Readability index

by

Thomas Jakobsen and Thomas Skardal

Supervisors: Morten Goodwin Olsen and Annika Nietzio

Project report for IKT407 in Autumn 2007

based on report template version 3.0 (2006)

Agder University
Faculty of Engineering and Science

Grimstad, 30 November 2007

Status: Final

Keywords: NLTK, Readability index, Python, Natural Language Processing

Abstract:
This report describes the research, experiments and results achieved when studying the field of determining the readability of a text (readability indexes). The main focuses is language classification, language research and determining the readability of Norwegian and English texts. The report is written to give the reader a good introduction to our research and results and/or as a resource to prepare for further development or research. The goal for this project is to make a plugin for the Natural Language Toolkit and we aim to contribute our code to this project. Our main field of interest is to determine which readability tests that are suitable for the Norwegian language. This has, to the best of our knowledge, never been done before and our work could therefore be a useful starting point for further research on this field. We have presented some proposals and arguments for which readability tests to emphasize for Norwegian language.

Version Control

Version1	Status2	Date3	Change4	Author5
0.1	DRAFT	2007-11-24	Created document and imported text from Google documents.	Thomas J, Thomas S
0.2	DRAFT	2007-11-26	Reviewed chapters: Background and solution	Thomas J, Thomas S
0.3	DRAFT	2007-11-27	Added introduction and reviewed the testing and validation chapter.	Thomas J, Thomas S
0.4	REVIEW	2007-11-28	Review by supervisors	Supervisors
0.5	DRAFT	2007-11-29	Made changes according to the feedback and suggestions from the supervisors.	Thomas J, Thomas S
0.6	DRAFT	2007-11-30	Added references, abstract discussion and conclusion	Thomas J, Thomas S
0.7	REVIEW	2007-11-30	Final review and error correction	Thomas J, Thomas S
1.0	FINAL	2007-11-30	Work on paper completed.	Thomas J, Thomas S

1 **Version** indicates the version number starting at 0.1 for the first draft and 1.0 for the first review version.
2 **Status** is DRAFT, REVIEW or FINAL
3 **Date** is given in ISO format: yyyy-mm-dd
4 **Change** describes the changes carried out since the previous version
5 **Author** is the one who did the change

Table of Contents

Table list

Figure list

1 Introduction

This report is a result of a project in the course IKT407 Web-mining and data analysis at University of Agder Autumn 2007.

Natural language processing [6] is the study of automated generation and understanding of natural human languages. This is becoming more and more interesting tasks to solve, as computer technology is integrated into almost every industry nowadays. We are going to study one specific field within natural language processing; readability. This involves the topic of determining the readability of a text. This indicates how difficult it is to read or understand a text. It also concerns language researching and analysis. There are a lot of existing readability tests and tools for English language. However, to the best of our knowledge such tools do not exist for Norwegian language. In addition to implementing the tests for English into our plugin, we have done some research on the Norwegian language and which tests that are applicable for this. The main field of interest is this research of which tests that can (or cannot) be used to measure the readability of Norwegian texts.

The task description is as follows:
"Develop a readability index as a plugin for the Natural Language toolkit. Investigate features and difference of the readability indices."
Develop a readability plugin for English text.
Extend the plugin to cover more languages (Norwegian).

To solve this problem we have done a lot of research on natural language processing and the two languages we have focus on. The plugin itself is programmed using the Python language.

1.1 Acknowledgements

We want to thank our supervisors Morten Goodwin Olsen and Annika Nietzio for excellent support and feedback.

1.2 Report outline

This report has four major important parts: Background, Solution, Testing and validation, and Discussion.

The *background chapter* introduces existing technologies and some technical issues that are relevant for this project. The content of the solution chapter includes implementation, requirements and design specification. *Testing and validation* is originally a part of the solution chapter, but since this is such an important section of this report it is mentioned as one of the major parts of this paper. In this section we will present our experiments and results. In the last part, discussion, we will discuss and interpret our results.

2 Problem description

In this project we are going to develop calculations of readability indexes as a plugin for the Natural Language Toolkit (NLTK). A readability index is a numeric value that indicates how difficult (or easy) it is to read and understand a text. There are several different tests for determining readability, and they have different fields of use. Most of them are intended for use with English text, but some of them are language neutral. In addition to develop the plugin to be able to calculate different readability indexes, we also have to do much research and testing to determine which tests that are useable with Norwegian language. We also need to explore the features of NLTK as it most certainly already provides functionality that is usable for us.

In order to create this plugin we need to be able to do the following:

- Classify the language of the text
- Analyze the text (count words, sentences syllables etc.), using rules for the given language.
- Calculate readabilities based on text analysis

Language classification has been done before, and there are many topics concerning document classification in general, which we will make use of.

What we find very interesting in this project is the opportunity to contribute to an existing project. To develop something that may be used by others is a huge motivation-factor for us. We are also looking forward to getting more into Python development.

3 Background

NLTK - Natural Language Toolkit

NLTK "is a suite of open source Python modules, data and documentation for research and development in natural language processing" [7]. It already contains much functionality that we need to solve our task; especially the different tokenizers will be useful.

3.1 Readability index

Readability tests

"Readability describes the ease with which a document can be read" [13]. There exist many different tests [9] to calculate readability. Readability tests are "considered to be predictions of reading ease but not the only method for determining readability" [13].

Some of the tests are language neutral, but there are some tests that are more suited for certain languages. Knowledge of the different readability tests is essential for us. We need to determine which of the tests that are useful for Norwegian and English.

