the documents to be retrieved.

5.4 A Modern Interpretation of Readability Measures

5.4.1 Tokens and Words

Tokenization is a solved problem. Still, it is worth being addressed here. The reason is not that the tokenizers we use generally lack performance. The question rather is: what is a token? It turns out that this depends on the interpretation or the purpose or even on conviction. A simple example can be used to highlight the issue: the previously mentioned OpenNLP tokenizer splits the input *won't* into *wo*$\oplus$*n't*. The standard analyzer in Lucene does not split it. For lookups in the word lists used in LFPs, the desired split is *won*$\oplus$*'*$\oplus$*t*.

The first style of tokenization is the one often expected by POS taggers. If *don't* is a contraction of two words, then it is logical that these are split into *do* and *n't* as a form of *not*. In the case of *won't*, this leaves us with the 'word' *wo*. But still, both components can be assigned a POS and the negation can be of importance in further processing steps such as parsing. This style of tokenization is linguistically motivated.

Lucene is not interested in understanding negation. If the user enters *won't* in the query, he or she probably is really interested in that occurrence, not in any analysis of that. If there was the request for any smarter treatment of the case,

Figure 5.5: Screen shot of the web application.

the index and the query could be expanded by *will not* in order to retrieve more documents. There is no linguistic motivation and no such motivation is needed.

We do not know about the reasoning in the Range program used for LFPs. As the word lists which we use for our implementation contain both *aren* and *t*, the interpretation of a token here is 'everything that is not a delimiter'. Furthermore, expressions such as type/token-ratio refer to tokens being words, while in NLP tokens usually include punctuation. The Original Dale-Chall readability score has entries such as *aren't* in the word list, so the contractions are single tokens or words in that interpretation.

We present these example to show a simple fact: different styles of tokenization do matter. They are an issue one should at least be aware of when implementing NLP algorithms. Sometimes a tokenizer even must be adjusted accordingly. The following section discusses more interpretations of language that inventors of readability formulas assume.

5.4.2 Readability Measures and Natural Language Processing

All readability measures discussed in section 2 once had been designed for manual application, except for the Automated Readability Index (ARI) and the Coleman-Liau Index. It is therefore important to emphasize that all scores computed by implementations of these measures are interpretations of the measures. It is not unlikely that the resulting scores differ from those presented in the original study. This is the case for three reasons: 1) the NLP used for the computations can be error-prone or simply using different linguistic interpretations (see section 5.4.1). 2) The analyses to be done by human annotators in manual applications can be error-prone because this type of work is repetitive, difficult, and tiring. 3) The use of relatively small samples in manual application can yield results different from the ones gained from an analysis of the entire text as done by implementations.

A precise implementation of readability measures would require different styles of tokenization, variants of syllable counting, and a well-motivated way of sentence segmentation—for almost each and every measure.

5.4.3 Implementing Readability Measures

The 'Simple' Measures

Our starting point for implementing readability measures was Java Fathom[11], a port of the *Lingua::EN::Fathom* module for Perl. It soon turned out that heavy modifications were necessary, so we reworked the entire library. In our implementation, all readability measures except for the Original Dale-Chall score share a common linguistic analysis and are provided by a centralized library. The syllable counter was ported from a rule-based implementation in Perl by Laura Kassner.[12]

[11] `http://www.representqueens.com/fathom/`

[12] Personal communication via e-mail in December 2008. We are not aware of any publication of the named syllable counter, neither as a paper nor as program code.

The readability library draws on the tokenization and sentence segmentation described in previous sections. Hence the interpretation of tokens is the one used in NLP in general, except for that we filter out punctuation tokens in order to only work with 'words'.

The syllable counter is ignorant towards numbers. Some measures such as Läsbarhetsindex (LIX) require to count the syllables in *1984* as five (nineteen-eighty-four) or as ten (one thousand, nine hundred and eighty-four), depending on the context. The implementation does not include that level of detail. Gunning (1968) advises not to count inflectional suffixes as syllables for his Fog Index. Again, this is a detail we omitted.

The Original Dale-Chall Score

The Original Dale-Chall score is implemented in a separate module. This module reworks the NLP-style tokenization to the version containing words as described in Dale and Chall (1948b): Tokens containing contractions are attached to the previous token. Dale and Chall also include a four-pages list of rules on how to lookup words on the Dale-Chall list of easy words. The decision whether a word is familiar or unfamiliar also depends on morphology. For example, *treatment* is supposed to be unfamiliar even though *treat* is on the word list. However, *treating* is supposed to be familiar, even though it is not on the word list. We inferred morphological rules from the rules given in natural language by Dale and Chall and implemented them using the morphology components of the BananaSplit[13], a program originally written by the author for splitting German compounds. The functionality provided by the BananaSplit morphological components is the one of removing suffixes from words and looking up the resulting words in a list at the same time.

Some rules described by Dale and Chall could not be implemented in the given time for the project. For example, proper nouns are considered familiar. Since named entity recognition is not available in our pipeline, we decided to mark those words familiar which are tagged as proper nouns by the OpenNLP tagger. The instruction to count compound names and places such as *St. John* or *Van Buren* as single words cannot be complied with. For abbreviations, only those are to be counted familiar that are present in the word list in unabbreviated form. We ignored this issue. We also ignored more esoteric rules such as the one in this example: "*Security Council*, if repeated more than twice within a 100-word sample, is counted as four unfamiliar words".

Most instructions however can be simply treated by ignoring them. For example, the supposed treatment of irregular plural forms is that they are unfamiliar if they are not on the list. Since the morphology we use is plain and simple and does not care for irregular forms, this is automatically the case.

No distinction between uppercase and lowercase spelling is made in the word list lookups.

[13] `http://www.drni.de/zap/bananasplit`

Lexical Frequency Profiles

For computing LFPs, we implemented a data structure similar to a collection of trees. The root node of each tree represents the word family and the children are members of that family. By connecting the nodes with double links, the program can easily traverse from family members to the word family and vice versa. Counting types (unique tokens) is done using a simple hash map. The implementation works on any number of word lists, given that the words on those lists are mutually exclusive.

Again, the tokenization provided by the OpenNLP tokenizer is reworked to fit the needs: contraction fragments such as *n't* from *don't* are split at the apostrophe, constructing three tokens for each contraction (e.g. $don \oplus ' \oplus t$). Punctuation tokens are subsequently removed before the LFP is constructed.

Again, no distinction between uppercase and lowercase spelling is made in the word list lookups.

6 Towards an Evaluation Strategy

'How good is it?' In this chapter we sketch strategies of evaluating our prototype Information Retrieval system. Using a large test collection obtained from various web sites, we furthermore shed some light onto the reliability of the methods for assessing readability in use.

6.1 Precision and Recall

Evaluation in Information Retrieval (IR) can be conducted by using standardized test collections and test queries (Manning et al., 2008, ch. 8). If ranking of results is disregarded, the classification of each result is binary: either the document is relevant or irrelevant. Based on this classification one can compute *precision* and *recall*, the metrics widely known in NLP:

$$\text{Precision} = \frac{\text{Number of Relevant Items Retrieved}}{\text{Number of All Items Retrieved}}$$

$$\text{Recall} = \frac{\text{Number of Relevant Items Retrieved}}{\text{Number of Relevant Items in the Collection}}$$

Reformulated in natural language, precision says how many of the retrieved documents actually were relevant. A high precision value means that many hits actually were relevant. However, precision does not say how many other relevant documents in the collection might have been left out in the search result. This is where recall comes into play: a high recall value indicates that many of the relevant documents have been retrieved. Of course, recall can still be high if a lot of irrelevant documents are in the results.

