



## Readability index

by

*Thomas Jakobsen, Thomas Skardal*

Supervisor: Morten Goodwin Olsen

Co-supervisor: Annika Nietzio

### **Project report for Web-mining in autumn 2007**

based on report template version 3.0 (2006)

Agder University

Faculty of Engineering and Science

Grimstad, 8 November 2007

Status: Draft

**Keywords:** NLTK, Python, readability, language classification

[This work is licensed under the Creative Commons Attribution-ShareAlike License \(http://creativecommons.org/licenses/by-sa/2.5/\).](http://creativecommons.org/licenses/by-sa/2.5/)

## Version Control

<b>Version<sup>1</sup></b>	<b>Status<sup>2</sup></b>	<b>Date<sup>3</sup></b>	<b>Change<sup>4</sup></b>	<b>Author<sup>5</sup></b>
0.1	Draft	2007-10-29	Started on discussion and implementation	Thomas Jakobsen Thomas Skardal
0.2	Draft	2007-11-08	Modifications and review	Thomas Jakobsen Thomas Skardal

---

**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

<u>1 Solution.....</u>	<u>5</u>
1.1 Implementation.....	5
1.2 Validation and Testing.....	6
Language Classification.....	6
Analyzing the text and calculating readabilities.....	7
<u>2 Discussion.....</u>	<u>8</u>
<u>Appendices.....</u>	<u>9</u>
<u>Appendix 1 References.....</u>	<u>9</u>

# 1 Solution

## 1.1 Implementation

As presented under chapter 2.3 our program-flow consists of three major parts: Language classification, text analyzing and readability tests. These components together defines the project scope of implementation.

### Language classification

The language classifier is the first important component. Our implementation of this is a Naive 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, that 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[1]. 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, or loading, the classifier, and of course classifying a text or URL.

### URLextracter

This class inherits the built-in Python module sgmlib[2]. It is used to parse the HTML at a given URL and collect nothing but links and text.

### 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 continue 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.

### 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. The module[3] and metric used for syllables have been used in similar projects 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. Since the tests are generally a mathematical formula, there is nothing more happening here than just using the information from the textanalyzer and some constants to calculate the different indexes. 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.

## 1.2 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 texts from each used for training and 10 different used for testing, we got these results when running the test five times:

Run	1 <sup>st</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>	4 <sup>th</sup>	5 <sup>th</sup>
Errors	2	0	2	3	4
Total	10	10	10	10	10
Accuracy	80%	100%	80%	70%	60%

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 “intelligence”, we first pointed the crawler to a 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. 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 these results when running the code five times:

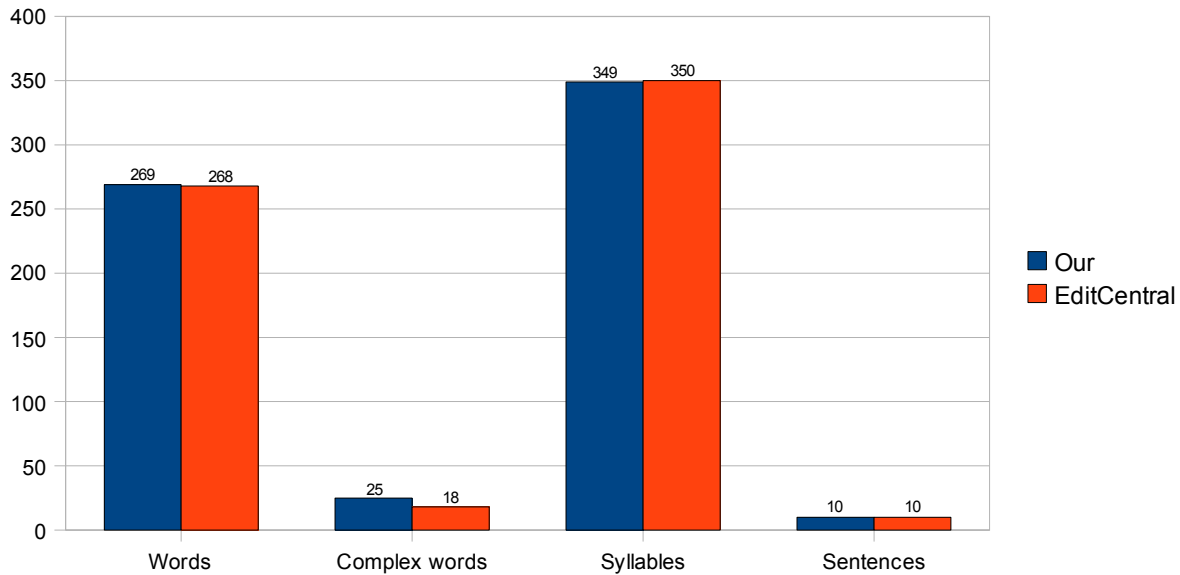
Run	1 <sup>st</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>	4 <sup>th</sup>	5 <sup>th</sup>
Errors	1	3	4	3	0
Total	100	100	100	100	100
Accuracy	99%	97%	96%	97%	100%