Some of the most common readability tests (with short description):

Readability test:	Intended for (language(s)):	Short description and formula:
Automated Readability Index (ARI) [18]:	English	Designed to gauge the understandability of a text. The output is an approximate representation of the U.S grade level needed to comprehend a text. $ARI = 4.71 * \left(\frac{characters}{words}\right) + 0.5 * \left(\frac{words}{sentence}\right) - 21.43$
Flesch Reading Ease[20]:	English	Designed to indicate how difficult a reading passage is to understand. Higher scores indicate material that is easier to read; lower numbers mark harder-to-read passages. $FRE = 206.835 - 1.015 * \left(\frac{total\ words}{total\ sentences}\right) - 84.6 * (\frac{total\ syllables}{total\ words})$
Flesch-Kincaid Grade Level[20]:	English	Designed to indicate how difficult a reading passage is to understand. The result is a number that corresponds with a U.S grade level. $FKGL = 0.39 * \left(\frac{total\ words}{total\ sentences}\right) + 11.8 \left(\frac{total\ syllables}{total\ words}\right) - 15.59$

Coleman-Liau Index[19]:	English	Designed to gauge the understandability of a text. The output is the approximate U.S. grade level thought necessary to comprehend the text. $CLI = \left(5.89 * \left(\frac{characters}{words}\right)\right) - \left(30 * \left(\frac{sentences}{words}\right)\right) - 15.8$
Gunning Fog Index[21]:	English	Designed to measure the readability of a sample of English writing. The resulting index is an indication of the number of years of formal education (U.S grade) that a person requires in order to easily understand the text on the first reading. $GFI = 0.4 * (\left(\frac{words}{sentence}\right) + 100 * \left(\frac{complex\ words}{words}\right))$
SMOG index[22]:	English	Designed to gauge the understandability of a text. The output estimates the number of years of U.S. education needed to fully comprehend the text. $SMOG = \sqrt{total\ complex\ words * (\frac{30}{total\ sentences})} + 3$
Linsear write[23]:	English	A readability metric for English text, developed for the Air Force to help them calculate the readability of their technical manuals. Formula from Wikipedia: • Find a 100-word sample from your writing. • Calculate the easy words (defined as two syllables or less) and place a number "1" over each word, even including a, an, the, and other simple words. • Calculate the hard words (defined as three syllables or more) and place a number "3" over each word as pronounced by the dictionary. • Multiply the number of easy words times "1." • Multiply the number of hard words times "3." • Add the two previous numbers together. • Divide that total by the number of sentences.
Rate Index (RIX)[17]:	Western European Languages	This is useful because it can be used on documents of any Western European language [3]. The output is a score between 0 (very easy) and 55+ (very difficult). $RIX = (\frac{long\ words}{sentences})$ (long words = words where number of characters > 6)

Lesbarhets Index (LIX)[16]	Western European Languages	This is useful because it can be used on documents of any Western European language [2][3]. The output is a index who indicates a grade level. An index below 0.1 is grade 1 while 7.2 and above is college grade. $LIX = \left(\frac{total\ words}{total\ sentences}\right) + (\frac{long\ words}{total\ words} * 100)$ (long words = words where number of characters > 6)

Table 1: Description of different readabilty tests

Known suitable tests for Norwegian language
LIX (Lesbarhets Index - which means Readability Index in Norwegian) is "useful because it can be used on documents of any Western European language" [16].
RIX (Rate Index) - is "useful because it can be used on documents of any Western European language" [17].

3.2 Analyzing language

Tokenizers
"Tokenizers divide strings into lists of substrings. For example, tokenizers can be used to find the list of sentences or words in a string" [8]. This is useful because the different readability indexes use "characteristics such as statistical average word length and sentence length" [24] to calculate the readability.

Norwegian language and readability tests
Most of the common readability tests are made for English language. Determining which tests that are useful for Norwegian language has never been done before as far as we know.

Analyzing the Norwegian language
To be able to calculate readability for Norwegian texts, we have to analyze the Norwegian language. The only component that needs a different calculation, compared to English, is syllables. Since a complex word only relies on the syllable count of the word, we don't need to change the calculation for these. However complex words need to be considered carefully when it comes to interpreting the readability results. This is because of the special form of compound words in the Norwegian language. After doing some research on the Norwegian language and comparing it to English we found some key areas that need some extra focus when it comes to calculating syllables and complex words:

- **Compound words:** Long compound words are typical for the Nordic languages. A compound word means two or more common words that are put together to form a new word. Although the English language also has compound words, there is a major difference. In the Norwegian language, the compound words are put together to one single (often very long) word.
- Norwegian rules for detecting **syllables** (see below).

- **Special characters:** The characters *æ ø å* are Norwegian special characters and need to be taken care of. Also the consonants *c, q, w, x* and *z* appear only in words of foreign origin.

Determining Norwegian syllables
Calculating syllables for Norwegian is naturally very different from calculating them for English. One example is silent e in English. These e's are not counted as a syllable, but should be counted in most Norwegian cases. For Norwegian the general rule is that each syllable contains one vowel each.

For this general rule, we have three major restrictions [26], [27]:

- The vowel must be before one consonant
  - *so-ve, ta-le*
- Or the vowel must be between two consonants
  - *jen-te, snak-ke*
- Or the vowel must be between two vowels that do not form a diphthong.
  - Examples of two vowels that do not form a diphthong: *oa, ee, io, eo, ia.*
  - The seven Norwegian diphthongs are *ai, au, ei, oy, oi, ui* and *øy.*
  - *ra-di-o, pi-a-no*
- Thus, if we detect a diphthong, we have to subtract one syllable from the count to avoid duplicate syllable counting.

These key areas will be further elaborated with examples in the *Validation and Testing* chapter.

3.3 Language classification

The Naïve Bayes classifier [28] has previously been used in several fields such as spam-classification [30]. It is a "simple probabilistic classifier based on applying Bayes Theorem with strong (naive) independence assumptions" [28]. This has been used in previous projects [4][5] and the results were good. Although it was proven that when the differences are small (topic detection), the training process needs more data than when the differences are bigger (language detection).

Bayes Theorem:

$$P(h|o) = \frac{P(o|h)P(h)}{P(o)}$$

Bayes Theorem [29] makes it possible to calculate the probability of a hypothesis (h) to be true, when an observation (o) is given.

- P(h) = The probability for a hypothesis to be true
- P(o) = The probability for a observation to be true
- P(h|o) = The probability for a hypothesis to be true given that a observation is true
- P(o|h) = The probability for a observation to be true given that a hypothesis is true

Stop words
For the language classification task, we need to consider the use of stop words. Stop words are "words which are filtered out prior to, or after, processing of natural language data"[11]. There exist many lists [11][12] of such stop words. NLTK's corpus contains a collection of stop words for 11 different languages, including Norwegian and English. The reason that we often want to filter out this words is that they occur very frequently and may make the classification harder. In our case the stop words need to be discussed later as they probably don't have a negative effect on the language classification.