Manning et al. (2008, p. 143) write: "Typical web surfers would like every results on the first page to be relevant [...]" This claim is confirmed in an eye-tracking experiment conducted by Granka et al. (2004). Their findings indicate that users focus on the first two results: "After the second link, fixation time drops off sharply." Concluding, we can say that a good search engine should aim to have the very best result right on top of the results list. Since precision and recall are set-based, one has to built sets of top k hits in ranked retrieval. Consequently, there are k sets for which the metrics are computed (Manning et al., 2008, ch. 8.4). The findings of Granka et al. suggest to focus on the results for $k < 3$.

6.2 Beyond Precision and Recall

In order to use precision and recall for the evaluation of the presented search engine prototype, a large test collection is necessary. More important, this test collection must contain documents satisfying a variety of information needs. The language level must be given for each document. Such a corpus is to our knowledge not available at the time of writing, given that it should be annotated with CEF levels. This issue is discussed in section 8.2.

It is unlikely that a text collection large enough for evaluating our system can be devised without putting an outsized effort into the endeavor. However, once the automatic assessment of CEF levels is fully available, it is possible that this assessment can be tested and evaluated separately from the search engine. A large test collection which is well-balanced concerning language difficulty levels should then be sufficient for evaluation. The remaining question then is: what should this collection contain? After all, the search engine may perform quite well on the collection but still the collection could contain documents that are rather irrelevant to language teachers and learners.

Therefore we propose as future research to have a large enough number of human judges that test the system in a real-world application setup. As a first step, a qualitative analysis via interviews with these users can be conducted. Depending on these first results, further evaluation strategies can be devised.

6.3 A Closer Look at Readability Measures

6.3.1 A Test Collection

Since testing our prototype involved the downloading of a large amount of data from the web, we actually have a corpus. However, this corpus is not annotated with levels of text difficulty. Still, there are things one can look at without having a gold standard. First of all, we used wget to collect seven large web sites. We deliberately focused on online encyclopedias. They all should use the same genre, they are all in a way controlled language. We expect differences between the texts from the seven sites in general. For example, Simple English Wikipedia should be more readable than Britannica Online.

In a preliminary investigation of the text models gained from 229.204 documents, we found that wget had stepped into several spider traps. For example, if the crawler follows the *edit* link on a wiki page, it may always end up on the login page, but with a different URL each time. Consequently, we removed all documents having the same domain name in the URL, and the same lengths in tokens and sentences, and the same Flesch Reading Ease. It is still possible that two different documents are removed this way, but it is very unlikely to happen. The result of the duplicate removal is a set of 190.872 text models of which we took a random sample of 1.000 documents for each web site. This procedure yields a set of 7.000 text models as a basis for further statistical analyses.

6.3.2 Examining the Test Collection

There are two questions that a closer at look our 7.000 text models sample can shed light onto: 1) how do all these measures discussed relate to each other? 2) Are there measures that are good candidates for discriminating the seven web sites in the data? The latter question is especially relevant as we hypothesize that the different web sites maintain different levels of text difficulty.

The numbers given in table 6.1 show that all readability scores exhibit a rather weak correlation with the length measures (counts of characters, sentences, and tokens). This is an important finding as it shows that the measures can be used for texts of arbitrary length. It is not natural to expect this independence since the original versions of these measures were designed to be used for small samples only, not for entire texts.

	R_ARI	R_ColemanLiau	R_FORCAST	R_FleschKincaid	R_FleschReadingEase	R_FogIndex	R_LIX	R_SMOG	R_oldDaleChall
Character Count	0.50	0.27	0.03	0.42	-0.13	0.47	0.40	0.59	0.19
Sentence Count	0.23	0.07	0.01	0.19	-0.04	0.23	0.18	0.29	0.15
Token Count	0.48	0.26	0.04	0.40	-0.11	0.46	0.39	0.58	0.17

Table 6.1: Pairwise correlation scores of document lengths in characters, sentences, and tokens with all readability measures discussed.

Figure 6.1 shows the distributions of the scores computed by those six measures that yield results on the U.S. grade level scale. For each measure, there are seven curves indicating the distribution of scores for the seven web sites from which the data were downloaded. The most strikingly different scores are the ones computed by the FORCAST formula. It is not entirely clear why these scores are so far-off, but after all we abused the formula since it has been created for U.S. Army reading for young adult men.

The other five grade level-based formulas, though varying in their offsets, produce more valuable results. They all show differences in the scores for Simple English Wikipedia and the Stanford Encyclopedia of Philosophy (plato.stanford.edu). Furthermore, they all classify the Encyclopedia of Earth in the upper difficulty region, while Britannica Online and Microsoft Encarta are closer together in the lower region. The New Scientist's web site seems to use a quite controlled language since it exhibits a rather narrow peak for all five 'good' grade level formulas. These first impressions from the distributions suggest the Coleman-Liau Index as a good candidate for discriminating difficulty levels. Given that the seven web