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

### Analyzing the text and calculating readabilities

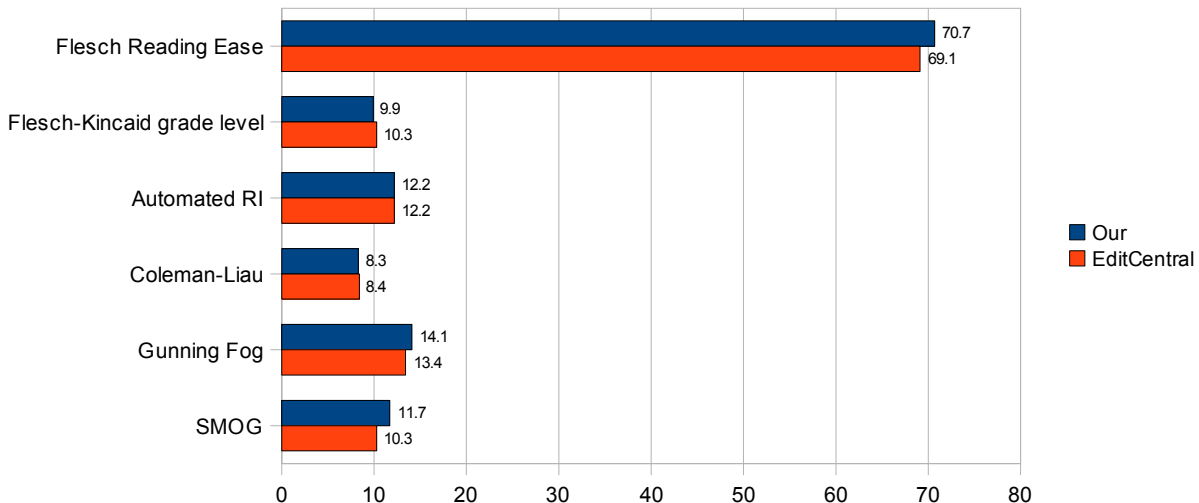
When testing the textanalyzer and the calculation of readabilities, we have used the readability calculator at EditCentral as a guideline. It provides information about such as words, sentences syllables, and also most of the tests that we have implemented.

Comparison of our and EditCentrals[4] text analyzis:



As you can see the aberration is mostly in the complex words.

Comparison of our and EditCentrals readability indexes.



The aberrations isn't that bad, but logically the tests that uses complex words have bigger aberration than those who just uses number of sentences and words.

## 2 Discussion

As stated in the problem description of this project, 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 shows that we have managed to do this adequately, hence we have fulfilled our first requirement; analyzing a text.

As stated above, the implementation of the readability-tests was easy, and with the accurate text-analysis it was no problem to get some output.

Although these two requirements are considered to be done they do not fulfil the second goal of this task, implementing support for more languages. They only consider English language and would not be reliable for other languages such as Norwegian yet. However, for English, we have proved by our tests that our implementation is competitive compared with online readability-tests. Still we have many improvements to do to, especially lingual challenges. We have not found a solid definition for determining complex words. This results in some cases some slight aberrations compared to the online tests.

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.

## Appendices

### Appendix 1 References

- [1] NLTK Corpus,  
<http://nltk.sourceforge.net/index.php/Corpora>
- [2] sgmlib(Simple SGML Parser), a Python module for parsing Standard Generalized Mark-up Language,  
<http://docs.python.org/lib/module-sgmlib.html>
- [3] Python-module and metric for syllable determination,  
<http://viewvc.red-bean.com/gnoetics/src/syllables.py?revision=1&view=markup>
- [4] EditCentral, a online readability calculator,  
<http://www.editcentral.com/gwt/com.editcentral.EC/EC.html>