3.4 Existing software

There are some already existing solutions [15][31] to our problem, but there doesn't seem to be any attempts on developing this for NLTK. Most existing solutions seems to be either commercial (software) or only available as online tests [31]. But there also exist open source software like for instance *Style* for Linux. Style is a rather old application, and was early claimed to be "in wide use at Bell Laboratories both professional and nonprofessional writers. (…) they find the programs useful and feel that their writing has improved" [1](1981). It is also possible to find both the Flesch Reading Ease and Flesch-Kincaid Grade Level in office software like Google Documents and Microsoft Word [32](not only the latest version, but at least since Office 2000. The tests have improved in newer versions). The online tests of course have the advantage that they are free and easy accessible, but on the other hand they require a connection to the internet and cannot be easily used by developers. The commercial software we have found seems to be aimed towards the end-user, and not for use in development and language processing.

Since there is no such thing available for NLTK we think this is a good addition since this is free and open source software for doing language processing.

4 Solution

4.1 Requirements

Functional requirements

- Our solution should provide support for calculating the readability of a text using several known techniques. Of the tests described in *Table 1*, we have decided to focus on 8 different tests. This should provide a broad specter of different tests and output. These tests are:
    - Automated Readability Index
    - Flesch Reading Ease
    - Flesch-Kincaid Grade Level
    - Coleman-Liau
    - Gunning fog
    - SMOG index
    - RIX (Rate Index)
    - LIX (Lesbarhets Index)

- Our plugin will have support for English and Norwegian language. We should be able to determine the language of a text (*English*, *Norwegian* or *unknown* if some other language is inputted).

Linguistical requirements

In order to be able to calculate the readabilities using the above tests we have some linguistical requirements.

- Language analysis
    - English
    - Norwegian
    - For both languages we need to determine several factors
        - The number of syllables
        - The number of sentences
        - The number of words
        - The number of characters
        - Complex words

Non-functional requirements

NLTK is written in Python, and therefore we are required to use this language.

Natural Language Toolkit requirements (excerpt from NLTK Developers guide [33]):

- Maintain high-quality coding standards; follow PEP-8 Style [34] for code.

- Use Epydoc [35] docstring.
- Provide a demo() function in each module to illustrate its functionality.
- Demo code should include import statements so that it can run standalone.
- Demo functions should have no arguments, and should be run when the module is called on the command line.

4.2 Design Specification

As stated in the requirements chapter we are required to program in Python. As IDE we will use Eclipse with the PyDev plug-in which enables easy Python development. For source control we use Subversion.
We will make this readability plug-in as component based as possible. Our main readability engine will then connect and engage the desired components based on user-requests.

4.2.1 Planned testing and experiments

There are a lot of cases that need to be taken into consideration, and to be able provide reliable results, as many as possible of these should be covered. In order to make improvements to our plugin, we need to find its weaknesses. This can only be done by performing tests, and interpret the results; hence we need a good plan for testing and validation.

Text analyzer
The accuracy and reliability of the readability tests requires precise analysis of the text input. Therefore we will test our own analysis module against results from various online analysis tools. We simply want to run a text through our analysis module and through the online tools, and compare the results.

The factors we need to test are:

- The number of syllables
- The number of sentences
- The number of words
- The number of characters
- Complex words

The only factor that differs from Norwegian to English when it comes to calculations is syllables counting. Therefore we need to make sure to run this through some Norwegian (or Danish) analyzers. For the others it does not matter if we run Norwegian text through an English analyzer (given that it can handle special characters such as æøå).

Since determining syllables in Norwegian is totally different from English we will do some further testing of this module. We want to run a text through our text analyzer and add every word to a text-file with the corresponding number of syllables that are counted by our algorithm. This file (500-1000 words) we will check manually. All the correctly calculated words will be deleted from this file. The ones that our implementation has counted incorrectly we will keep in the file, with the corrected number of syllables. This file will then be used as a "special cases" file. The next time we run the text, the algorithm will check if a word exists in this file. If it does, we extract the correct number of syllables. This will increase the accuracy of the syllable count.

Readability tests
When testing the readability tests we use the same strategy as for the text analyzer. We want to use some sample texts and run them through all our readability tests. Then we use the same texts on the online readability tests and compare the results. To ensure consistency in the test results we will use as many online sources as possible.

Language classifier

- **Training**
  During the training of the classifier we will use our own web-crawler to download web-pages that we can extract text from. By doing this we can train the language classifier to be close to 100% accurate.
- **Testing the accuracy**
  Since we only consider two languages in this project, Norwegian and English, we will test the language classifier manually. We will just try with a certain amount (10-20 texts) of Norwegian and English texts and ratify the accuracy of the classifier.

Testing the whole flow of events
In the end we will test the whole process to ensure that the different components work together as intended. This will only be internal testing of the software and will help us making the components more robust. This phase includes debugging and testing of special cases.

Planned system flow

The class readability takes either a string or URL as input for calculations of readability index. These steps are processed as shown in *Figure 1*:

- Check if it is an URL. If it is: use the class *URLextracter* to receive the content
- Use the class *NaiveBayes* to classify the language of the text
- Analyze the text by using the class *TextAnalyzer*. This class needs to be told which language the text is in, in order to count syllables correctly.
- With all the statistics from the text and a known language, the suggested tests are now provided, and the ones remaining are listed with a "warning".