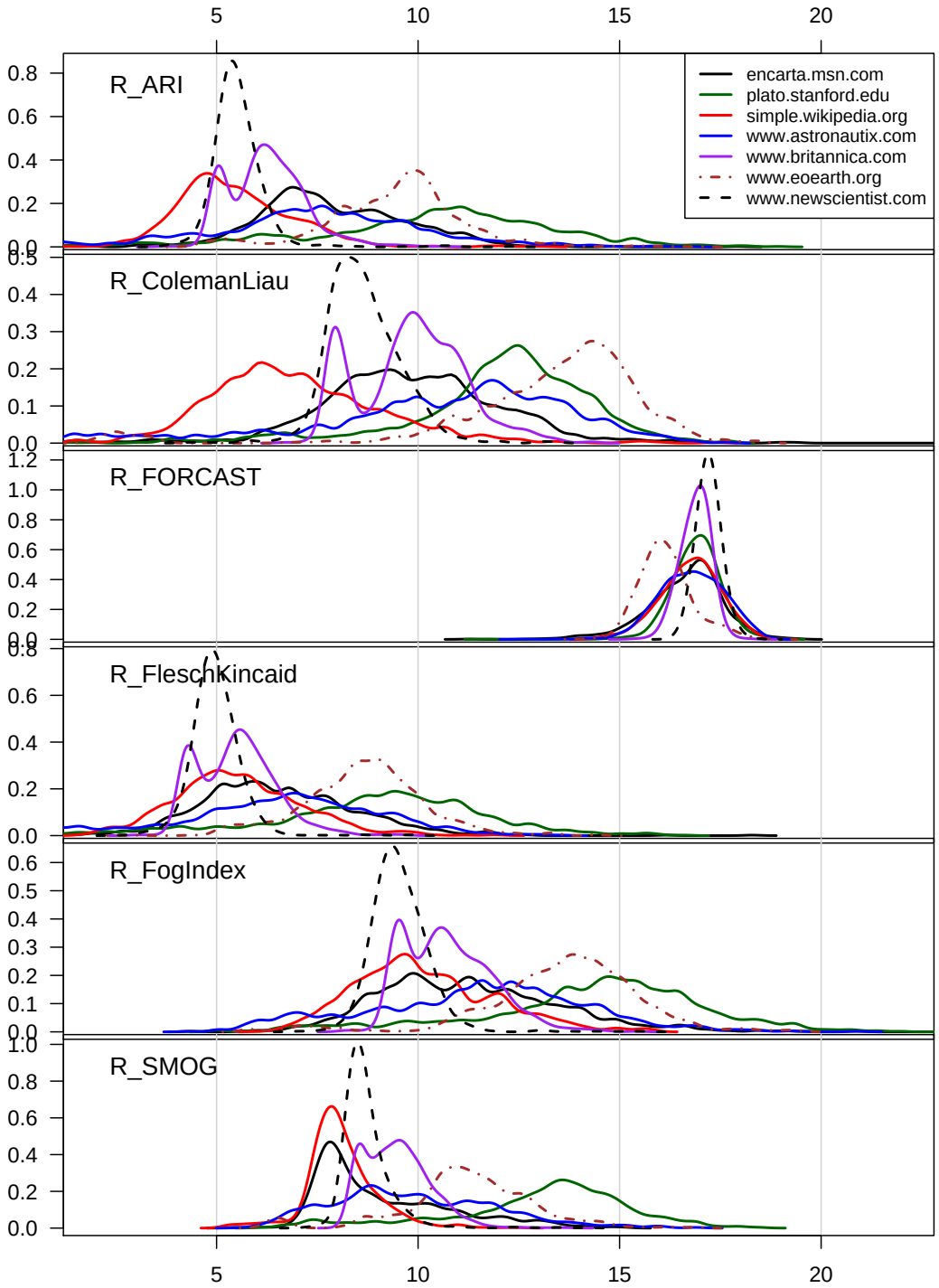

Figure 6.1: Distributions of six readability scores for seven web sites (y-axes: kernel density estimates, N=1000, bandwidth=0.2. x-axes: U.S. grade levels).

sites truly exhibit different levels of text difficulty, the Coleman-Liau Index seems to discriminate them well. In addition to that, it is also computationally more attractive because it does not use syllable counting. Syllable counting is not as accurate as counting characters. Furthermore it is also slower, which in fact is an argument, given one is processing such a vast amount of documents as it is the case in IR.

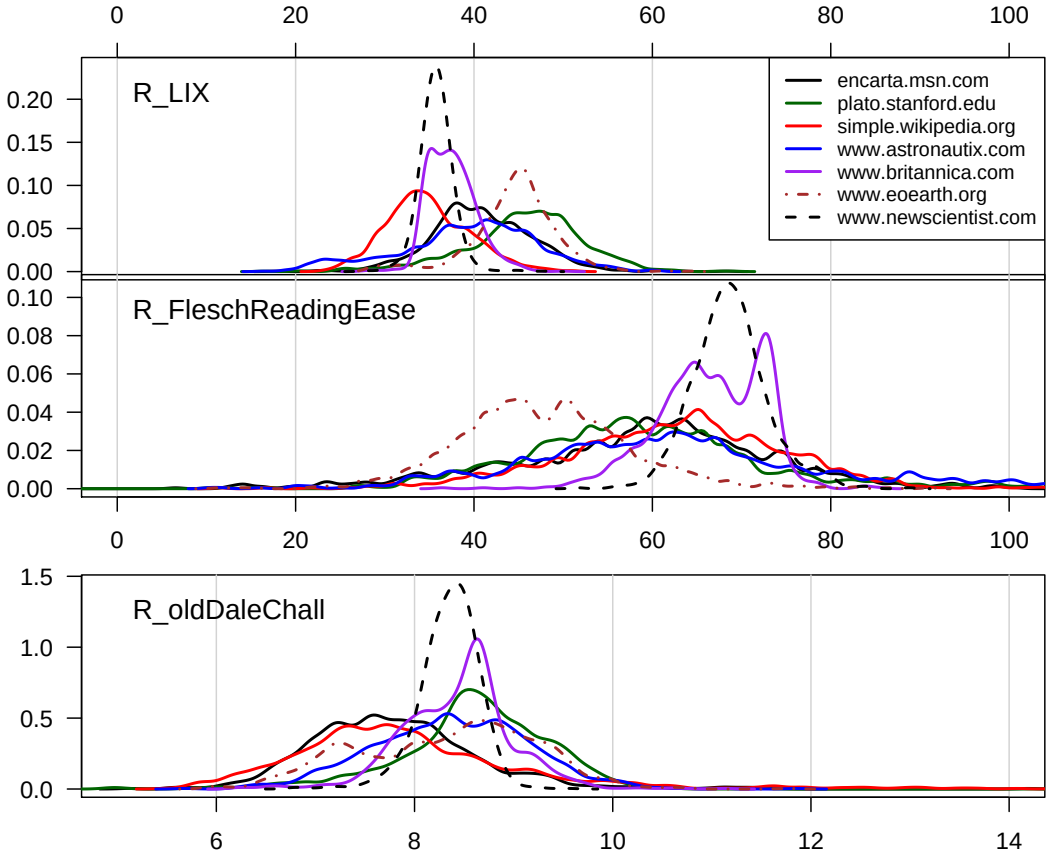

Figure 6.2: Distributions of two readability scores for seven web sites (y-axes: kernel density estimates, N=1000, bandwidth=0.8 [top, middle], bandwidth=0.1 [bottom]).

The distributions of the scores using their own scale each are depicted in figure 6.2. The Flesch Reading ease discriminates well between the Encyclopedia of Earth and Britannica Online. For the other web sites except for the New Scientist, there is no obvious distinction. LIX seems to work better, showing a pattern resembling the Coleman-Liau Index. However, LIX puts the Encyclopedia Astronautica on the same level as Encarta, while the Coleman-Liau Index puts it more towards the level of the Stanford Encyclopedia of Philosophy.

The Original Dale-Chall Score with its focus on vocabulary load draws a different picture. In Terms of vocabulary, Encarta and Simple English Wikipedia

are close, while all the others are more or less equally 'harder', indicating that they use more vocabulary from outside the Dale list of 3.000 familiar words.

The New Scientist's website again exhibits its outlying bump in all three measures shown in figure 6.2.

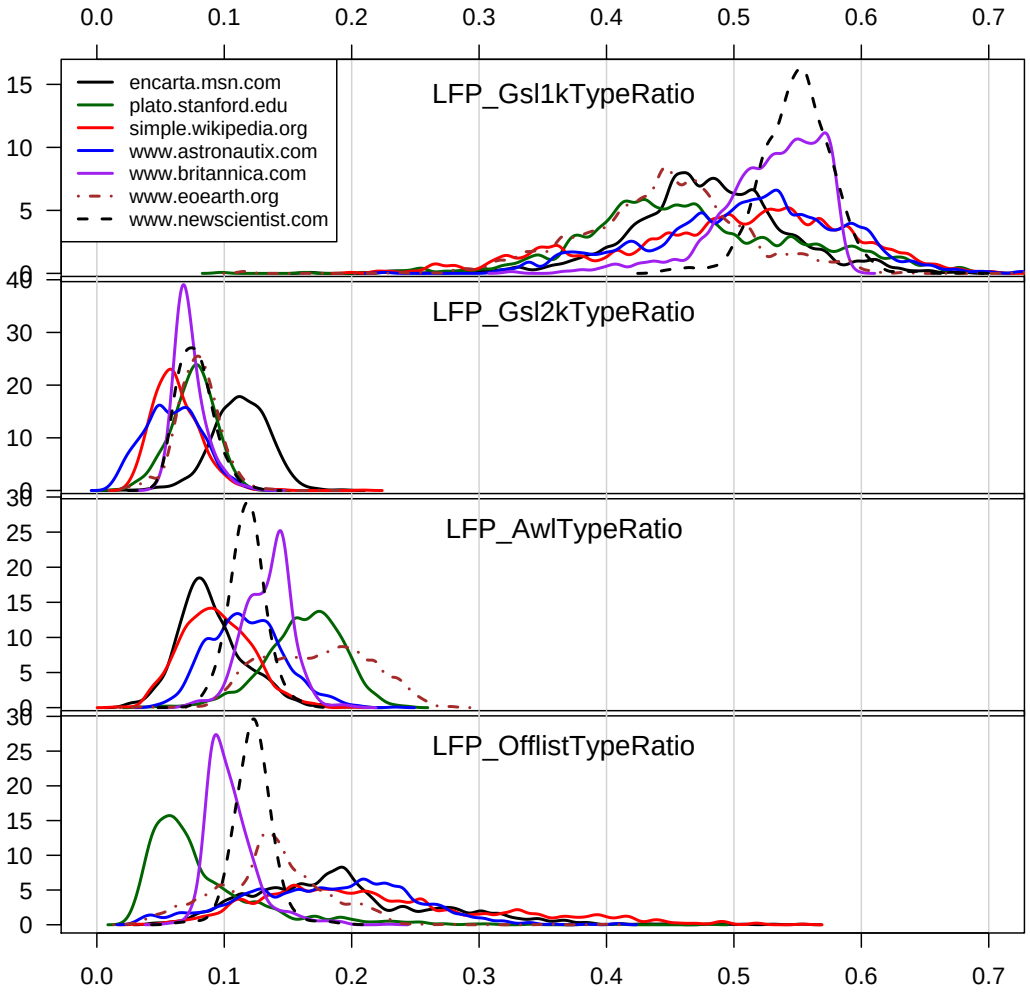

Figure 6.3: Distributions of LFP ratios (types on a word list to all types) for seven web sites (y-axes: kernel density estimates, N=1000, bandwidth=0.005. x-axes: ratio).

More hints about vocabulary are given in the distributions of the Lexical Frequency Profiles (LFPs) shown in figure 6.3. It is important to notice that these four distributions are not independent, since the word lists do not overlap. The ratios displayed can be read as answers to the question: 'how many types found in the text were on the word list?' The first 1.000 words of the General Service List (GSL) cover many types found in the texts, while the second 1.000

words, the words on the Academic Word List (AWL) and all the rest (off-list) are approximately equally less-used.

Except for Britannica Online and the New Scientist, all web sites use a lot of vocabulary from the GSL 1k band. Encarta's texts draw on GSL 2k band, while the others are more or less similar there. The more revealing results can be observed on the AWL band. It shows that Encarta and Simple English Wikipedia use less academic language than the Encyclopedia Astronautica. Britannica Online, the Encyclopedia of Earth, and the Stanford Encyclopedia of Philosophy use even more academic words.

The off-list distribution also gives hints on the word lists themselves: While the types in the Stanford Encyclopedia of Philosophy and Britannica Online are found on the AWL and therefore few of them are off-list, the Encyclopedia Astronautica uses a lot of off-list tokens. This is not so surprising since a lot of special terms from astronautics are used. It is unclear why Encarta and Simple English Wikipedia use almost the same amount of off-list vocabulary.

Coming back to the desired borderline between the seven web sites, one can say with respect to LFPs that the most discriminative figures are produced by the ratio of types on the AWL. However, the relatively high amount of off-list vocabulary in Simple English Wikipedia and Microsoft Encarta hint to the direction that there may be other discriminative vocabulary bands that are covered by none of the three word lists.

Concluding, we can say that based on the results of our test collection and the assumption that the seven web sites under investigation contain texts of different difficulty levels, the Coleman-Liau Index seems to be the most promising discriminative measure among the traditional readability formulas. Another interesting candidate among those is the Läsbarhetsindex. Looking at vocabulary, the type ratio from the AWL in LFPs is a more promising measure than the Original Dale-Chall Score. However, this must be taken with a grain of salt since the high amount of off-list vocabulary for certain web sites hint to the direction that there is something beyond the scope of this measure that must be investigated.

7 Related Work

7.1 Text Categorization

Tzoukermann et al. (2003, p. 541) write about technologies related to Information Retrieval (IR): "Two related technologies include text categorization and question answering. Text categorization refers to technologies to determine whether a document is a member of a given category." While we exclude question answering from our definition of IR given in section 3.1, we cannot deny that the system discussed in the presented thesis is related to text categorization (TC).

There are two fundamental paradigms in TC: *single-label* TC assigns exactly one category to each document, while *multi-label* TC assigns any number of categories to each document (Eichler, 2005). In the early days of TC, people focused on rule-based approaches with manually defined rules, a strategy that later on was replaced by *machine learning* approaches (Sebastiani, 2002).

A remarkable example of a TC approach is presented by Cavnar and Trenkle (1994): using n-gram profiles of documents of known categories, they successfully re-categorize test data unknown to the system to those categories. For language classification, they report an overall classification rate of 99.8%.[1] In the same publication, Cavnar and Trenkle discuss the application of their approach to *subject classification*. In an evaluation experiment they successfully re-categorize posts to their original Usenet groups using lists of FAQ as training data.

Joachims (1998) presents an approach of TC using machine learning. He counts each word with a term frequency of three or more as a feature, with the frequency being the value. Using a training corpus with known labels, he successfully re-classifies the test-data by using Support Vector Machines (SVMs).

Coming back to the presented IR system for language learning, one can first of all say that what we present in fact includes a text categorization system. The categories are levels of language proficiency required from a reader to understand a certain text. It is a multi-label system as the borders between a level system such as Common European Framework (CEF) may be fuzzy. It is also imaginable to have simple categories such as *easy*, *hard* and *academic*, where most likely hard and academic will overlap in many documents.

Values for readability scores and other measures can be derived automatically from annotated documents (cf. sec. 8.2). However, manual intervention is not unlikely to happen when a query model is created. This allows the manual merging of categories. For example, if there is a corpus annotated with CEF levels,

[1] We use an implementation of their approach for language classification in our prototype IR system, see section 5.2.2.

we may be able to derive a query model from that automatically. There can be another corpus that is annotated with information about the appropriateness of documents (cf. sec. 4.1) for language learners from which we may be able to derive another component that can be simply added to existing query models.

To conclude: the presented search engine prototype includes a specialized version of multi-label text categorization that in its final version will use a hybrid approach consisting of automatic corpus analyses and manually specified rules.

7.2 Similar Approaches

7.2.1 REAP Search

The REAP system described by Heilman et al. (2008) aims at a similar task: retrieving reading materials that are appropriate for learners. However, there are some fundamental differences in their work.

Our system is a search engine prototype. A search engine in our understanding can be used to retrieve data from the Web according to certain criteria. Apart from a possible choice of websites that are included or excluded from crawling, it is a tool for freely obtaining information, with certain restrictions made on purpose, such as returning readable texts or disregarding texts containing profanity.

REAP puts the focus on a different spot: it is used for constructing a digital library of suitable readings. The 'books' in that library are gained from the web via query-based crawling. They submit queries created from target words to the AltaVista search engine.[2] Before a text is presented to the language learner, it is controlled by a human instructor. The text is then served via a specialized reading interface that includes further functionality such as displaying dictionary definitions for words on request. In our scenario, the New WERTi system could be engaged as a reading interface, given that functions such as dictionary would be implemented. Web pages could simply not be linked to directly but via WERTi.

The assessment of text difficulty in REAP is done using a machine learning approach by Collins-Thompson and Callan (2005). This approach focuses on vocabulary as an indicator for readability. They claim that it is superior to other readability measures when evaluated with human-assigned difficulty levels. Their solution seems to be performing well, but it lacks the versatility of our system: what if the query should ask for documents containing exceptionally many occurrences of conditionals or gerunds because the learner is supposed to be confronted with these phenomena in order to learn them? Given an appropriate analysis of the documents, we can easily construct a query model that filters for these criteria.