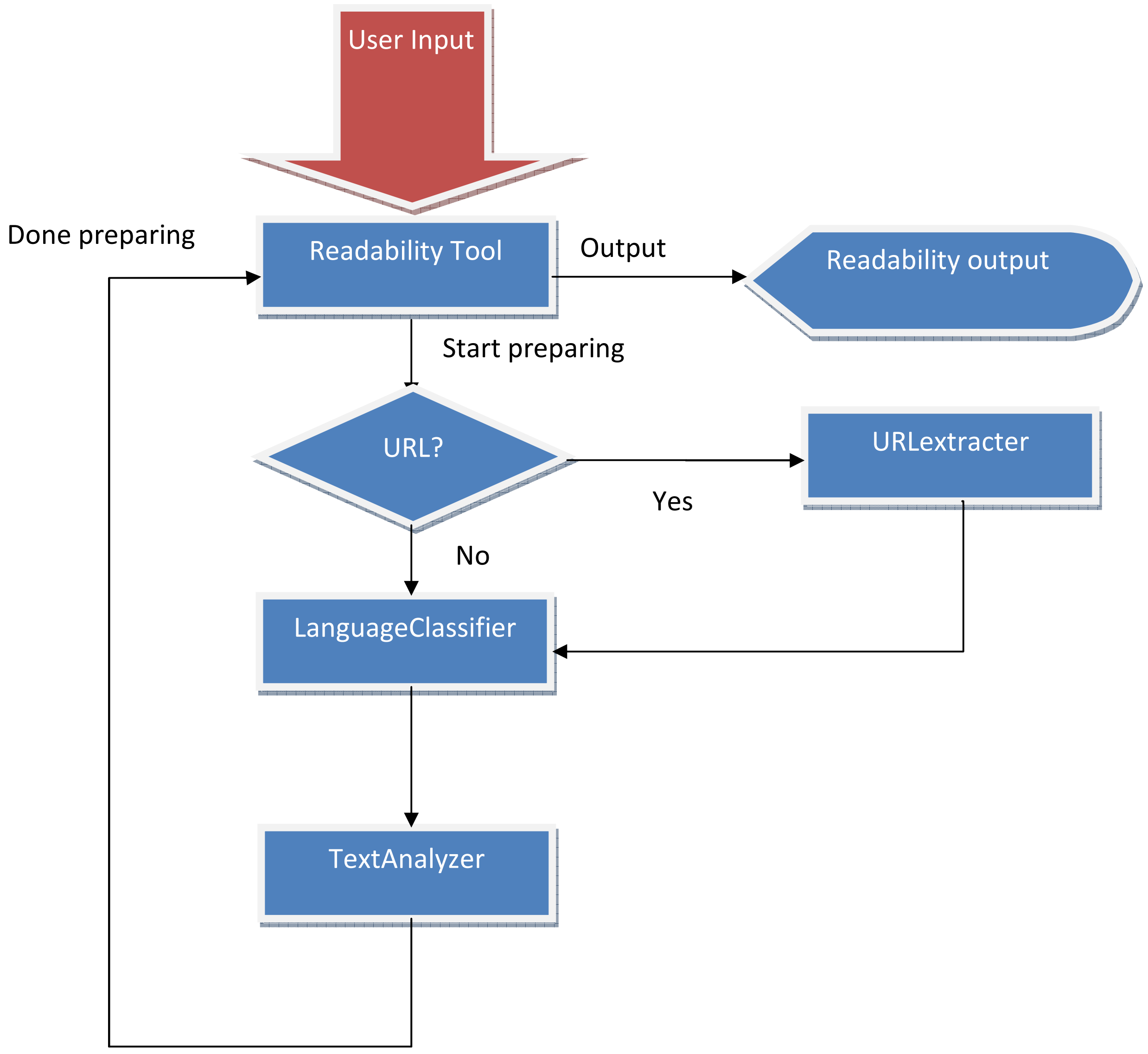


Figure 1: System flow chart

4.3 Implementation

As presented *Figure 1* our program-flow consists of three major parts: Language classification, text analyzing and readability tests. These components together define the project scope of our implementation.

4.3.1 Language classification

The language classifier is the first important component. Our implementation of this is a Naïve Bayes classifier trained for English and Norwegian. It takes text or an URL as input and classifies the language of the content. The classifier has been trained using pure text, which is text where all special characters and numbers are removed. We also take advantage of the stop-words that are already available in NLTKs corpus. There is one set of stop-words for both Norwegian and English. These stop-words are words that we are not interested in when calculating the probabilities. The class provides the user with methods for training, the classifier, and of course classifying a text or URL. When the probabilities are calculated, we are also keeping count of how many of the words that are unknown. If the ratio between known and unknown words (without repetition) is 0.25 or less, we classify it as unknown. As shown in *Table 2* this threshold seems to be reasonable at this state, but should probably be reconsidered if the classifier is trained to cover more languages.

URL	Known	Unkown	Ratio	Classified as
harvardscience.harvard.edu	74	80	**0,93**	eng
vg.no	305	357	**0,85**	no
bbc.co.uk	86	60	**1,43**	eng
startsiden.no	46	65	**0,71**	no
news.com	364	176	**2,07**	eng
munimadrid.es	29	205	**0,14**	unknown
welt.de	134	675	**0,20**	unknown
news.pl	34	143	**0,24**	unknown
ekstrabladet.dk	241	642	**0,38**	no
gazzetta.it	98	574	**0,17**	unknown

Table 2: Finding a reasonable threshold for known/unknown classification.

URLextracter
This class inherits the built-in Python module sgmllib [36]. It is used to parse the HTML at a given URL and collect nothing but links and text. This class is used both by the language classifier, for determining language of an URL, and also by our crawler. In general, it is used whenever it is needed to extract text from an URL.

Crawler
We decided to implement a crawler to make the process of gathering test data easier. The crawler is very simple, and uses the URLextracter to get pure text from a URL and store it as a file. It uses a "random walk" strategy, that means it selects randomly one of the links found on the URL, extracting the text and continues doing this until it is terminated. The crawler contains no more functionality than this, and is only a tool we have used in the training process of the classifier.

```
# Pseudo code - random walk
def crawl_random_walk(page):
    crawl_the_page(page)                  #crawl this page
    next_page = select_random(page.links) #select a new random link
    crawl_random_walk(next_page)          #go to next page
```

Figure 2: Pseudo code - random walk

Text analyzing
Analyzing the text is the most crucial point of our project. If the text analysis fails or is inaccurate, the output of our readability tests will be less useful. Our text analyzing module consists of various methods for retrieving information about the input text such as getting word count, sentence count, syllables and complex words.

Analyzing the English language
The module [3] and metric used for syllables have been used in similar projects[4] in the previous years, and is taken from Red-Bean. We also take advantage of NLTK's RegexpTokenizer, which divides a text according to a regular expression, to collect all the words, and the PunktSentenceTokenizer to divide it into sentences.

Readability tests
After the input text or URL has been processed and analyzed the readability tests are ready to calculate the different indexes. The tests use the information from the text analyzer to calculate the different indexes and are thus deterministic and formal. The user will also be provided with suggestions for which tests to focus on, and which one to ignore according to the language that is classified. The class provides the user with separate methods for each test, as well as a "full" test.

4.4 Validation and testing

Language Classification

When training the classifier, we started out collecting texts written in Norwegian and English manually by just copy and paste text content from the web into files. With 30 text files from each used for training and 10 different used for testing, we got the following results when running the test five times:

Run	1st	2nd	3rd	4th	5th	Avg
Errors	2	0	2	3	4	2,2
Total	10	10	10	10	10	
Accuracy	80%	100%	80%	70%	60%	78%

Table 3: We can clearly see that this classifier is not very trustworthy.

To get more precise results we then used our crawler to harvest more data easily. Since the crawler has no strategy for which links to follow, we first pointed the crawler to a well known Norwegian or English page. Then we monitored the URLs that it was crawling, and just stopped it manually if it “got lost” on pages with wrong language. The reason that the crawler sometimes "get lost" is that there for instance can exist links from a Norwegian page to a page with different language. We also filtered out all texts below 800 bytes to make sure that there was a minimum length of the texts.

Using the dynamically mined data for training and testing we got the following results when running the code five times:

Run	1st	2nd	3rd	4th	5th	Avg.
Errors	1	3	4	3	0	2,2
Total	100	100	100	100	100	
Accuracy	99%	97%	96%	97%	100%	98%

Table 4: As we assumed, the results are getting more stable and accurate when the amount of training data is increased.

Even though we got a satisfying result on classifying, we tried to find out if there was any common properties among the texts that were classified wrong. Quite surprisingly, all of the texts that were classified wrong actually were wrong. There had been stored some English text among the Norwegian, and some German and French among the English texts. (There were also a couple of pages containing nothing but garbage, probably binary files stored as text.) When removing these erroneous files, and moving the usable files to correct locations, we now have even better accuracy as shown in *Table 6.* Even though we get 100% accuracy here, there will most probably be cases where this is not true, so we cannot claim to have proven 100% accuracy.

Run	1st	2nd	3rd	4th	5th	Avg.
Errors	0	0	0	0	0	0
Total	100	100	100	100	100	
Accuracy	100%	100%	100%	100%	100%	100%

Table 5: Erroneous files deleted or replaced

As mentioned in chapter implementation, we classify text that contains less than 25% known words as unknown. When trying to classify a Spanish, a German and a Danish page, the Danish page is classified as Norwegian, which was expected since there are many common Norwegian and Danish words. Both the Spanish and German pages are classified as "unknown".

4.4.1 Analyzing the text and calculating readabilities

When testing the text analyzer and the calculation of readabilities, we have used the readability calculator at EditCentral [31] as a guideline. It provides information about such as words, sentences syllables, and also most of the tests that we have implemented. When running the following tests we have used the (English) demo text found at EditCentral.

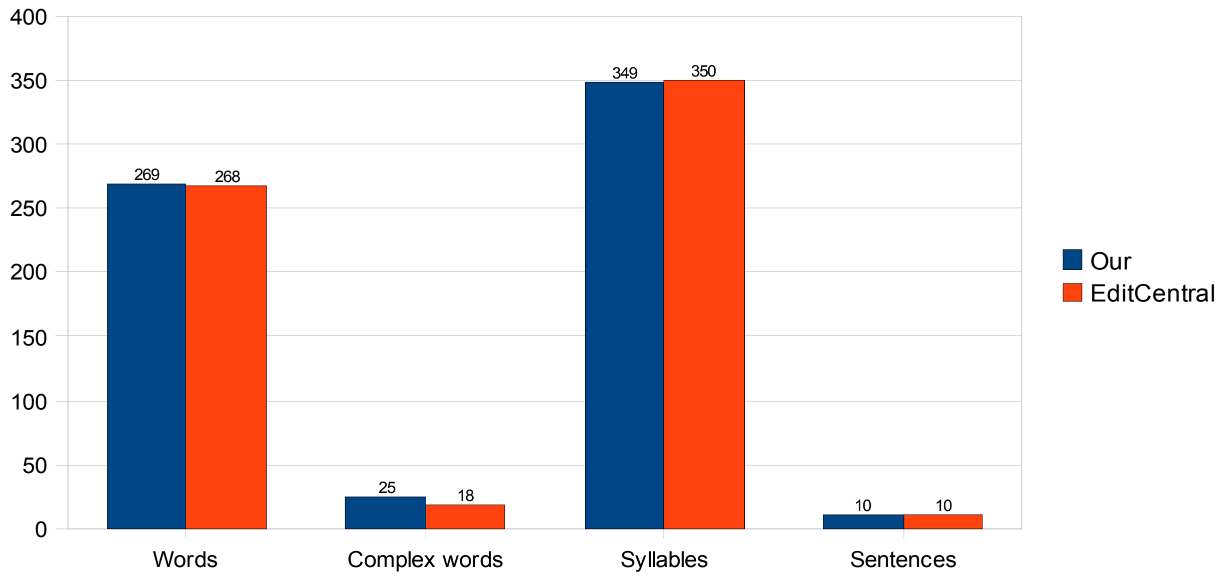


Figure 3: Comparison of our and Edit Central's text analysis

As you can see the aberration is mostly in the complex words. Next, we have compared the results of the readability tests.

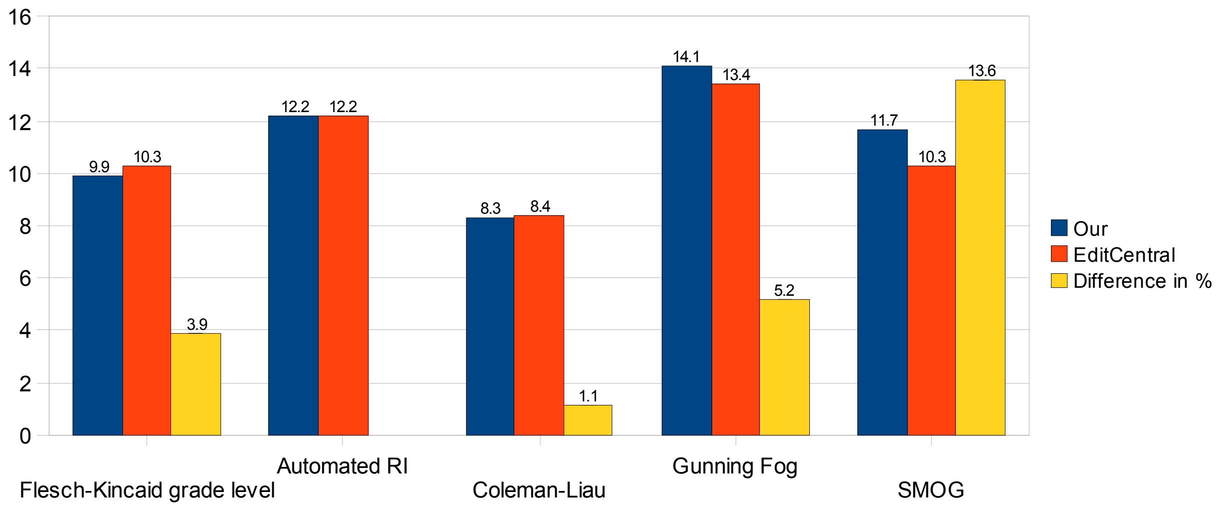


Figure 4: Comparison of readability grade level results

The aberrations in *Figure 4* and *Figure 5* are not that bad, but logically the tests that uses complex words have bigger aberration than those who just uses number of sentences and words.