It must be noted though, that the harder the constraints on the documents are, the larger the document collection must be in order to contain enough documents

[2] They do not report on the relation of their approach to the bootstrapping of Web corpora, cf. Baroni and Bernardini (2004).

that fulfill them. Therefore we are of the opinion that crawling 'all the web'—or at least a reasonably large part of it— is superior to creating a digital library.

What REAP does and what our prototype does not is to classify the documents into topics using a machine learning approach. Heilman et al. used pages listed and categorized in the Open Directory Project[3] as training data. Their system performs multi-labeled TC on the input documents based on these training data. Their motivation for doing this classification is that it allows for obtaining documents on a given topic without specifying a too restrictive textual query.

To summarize: Heilman et al. (2008) present a system that is targeted more towards classroom use. The reading materials are selected by an instructor using the REAP Search system and subsequently presented to the students via a reading interface. Their system bases readability on vocabulary load. Our system is targeted more towards learners themselves. It aims to allow them to find whatever information they want—yet at a level that they can understand. Since our text categorization approach is more versatile, it is also more complex. However, it comes with the potential to retrieve documents matching a large variety of criteria of language learning: from vocabulary load (via LFPs) over general readability (readability measures) up to specific grammatical phenomena (via parsing, to be implemented, see section 8.1.3).

7.2.2 Read-X and Toreador

Read-X and Toreador are two interacting but independent components introduced by Miltsakaki and Troutt (2008). Their design is similar to the one of REAP. The most important difference is that Read-X obtains the texts of interest on-line via the Application Programming Interface (API) of the Yahoo! search engine[4]. All educationally motivated processing is done on-line instead of during indexing as in REAP or in our system. Toreador includes a function that is similar to the reading interface described for REAP. For words that are supposed to be difficult for the reader, Toreador includes a dictionary assistant based on WordNet (Fellbaum, 1998).

Read-X is able to extract plain text from the formats HTML, PDF, XML, and Microsoft Word. For assessing the difficulty of text, Miltsakaki and Troutt use the LIX formula and the Coleman-Liau index (previously discussed in section 2.2). Furthermore they use the RIX formula by Anderson (1983), which is based on LIX. Miltsakaki and Troutt conclude that the measures they use have shortcomings and that more sophisticated measures must be developed. Similar to our goals, they aim at constructing models of text difficulty consisting of multiple factors.

Their approach of using a commercial search engine reveals certain benefits. The most salient one probably being that it does not require a centralized crawler and indexer to be run on a server. Indexing and using Natural Language Processing (NLP) techniques as done in our system consume much more processing power

[3] `http://www.dmoz.org`
[4] `http://de.yahoo.com`

and storage space. Read-X leaves this to the user's machine. However, it may also be inconvenient for users to wait until the program has downloaded and analyzed all hits retrieved from Yahoo!. While controlling the crawling and indexing oneself is tedious and requires a significant amount of additional expertise, leaving it to a commercial search engine also means giving control over the delivered contents to that search engine.

8 Future Work

The first insights presented in chapter 6 (evaluation) as well as the related work suggest that there is much more to explore. In particular, our assessment of readability ignores sentence structure. In this chapter, we focus on possibilities for using syntactic complexity and on the necessary steps that must be taken in order to assess text difficulty in terms of levels widely used in educational systems.

8.1 Syntactic Measures

8.1.1 Introduction

The method for assessing reading difficulty in the presented thesis is based on readability measures and Lexical Frequency Profiles (LFPs). The Original Dale-Chall Score and the LFPs measure vocabulary load. The other measures try to assess a general complexity score from surface information such as sentence length, or word length in syllables or characters. Sentence length and word length are indicators of syntactic and morphological complexity. Longer sentences are on average more likely to consist of a more complex syntactic structures than short sentences. Long words are often more rare than short words, thus they indirectly hint to the amount of vocabulary load.

From the perspective of language learning, it is desirable to include a more direct measure of syntactic complexity than the one of counting sentence length. After all, it seems natural that the difficulty of a sentence depends on the type of its syntactic constituents, not only on their sheer number. Future work should therefore investigate how syntactic measures can constitute to the assignment of general levels of language difficulty to text documents. The following sections aim at providing a brief selection of relevant existing approaches and at establishing the relation to the special scenario of language learning.

8.1.2 Measuring Syntactic Complexity

Lu (2009) presents an implementation of 14 different measures of lexical complexity based on previous work by various researchers. These 14 measures are grouped into five types. An overview is provided in figure 8.1. According to the definition as used by Lu, a sentence is a group of words ending with sentence-final punctuation. A clause is "a structure with a subject and a finite verb" and a T-unit is "a main clause plus any subordinate clauses". These definitions are based on Hunt (1965).

Type	Measure
Length of production	Mean length of clause Mean length of sentence Mean length of T-unit
Sentence complexity	Mean number of clauses per sentence
Subordination	Mean number of clauses per T-unit Mean number of complex T-units per T-unit Mean number of dependent clauses per clause Mean number of dependent clauses per T-unit
Coordination	Mean number of coordinate phrases per clause Mean number of coordinate phrases per T-unit Mean number of T-units per sentence
Particular structures	Mean number of complex nominals per clause Mean number of complex nominals per T-unit Mean number of verb phrases per T-unit

Table 8.1: Measures of syntactic complexity automatically computed and evaluated by Lu (2009).

Lu bases his implementation on the Stanford Parser (Klein and Manning, 2003). After the parsing step, the resulting syntactic trees are examined for subtrees constituting the units required by the analysis. Lu conducted an evaluation experiment on ten randomly selected documents from the Written English Corpus of Chinese Learners, a corpus containing college-level essays by ESL learners. These ten documents were analyzed by two human annotators. They counted all relevant units such as clauses and sentences. The system's performance correlated with values ranging from $r = .896$ to $r = 1.0$ with the human counts.

We found eight of the 21 figures contained in an LFP to be candidates for assessing the difficulty level of a text (see discussion in section 2.3.2). The question for our purpose of assessing readability levels to documents now is: which of the 14 measures computed by Lu's implementation are useful for this intent? The named corpus is equipped with school level information for each document. Lu reports on a number of measures that discriminate successfully between school levels, but apparently there is no measure that puts these levels on a single scale. As with LFPs, it is fairly possible that a number of these individual measures of syntactic complexity will be useable in a query model for a given level of language difficulty. Yet, for another level of difficulty, one may want to use other measures.

In both cases, the question of which measures to use must be subject to future research. Furthermore, the relation between production and perception remains subject to discussion in both cases: neither LFPs nor the syntactic measures have been designed to measure perceptional abilities required from the readers, as opposed to readability measures.

8.1.3 Teaching Sequence of Linguistic Structures

There is a sequence or order in which linguistic structures are presented to learners by language teachers. Since the users of an Information Retrieval (IR) system for language learning base their abilities on an educational system, it is worth examining which linguistic forms they are able to handle at which level. To have such a functionality could greatly improve the usefulness of the system for school use.

The Revised D-Level Scale by Covington et al. (2006) provides six levels of language development. Lu (2008) shows that it is possible to measure these levels automatically. However, there are two issues: first, the D-level Scale is designed for first language acquisition (FLA). Whether or not the acquisition order of phenomena agrees with sequences practically used in teaching (second language acquisition) must be examined carefully. Second, the D-Level Scale again is a scale for measuring production, not perception.

Unit	Structures taught
1	Present perfect progressive with *since* and *for* Past perfect progressive Attributive use of adjectives after nouns Adverbs of degree
2	Perfect infinitive with modal verbs Passive infinitive with full verbs and modals
3	Gerund as subject, object, and after verbs and adjectives with prepositions Object plus *-ing* form Present and past progressive passive Passive with verbs with prepositions
4	Verb plus object plus infinitive Infinitive after question words and after superlatives Infinitives vs. Gerund
5	Non-defining relative clauses Participles as adjectives

Table 8.2: Teaching Sequence of Linguistic Forms from a German text book for English as a second language learners (Weisshaar, 2008).