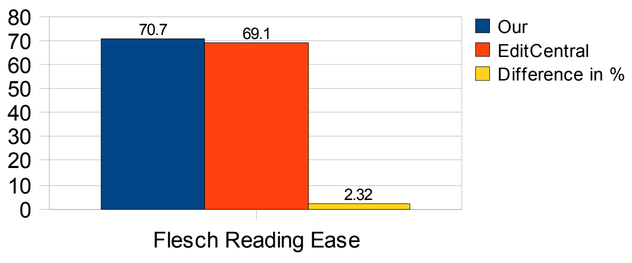


Figure 5: Comparison of Flesch Reading Ease

4.4.1.1 Analyzing the Norwegian language

As mentioned in the background chapter there is only one calculation we need to change to analyze Norwegian (instead of English); syllables. Therefore we examine these key areas:

Compound words
One example of a compound word is the Norwegian word *"En midtsommernattsdrøm"* which would be *"A Midsummer Night's Dream"* in English [25]. Thus the number of characters per word would be very different values for either of the languages. In this example the number of characters per word would be 10.5 for Norwegian and only 5.25 for English. The value for Norwegian is twice the amount compared to English. But we have to answer another question now; isn't the Norwegian compound word more difficult to read than the English 4-word sentence? The answer might seem obvious, as it should be much more difficult to read one long complex word than 4 shorter words. However for the English sentence we have 4 words which will increase the difficulty compared to the Norwegian word. These are things we need to consider when determining which tests that are suitable for Norwegian. There are a lot of other factors that should be analyzed and considered. This is just one short example, but due to time limitations we have only been able to do some basic research on this field.

Syllables
After a lot of testing of our Norwegian syllables implementation we have discovered one aberration to the restrictions explained in the *background chapter*.

The following point is just our experience gained by using our own implementation, our basic knowledge of Norwegian grammar and are based on our relatively little research on the Norwegian language. We are not experts on Norwegian language thus this point must be further researched before used.

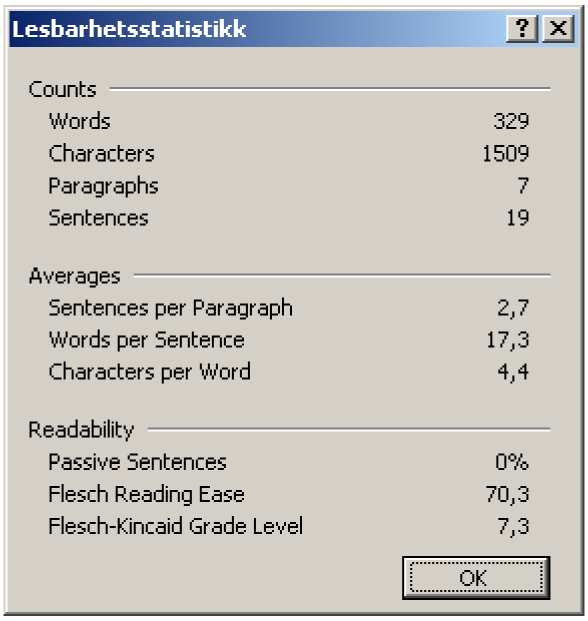
The exception
We found that the syllable counter became more accurate if we didn't count *ei* as a diphthong. When we tested the accuracy manually we found this pattern in many of the words that were counted incorrectly. When we tested again without *ei* as a diphthong, there were less faulty syllable counts. Examples: *leite, beite, leie, utarbeidet, aribeidsforhold, utvei*. All these and many more were counted correctly after removing *ei* from the diphthong list. Basically this means that we count *ei* as a syllable.

4.5 Comparing our results with Microsoft Word 2007

As Microsoft Word is a very commonly used word processor, we decided to use this as a test of our own software. We do this simply because Word has been developed for several years, and should provide reliable information. When we compare the readability calculations and text statistics of both English and a Norwegian text [38][39] we got the results as shown in *Table 7*. It is obvious that there are some aberrations. Concerning the words, this is probably caused by the regular expression we use to split text into word, as there probably still are some uncovered cases. Also we are removing everything that is neither a number nor a letter. This issue is then also the reason for the difference in the LIX (Lesbarhetsindeks) score. In the English results, syllable counting is probably also affecting the scores.

English		**Word 2007**	**Us**
Words		329	335
Characters		1509	1482
Sentences		19	19
Flesch Reading Ease		70,3	78,2
Flesch-Kincaid Grade Level		7,3	6,6
Norwegian		**Word 2007**	**Us**
Characters		1676	1658
Words		386	394
Sentences		19	19
LIX		37,7	38,8

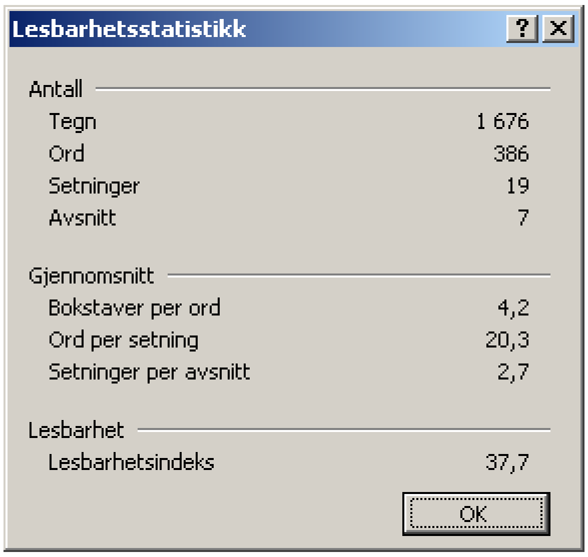
Table 6: Comparison with MS Word 2007



Lesbarhetsstatistikk

Counts	
Words	329
Characters	1509
Paragraphs	7
Sentences	19
Averages	
Sentences per Paragraph	2,7
Words per Sentence	17,3
Characters per Word	4,4
Readability	
Passive Sentences	0%
Flesch Reading Ease	70,3
Flesch-Kincaid Grade Level	7,3

Figure 7: English results in Word 2007



Lesbarhetsstatistikk

Antall	
Tegn	1 676
Ord	386
Setninger	19
Avsnitt	7
Gjennomsnitt	
Bokstaver per ord	4,2
Ord per setning	20,3
Setninger per avsnitt	2,7
Lesbarhet	
Lesbarhetsindeks	37,7

Figure 6: Norwegian results in Word 2007

The Norwegian sample was also compared with results from the Danish LIX test [40]. The Danish test resulted in a LIX score of 39.8, which is higher than both the result from Word, and our own result. We find these results satisfying, as we are in between of these two other results. This can only mean that our implementation is working well.