We suggest to use the sequence of teaching linguistic forms as it is actually used in educational systems. As an example, the order of forms taught in the fourth year of ESL teaching according to the corresponding English text book Weisshaar (2008) is given in table 8.2. The textbook is organized in units. Apart from readings, exercises, and vocabulary, each unit targets specific linguistic forms. For example, unit 1 deals with the present perfect progressive, while the gerund and the progressive passive are dealt with in unit 3. Progressive forms

require the learner to be able to generally handle *-ing* forms. With the gerund, another type of *-ing* form occurs.[1]

A limited analysis of linguistic forms could cover the counting of tenses of verbs in a text. Language courses usually start with the present tense, continuing with simple future and simple past. Structures such as gerunds and future-in-the-past are taught at higher levels. Ott and Ziai (2008) have shown the general possibility to distinguish different types of *-ing* forms using a grammar for the *vislcg3* constraint grammar parser[2], the successor of *cg-2* (Tapanainen, 1996) developed by Tino Didriksen and Erhard Bick at Syddansk Universiteit. While this is a special case because most tag sets and hence most POS taggers do not make any distinction between gerunds and progressives and other *-ing* forms, their implementation as a side-effect also detects going-to future (in the past), progressive forms, and participles. With an extension to their grammar, the detection of other tenses should be possible as well.

Once the occurrences of tenses are identified in a text, their ratio to the count of all verb forms in the text can be computed. Hence there is a single value for each and every tense, allowing to construct a query model that allows for the retrieval of documents containing a significant amount of occurrences of the respective tense. This is of interest to applications such as the New WERTi, which can be used for the training of specific linguistic forms.

Coming back to the general theme of constructing query models, the detection of tenses could also help to distinguish levels of text difficulty.

8.2 Mapping Readability to Levels of Language Difficulty

In order to query for levels of language difficulty using our approach, there must be a query model for each such level. Since query models can consist of multiple constraints for readability measures and other automatic judgements, there are two questions to answer: 1) which are the measures or judgements that are useful? 2) What are the ranges for these measures that map to a level of language difficulty?

The answers to these questions can be found using a corpus. For our particular interest, the documents in this corpus should be annotated with CEF levels. We foresee two major challenges for the endeavor of creating such a corpus: 1) the corpus must contain enough text ranging from the simplest to the most advanced levels. It is likely that it will be an issue to find enough material at the simple language level. 2) The annotation of CEF levels of text is not discussed in the key publication (Council of Europe, 2001). A suitable annotation manual must be

[1] While there are grammarians such as Huddleston and Pullum (2006, p. 1220) who refuse the existence of a pure gerund, it is used in ESL. Therefore we simply face the facts and adhere to the gerund.

[2] `http://giellatekno.uit.no/doc/tools/docu-vislcg3.html`

devised in cooperation with language teachers who know how to judge the level of a text.

Statistical analyses on such a corpus can be used to infer the ranges in text models. Furthermore, the reliability of readability measures and other judgments can be assessed: the 'good' measures should all be in the same range for each annotated level. It is likely that some measures are predictive only on high or low levels. The concept of an underspecified query model allows to use the measure that is most predictive for a given level.

An additional question to be answered by a large-enough corpus is the one of how different text types or genres affect readability measures. One can think of two possible outcomes: either, genres correlate with difficulty as annotated by humans. That is to say that certain genres are more difficult than other ones. Or they do not correlate, a case in which readability formulas might turn out to fail in judging text difficulty reliably across genres.

8.3 Production vs. Perception

Readability measures are supposed to judge the reading difficulty of a text. In other words, they measure the difficulty of a perceptional task. LFPs are designed to analyze text written by language learners. Hence they produce information about the productional abilities of the learner, in particular about the vocabulary he or she uses actively. Syntactic complexity measures based on the Revised D-Level Scale as described in section 8.1.2 refer to production as well. Even more, they aim to reflect the development in FLA. A question to be answered by further research is how the relation between perception and production is established in terms of measures. For example, is it safe to use the D-Level Scale to judge reading difficulty of texts?

As previously mentioned in section 2.3.2, Laufer and Goldstein (2004) have shown in a study that active and passive vocabulary in learners are correlated. A small increase in the learner's active vocabulary implies a large increase of his or her passive vocabulary. Naturally, the passive vocabulary is what is used in reading. Therefore it is possible to use LFP as indicators for readability. Of course, the numbers in the LFP are not comparable to those computed for production.

For the method of measuring syntactic complexity that has yet to be defined for the purpose of IR for Intelligent Computer-assisted Language Learning (ICALL), the questions of production vs. perception and perhaps even second language acquisition vs. first language acquisition must be addressed.

9 Conclusion

In the presented thesis, we discussed several text difficulty measures and their automatic application on documents indexed by a search engine prototype. We have shown a system that serves as a basis for retrieving documents at the proficiency level of a language learner. Such a system can be used in various scenarios, either simply to provide understandable information, or in an educational environment where interesting and up-to-date reading materials can be a benefit over traditional, constructed texts.

We have found that a combination of text models holding the difficulty scores as a key-value table and query models formalizing language proficiency levels in terms of allowable ranges of difficulty scores can be used to create a simple additional parameter in the search engine's interface. This simple parameter reflects a language proficiency level given in a system that the learner is aware of. We suggest to use CEF levels as they are becoming increasingly popular in language teaching in Europe.

The measures explored include traditional readability scores as well Lexical Frequency Profiles (LFPs; Laufer and Nation, 1995). This includes nine traditional readability measures of which eight are based on surface indicators such as average sentence or word length, and one is based on vocabulary load. LFPs were originally designed for measuring active vocabulary, but we found that they also can be used to measure the required passive vocabulary required from learners by a text. It is currently unclear which combinations of traditional readability scores, LFPs, and other yet to be explored measures—such as those of syntactic complexity—will eventually lead to the successful classification of texts into the levels specified by the Common European Framework (CEF). Therefore we suggest this mapping to be investigated in future research.

In a preliminary evaluation experiment, we built a test collection of almost 200,000 unique documents from the Web. From these data, a random sample of seven online encyclopedias with 1,000 documents each was examined. We hypothesize that these seven encyclopedias contain language of different levels. Looking at the text models of these 7,000 documents, we found that of the nine traditional readability measures under investigation, the Coleman-Liau Index and the Läsbarhetsindex yield results that are likely to allow the discrimination of the seven web sites. Concerning vocabulary load imposed on learners as measured by our specific use of LFPs, we found the ratio of types on the Academic Word List to all types in the text to be likely to be discriminative.

Related approaches combine the retrieval of documents with a reading interface that provides supportive functions such as a built-in dictionary. For our approach, this functionality can be provided by an additional system such as the New

WERTi (Dimitrov, 2008). Our approach turned out to be a more general text categorization strategy that is specialized to the determination of text difficulty levels. Other features of interest in language learning can be implemented using the presented query model strategy as well. For example, it is imaginable to find documents that contain a significant amount of occurrences of a linguistic phenomenon that the learner is to practice in a teaching unit.

Concluding, we can say that we have shown a promising track to follow in order to provide text at the learner's level. An important milestones yet to be reached is the identification of the language proficiency required by a document in terms of a standardized and widely-used system such as the one of CEF levels.