4.6 Results from text written in both Norwegian and English

We have used both the Universal Declaration of Human Rights [37], as well as two samples from the Pragmatic Programmer's Guide [38][39] (ch. 1, ch. 2). All of these three texts are written both in English and Norwegian, and we are interested in how much the results differ between the two languages. To calculate the following results we have used our own implementation.

	Ruby – Preface			Ruby – What kind of lang...			Universal Declaration of Human Right		
	English	Norwegian	Difference	English	Norwegian	Difference	English	Norwegian	Difference
Number of characters	1482	1658	10,6%	1742	1794	2,9%	7297	7166	-1,8%
Words	335	394	15,0 %	352	366	3,8 %	1532	1446	-5,9 %
Sentences	19	19	0,0%	22	22	0,0%	92	97	5,2%
Syllables	441	620	28,9 %	561	650	13,7 %	2304	2515	8,4 %
Complex Words	36	46	21,7%	61	70	12,9%	306	328	6,7%
ARI	8,2	8,8	6,8%	9,9	10	1,0%	9,3	9,4	1,1%
Flesch Reading Ease	78,2	53,4	-46,4 %	55,8	40,3	-38,5 %	63,4	45,5	-39,3 %
Flesch-Kincaid Grade	6,6	10,8	38,9%	9,5	11,6	18,1%	8,4	10,4	19,2%
Gunning Fog	11,1	12,7	12,6 %	13,3	14,1	5,7 %	14,4	14,7	2,0 %
SMOG Index	10,5	11,5	8,7%	12,1	12,8	5,5%	13	13,1	0,8%
Coleman-Liau	8,6	7,5	-14,7 %	11,5	11,3	-1,8 %	10,5	11,4	7,9 %
LIX	34,3	38,8	11,6%	43,8	41,8	-4,8%	44,7	42,1	-6,2%
RIX	2,9	3,7	21,6 %	4,5	4,2	-7,1 %	4,7	4,1	-14,6 %

Table 7: Comparison of results from texts written in both English and Norwegian

The difference is calculated using this formula: $\frac{Norwegian-English}{Norwegian} \times 100$

We can clearly see that the amount of complex words is higher in the Norwegian versions. The reason for this is of course the use of compound words in Norwegian language, which increases the frequency of words with many syllables. There also seems to be more syllables in general in Norwegian language.

4.7 Determining suitable readability tests for Norwegian texts

We have done some research and testing, and found some arguments and opinions of which of the tests that are more suitable than others for calculating readability of Norwegian texts.

Readability tests for Norwegian language

Since determining which of the English readability tests that can be recommended for Norwegian language never has been done before, it's important to discuss our thoughts and conclusions. For the first, there are many factors to have in mind when comparing Norwegian and English. As described earlier in the chapters *background* and *validation and testing* the Norwegian language has a more complex use of compound words. This affects the results in the tests. Also when comparing the text analysis of Norwegian and English texts, we can see a tendency that all variables gets higher values when analyzing a Norwegian text with the corresponding English version and vice versa. Does this mean that the Norwegian language is harder to read than English? This is very hard to prove, and therefore we won't like to conclude with it as we have done way to little testing to do that. Also there are numerous other factors that we haven't taken into consideration. However, we will conclude with that this is a consistent tendency in all our test results. Therefore we need to do some adjustments when reading the output of the "Norwegian" readability tests. Most of the tests shows that the Norwegian texts needs a higher level of education to understand for all the texts we have tested. Our testing suggests that because of the special rules of Norwegian compound words, the number of complex words will often be much larger for Norwegian tests compared to English. This is also the case for syllables. Norwegian texts have significantly more syllables than the corresponding English translation (or the other way around).

Conclusion
We will at this point not recommend the tests that use syllables and complex words as a part of the calculation due to the discussion above.
These tests are:

- Gunning Fog Index (complex words)
- Flesch Reading Ease (syllables)
- FleschKincaid Grade Level (syllables)
- SMOG Index (complex words)

Our conclusion is that these factors needs more research to determine how much they should be emphasized.

The tests we can recommend based on our research, experiments and analysis are:

- Automated Readability Index
- Coleman Liau

The reason we can recommend these tests is that our test results show that these tests have the lowest aberrations. The other reason is that they use the same factors in the calculations as LIX and RIX. LIX and RIX are suited for Western-European languages as described in *Table 1* and have previously been used for determining Swedish [16][17] and Danish [40] readability indexes.

5 Discussion

Text analyzer results
The basis of all of our results is the text analysis. Hence we can say that this is the most important part of this project. Without a correct analysis, all of our later calculations would be inaccurate and less useful. As stated in the testing chapter we used several sources to compare our results with. Since syllables counting is the only factor that is calculated differently from English to Norwegian, this is the most interesting part to discuss. The other factors like word count and sentence count are pretty straight forward. The syllables however, required a lot of research. Making a syllables counter for the Norwegian language has to our best knowledge never been done before. Therefore our implementation of this is important to discuss. What is the background for our results? What is the algorithm based on? How did we proceed?

Norwegian syllables counter
For counting English syllables, we used an existing Python module [41]. When starting the work on the Norwegian syllables counter, this module was our starting point. After a lot of research on the Norwegian language (focused towards syllables) we found some basic rules described in the background chapter. Our results are based on these rules with some modifications (also explained earlier).
After testing the counter we got some edifying results. It was pretty accurate (80-90%). After doing some modifications we got an accuracy of more than 90%. The reason we can rely on this accuracy is that we tested it manually. We ran the Norwegian version of the *Universal Declaration of Human Rights* (1728 words) through the syllables counter and added all the words to a file together with the corresponding number of syllables counted for each word. Then we went through all these 1728 words manually.

Language classification results
As the differences between Norwegian and English language is quite big, we consider the results from testing the language classifier for satisfying. We were able to gather enough data to get a stable and accurate classifier. Concerning the removal of stop words, this was tested with both cases. There were no noticeable differences, as the classifier has very high accuracy now. But, when the classifier is extended to cover more languages, they should perhaps not be removed. The reason for this is the frequent occurrence of the stop words, and hence they could be a good hint for language identification. When trying to classify languages that have many equal words such as Norwegian and Danish this may be extra helpful if the stop words are different.