Bibliography

Amaral, L. and Meurers, D. (2008). From recording linguistic competence to supporting inferences about language acquisition in context: Extending the conceptualization of student models for intelligent computer-assisted language learning. *Computer-Assisted Language Learning*, 21(4):323–338.

Anderson, J. (1983). Lix and rix: Variations on a little-known readability index. *Journal of Reading*, 26(6):490–496.

Arbeitskreis der Sprachenzentren, Sprachlehrinstitute und Fremdspracheninstitute (2006). Rahmenordnung UNIcert.

Baroni, M. and Bernardini, S. (2004). Bootcat: Bootstrapping corpora and terms from the web. In *Proceedings of LREC 2004*.

Bauer, L. and Nation, P. (1993). Word families. *International Journal of Lexicography*, 6(4):253–279.

Björnsson, C.-H. (1968a). *Läsbarhet*. Lärarbiblioteket. Liber, Stockholm.

Björnsson, C.-H. (1968b). *Lesbarkeit durch Lix*. Pedagogiskt Centrum i Stockholm, Stockholm.

Cavnar, W. B. and Trenkle, J. M. (1994). N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US.

Caylor, J. S., Sticht, T. G., Fox, L. C., and Ford., J. P. (1973). Methodologies for determining reading requirements of military occupational specialties: Technical report no. 73-5. Technical report, Human Resources Research Organization, Alexandria, VA.

Coleman, M. and Liau, T. (1975). A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60:283–284.

Collins-Thompson, K. and Callan, J. (2005). Predicting reading difficulty with statistical language models. *Journal of the American Society for Information Science and Technology*.

Council of Europe (2001). *Common European Framework of Reference for Languages*. Cambridge University Press, Cambridge.

Covington, M. A., He, C., Brown, C., Naçi, L., and Brown, J. (2006). How complex is that sentence? a proposed revision of the Rosenberg and Abbeduto D-Level Scale. CASPR Research Report 2006-01, The University of Georgia, Artificial Intelligence Center, Athens, GA.

Coxhead, A. (2000). A new academic word list. *Teachers of English to Speakers of Other Languages*, 34(2):213–238.

Dale, E. and Chall, J. S. (1948a). A formula for predicting readability. *Educational research bulletin; organ of the College of Education*, 27(1):11–28.

Dale, E. and Chall, J. S. (1948b). A formula for predicting readability: Instructions. *Educational research bulletin; organ of the College of Education*, 27(2):37–54.

Dimitrov, A. (2008). Rebuilding WERTi: Providing a platform for second language acquisition assistance. Technical report, Seminar für Sprachwissenschaft, Universität Tübingen.

DuBay, W. H. (2004). *The Principles of Readability*. Impact Information, Costa Mesa, California. Report published online.

Eichler, K. (2005). Automatic classification of Swedish email messages. Bachelor of Arts Thesis, University of Tübingen.

Evetts, J. and Gauthier, M. (2005). *Literacy Task Assessment Guide*. National Literacy Secretariat, Canada.

Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Ferrucci, D. and Lally, A. (2004). UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3–4):327–348.

Flesch, R. F. (1943). *Marks of Readable Style: A Study in Adult Education*. PhD thesis, Columbia University.

Flesch, R. F. (1948). A new readability yardstick. *Journal of Applied Psychology*, 32(3):221–233.

Gospodnetić, O. and Hatcher, E. (2005). *Lucene in Action*. Manning, Greenwich, CT.

Götz, T. and Suhre, O. (2004). Design and implementation of the UIMA Common Analysis System. *IBM Systems Journal*, 43(3):476–489.

Granka, L. A., Joachims, T., and Gay, G. (2004). Eye-tracking analysis of user behavior in WWW search. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 478–479, New York, NY, USA. ACM.

Gunning, R. (1968). *The Technique of Clear Writing*. McGraw-Hill Book Company, New York, 2nd edition.

Guoyi, X. and Nation, P. (1984). A university word list. *Language Learning and Communication*, 3(2):215–229.

Heift, T. (2003). Multiple learner errors and meaningful feedback: A challenge for ICALL systems. *CALICO Journal*, 20(3):533–548.

Heift, T. and McFetridge, P. (1999). Exploiting the student model to emphasize language teaching pedagogy. In *Natural Language Processing. Computer-Mediated Language Assessment and Evaluation in Natural Language Processing, ACL/IALL*, pages 55–62.

Heilman, M., Zhao, L., Pino, J., and Eskenazi, M. (2008). Retrieval of reading materials for vocabulary and reading practice. In *Proceedings of the 3rd Workshop on Innovative Use of NLP for Building Educational Applications*, Columbus, Ohio.

Huddleston, R. and Pullum, G. K. (2006). *The Cambridge grammar of the English language*. Cambridge University Press, Cambridge, 4th edition.

Hunt, K. W. (1965). Grammatical structures written at three grade levels. NCTE Research Report No. 3.

Jakobsen, G. (1971). *Dansk LIX 70*. Landsforeningen af læsepædagoger. Dragør, København.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. In Nédellec, C. and Rouveirol, C., editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142.

Kincaid, J. P., Fishburne, R. P. J., Rogers, R. L., and Chissom, B. S. (1975). Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease formula) for Navy enlisted personnel. Research Branch Report 8-75, Naval Technical Training Command, Millington, TN.

Klare, G. R. (1963). *The Measurement of Readability*. Iowa State University Press, Ames, Iowa.

Klare, G. R. (2000a). The measurement of readability: useful information for communicators. *ACM Journal of Computer Documentation*, 24(3):107–121. Reprint of the first chapter of Klare (1963).

Klare, G. R. (2000b). Readable computer documentation. *ACM Journal of Computer Documentation*, 24(3):148–168.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430, Morristown, NJ. Association for Computational Linguistics.

KWMBl (2008). Amtsblatt der Bayerischen Staatsministerien für Unterricht und Kultus sowie Wissenschaft, Forschung und Kunst.

Laufer, B. and Goldstein, Z. (2004). Testing vocabulary knowledge: Size, strength, and computer adaptiveness. *Language Learning*, 54(3):399–436.

Laufer, B. and Nation, P. (1995). Vocabulary size and use: Lexical richness in L2 written production. *Applied Linguistics*, 16(3):307–322.

Lu, X. (2008). Automatic measurement of syntactic complexity in second language acquisition. Slides of talk presented at the CALICO-08 Pre-Conference Workshop on *Automatic Analysis of Learner Language: Bridging Foreign Language Teaching Needs and NLP Possibilities*. San Francisco.

Lu, X. (2009). Automating measures of L2 syntactic complexity. Paper presented at the Workshop on Automatic Analysis of Learner Language at the 26th Annual Symposium of the Computer Assisted Language Instruction Consortium (CALICO09). Tempe, AZ. March 1014.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge.

McCallum, D. R. and Peterson, J. L. (1982). Computer-based readability indexes. In *ACM 82: Proceedings of the ACM '82 conference*, pages 44–48, New York, NY, USA.

McLaughlin, G. H. (1969). SMOG grading – a new readability formula. *Journal of Reading*, 12(8):639–646.

Miltsakaki, E. and Troutt, A. (2008). Real time web text classification and analysis of reading difficulty. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 89–97, Columbus, Ohio. Association for Computational Linguistics.

Minnen, G., Carroll, J., and Pearce, D. (2001). Applied morphological processing of english. *Natural Language Engineering*, 7(3):207–223.

Murphy, M. and McTear, M. (1997). Learner modelling for Intelligent CALL. In *Proceedings of The 6th International Confeence on User Modeling*, pages 301–312, Sardinia, Italy.

Ott, N. and Ziai, R. (2008). ICALL activities for gerunds vs. to-infinitives: A constraint-grammar-based extension to the New WERTi system. Unplublished term paper for the course *Using Natural Language Processing to Foster Language Awareness in Second Language Learning* taught in summer 2008 at Tübingen University by Detmar Meurers.

Porter, M. F. (1997). An algorithm for suffix stripping. In *Readings in information retrieval*, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Salton, G. and McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill Inc, New York.

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

Schindler, U. and Diepenbroek, M. (2008). Generic XML-based framework for metadata portals. *Computers & Geosciences*, 34(12):1947–1955.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Smith, E. A. and Senter, R. J. (1967). Automated Readability Index. Technical Report AMRL-TR-66-220, Aerospace Medical Research Laboratories, Wright-Patterson Airforce Base, OH.

Tapanainen, P. (1996). *The Constraint Grammar parser CG-2*. Number 27 in Publications of the Department of General Linguistics. University of Helsinki.

Tzoukermann, E., Klavans, J. L., and Strzalkowski, T. (2003). Information retrieval. In Mitkov, R., editor, *The Oxford Handbook of Computational Linguistics*, chapter 29, pages 529–544. Oxford University Press.

UK Parliament (2002). Education act 2002.

Weisshaar, H. (2008). *Green Line Schülerbuch 4 – Band 4: 8. Klasse*. Ernst Klett, Stuttgart, Germany.

West, M. (1953). *A General Service List of English Words*. Longmans, London.

Zipf, G. K. (1936). *The Psycho-Biology of Language*. Routledge, London.

Zipf, G. K. (1949). *Human behavior and the principle of least effort*. Addison Wesley.

A Appendix

A.1 Supplementary Tables and Figures

Measure/Method vs. Required Analyses	Word Count	Type Count	Character Count	Syllable Count	Sentence Count	Word List(s)	Syntactic Analysis
Original Dale-Chall	✓				✓	✓	
Flesch Reading Ease	✓			✓	✓		
Flesch-Kincaid	✓			✓	✓		
Gunning Fog Index	✓			✓	✓		
Simple Measure of Gobbledygook	(✓)			✓	✓		
Läsbarheitsindex	✓		✓		✓		
Coleman-Liau Index	✓		✓		✓		
Automated Readability Index	✓		✓		✓		
FORCAST	✓			✓			
Lexical Frequency Profiles	✓	✓				✓	

Table A.1: Comparison of measures and methods discussed: the required (linguistic) analyses.

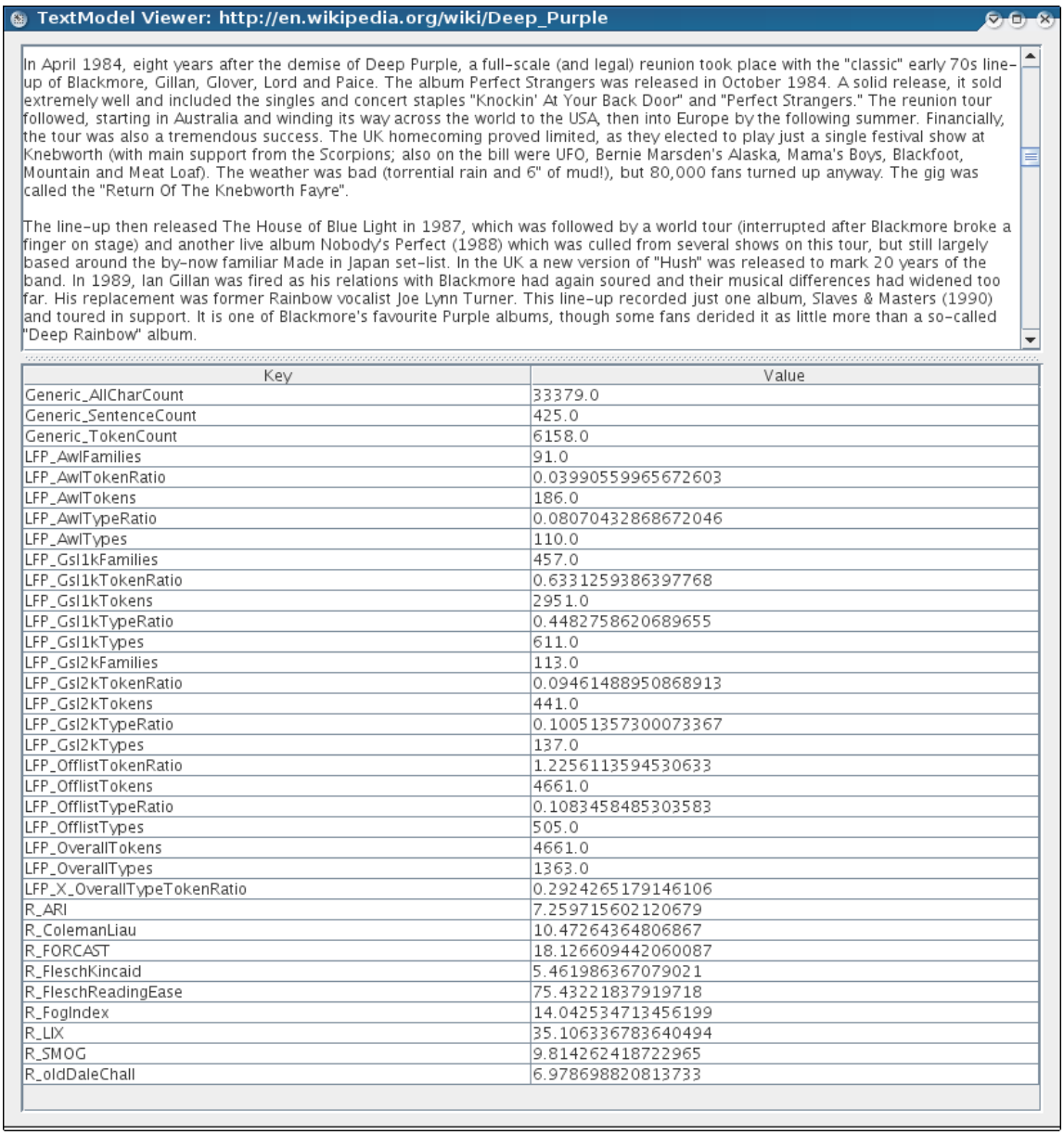

Figure A.1: Screen shot of the Text Model Viewer used for debugging.

A.2 List of Abbreviations

AE Analysis Engine

API Application Programming Interface

ARI Automated Readability Index

AWL (The New) Academic Word List

BBC British Broadcasting Corporation

CALL Computer-assisted Language Learning

CAS Common Analysis Structure

CEF Common European Framework (of Reference for Languages)

DNS Domain Name System

ESL English as a second language

FAQ frequently asked question

FLA first language acquisition

GNU GNU's not Unix

GSL General Service List

HTML Hypertext Markup Language

IALS International Adult Literacy Survey

IBM International Business Machines Corporation

ICALL Intelligent Computer-assisted Language Learning

IR Information Retrieval

LFP Lexical Frequency Profile

LIX Läsbarhetsindex

NLP Natural Language Processing

PDF Portable Document Format

POS Part of Speech

SLA second language acquisition

SMOG Simple Measure of Gobbledygook

SVM Support Vector Machine

TAGARELA Teaching Aid for Grammatical Awareness, Recognition and Enhancement of Linguistic Abilities

TC text categorization

UIMA Unstructured Information Management Architecture

URI Uniform Resource Identifier

URL Uniform Resource Locator

UWL University Word List

VP verb phrase

WERTi Working with English Real-Texts: An Intelligent Workbook for English

WWW World Wide Web

XML Extensible Markup Language