Readability test results
The more we improved our text analyzer, the more accurate output we got from the readability tests.
For English, we have showed by our tests that our implementation is competitive compared with online readability-tests. Still we have many improvements to do to, especially linguistical challenges. We have not found a solid definition for determining complex words. This result in some cases some slight aberrations compared to the online tests.

6 Conclusion

In this project we have examined and implemented a solution for readability tests for English and Norwegian language. As a part of this implementation we have made a text analyzer that handles both languages. We have also made a language classifier to be able to determine the language of a text. Putting these components together we have compared and tested our results against existing software and solutions. As the main field of interest we have examined which tests that can be used to measure the readability of Norwegian texts.

As stated in the problem description, we were supposed to create a readability index plug-in for the Natural Language Tool Kit. To implement the readability-tests themselves was quickly done and very easy. The difficult part was to get good, reliable and useful output from the tests. We had to analyze the input text to extract the correct values for the variables needed by the readability tests. The test results show that we have managed to do this adequately, hence we have fulfilled our first requirement; analyzing a text. When having a good text analysis it was no problem to get some good output from the readability tests. Then in the last part of the task we examined how the English readability tests responded to Norwegian texts. We then did analysis on this to be able to argue for our conclusion.

7 Appendices

Appendix 1 Glossary & Abbreviations

ARI	**Automated Readability Index**
Complex Words	**Word with 3 or more syllables**
HTML	**Hypertext Markup Language**
IDE	**Integrated Development Environment**
LIX	**Lesbarhets Index**
NLTK	**Natural Language ToolKit**
Plugin	**An application extension**
RIX	**Rate Index**
URL	**Uniform Resource Locator**

Appendix 2 References

[1] Lorinda Cherry, “Computer Aids for Writers”, 1981
[2] Anderson, Jonathan, “Analysing the Readability of English and Non-English Texts in the Classroom with Lix”, 1981
[3] Anderson, Jonathan, “Lix and Rix: Variations of a Little-Known Readability Index”, 1983
[4] Zhang Xiaorui, e-Content accessibility for dyslexic people, 2006
[5] Erik Kristoffersen, Erling Kristiansen and Marius Andre Sæthre, Document classification, 2003
[6] Natural Language Processing, http://en.wikipedia.org/w/index.php?title=Natural_language_processing&oldid=173665014
[7] Natural Language ToolKit Homepage, http://nltk.sourceforge.net/index.php/Main_Page
[8] NLTK Tokenizers, http://nltk.org/doc/guides/tokenize.html
[9] Readability tests, http://en.wikipedia.org/w/index.php?title=Readability_test&oldid=171400164
[10]Stop words, http://en.wikipedia.org/w/index.php?title=Stop_words&oldid=174281061
[11]List of stop words 1, http://www.ranks.nl/stopwords/

[12]List of stop words 2,
http://www.library.ubc.ca/ovidweb/webhelp/ref/stops.htm
[13]Everything you ever wanted to know about readability tests…,
http://www.gopdg.com/plainlanguage/readability.html
[14]List of different readability tests,
http://en.wikipedia.org/w/index.php?title=Category:Readability_tests&oldid=76546758
[15]Oleander Solutions, Readability Studio,
http://www.oleandersolutions.com/readabilitystudio.html
[16]Lesbarhets index,
http://www.oleandersolutions.com/lix.html
[17]Rate Index,
http://www.oleandersolutions.com/rix.html
[18]Automated Readability Index,
http://en.wikipedia.org/w/index.php?title=Automated_Readability_Index&oldid=166426317
[19]Coleman-Liau index,
http://en.wikipedia.org/w/index.php?title=Coleman-Liau_Index&oldid=173389238
[20]Flesch-Kincaid Readability tests,
http://en.wikipedia.org/w/index.php?title=Flesch-Kincaid_Readability_Test&oldid=173388903
[21]Gunning fog index,
http://en.wikipedia.org/w/index.php?title=Gunning_fog_index&oldid=171180096
[22]SMOG index,
http://en.wikipedia.org/w/index.php?title=SMOG_Index&oldid=160567801
[23]Linsear write,
http://en.wikipedia.org/w/index.php?title=Linsear_Write&oldid=106624968
[24]Wikipedia, Readability test,
http://en.wikipedia.org/w/index.php?title=Readability_test&oldid=171400164
[25]Norwegian language, compound words,
http://en.wikipedia.org/w/index.php?title=Norwegian_language&oldid=174772750
[26]Norwegian syllables,
http://en.wikibooks.org/w/index.php?title=Norwegian/Introduction&oldid=999577
[27]More on Norwegian syllables,
http://www.caplex.no/Web/ArticleView.aspx?id=9801518
[28]Naïve Bayes Classifier,
http://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&oldid=166344024
[29]Bayes Theorem,
http://en.wikipedia.org/w/index.php?title=Bayes%27_theorem&oldid=174115731#Statement_of_Bayes.27_theorem
[30]Bayesian spam filtering,
http://en.wikipedia.org/w/index.php?title=Bayesian_spam_filtering&oldid=170966656
[31]EditCentral, Online readability tests,
http://www.editcentral.com/
[32]Readability information in Microsoft Word,
http://office.microsoft.com/en-us/help/HP101485061033.aspx
[33]NLTK Developers Guide,
http://nltk.sourceforge.net/index.php/Developers_Guide
[34]PEP-8 Style,
http://www.python.org/dev/peps/pep-0008/

[35]Epydoc, Python docstrings,
http://epydoc.sourceforge.net/
[36]Sgmllib Python module, Simple SGML parser,
http://docs.python.org/lib/module-sgmllib.html
[37]Universal Declaration of Human Rights,
http://www.unhchr.ch/udhr/lang/eng.htm
[38]English version of Ruby on Rails programming guide,
http://www.ruby.no/prosjekt/ProgrammeringIRuby/OriginalProgrammingRuby/
[39]Norwegian version of Ruby on Rails programming guide,
http://www.ruby.no/prosjekt/ProgrammeringIRuby/preface.html
[40]Danish readability website,
http://www.internetformidling.dk/teksttest/
[41]Python module for English syllable counting,
http://viewvc.red-bean.com/gnoetics/src/syllables.py?revision=1&view=markup