



## Readability Index

by

*Thomas Jakobsen, Thomas Skardal*

Supervisor: Morten Goodwin Olsen

Co-supervisor: Annika Nietzio

### **Project report for Web-mining and data analysis in Autumn 2007**

based on report template version 3.0 (2006)

Agder University

Faculty of Engineering and Science

Grimstad, 18 October 2007

Status: Draft

**Keywords:** NLTK, Python, Readability, Language classification

## Version Control

<i>Version</i> <sup>1</sup>	<i>Status</i> <sup>2</sup>	<i>Date</i> <sup>3</sup>	<i>Change</i> <sup>4</sup>	<i>Author</i> <sup>5</sup>
0.1	Draft	15 Oktober 2007	Started writing on background, design spec and requirements	Thomas S Thomas J

---

**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 Background.....</u>	<u>4</u>
1 NLTK - Natural Language ToolKit .....	4
2 Readability tests .....	4
2.1 Language classification .....	4
<u>2 Solution.....</u>	<u>5</u>
2.2 Requirements .....	5
2.3 Design Specification .....	5
<u>Appendices.....</u>	<u>8</u>
<u>2.4 References.....</u>	<u>8</u>

# 1 Background

## 1 NLTK - Natural Language ToolKit

NLTK [1] is a library of open source Python modules, data and documentation for research and development in natural language processing [2]. There are much functionality already implemented that will be useful for us when we're working on this task.

## 2 Readability tests

Readability tests [3] are used to calculate the difficulty of reading and understanding a text. There are several different tests, and they are using different formulas to calculate the readability. These formulas use statistics from the text, such as number of words, number of sentences etc. and some of these can be easily found by using functionality from within NLTK. Some of the tests are language neutral, but there are some tests that is designed for specified languages.

### 2.1 Language classification

To determine which language a text is written in we will be using a Naïve Bayes classifier [4]. This has been done before in other projects in this course. We've been reading the reports, and they will be a good resource for our task. We've also had an exercise concerning classification of newsgroups. There is a Navie Bayes classifier in NLTK. However we have not yet decided whether to use this or to develop our own implementation.

#### Bayes Theorem [5]:

This formula finds the probability of a hypothesis to be true, when an observation is given.

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

## 2 Solution

### 2.2 Requirements

#### Functional

Our solution should provide support for calculating the readability using several known techniques. It should also be able to determine which language the text is written in. We will only focus on English and Norwegian during our project. To accomplish this we've decided that these requirements are needed:

- Determine sentences, syllables, words etc. needed by the different tests.
- Readability tests [3]
  - Automated Readability Index
  - Flesch-Kincaid
  - Coleman-Liau
  - Gunning fog
  - SMOG index
  - Linsear write
  - ...?
- Text classifier to determine the language of a text

#### Non-functional

Since NLTK is written in Python, we are of course also using Python for our implementation. NLTK's developer team has set some coding standards that we have to follow.

### 2.3 Design Specification

This project will be developed using Python as that is the standard language used in the Natural Language tool kit. As IDE we will use Eclipse with the PyDev plug-in [7] which enables easy Python development. For source control we use Subversion. We are consecutively considering the need for unit-testing, but at the current stage we have not started with this. When we start unit-testing we will use the built-in unit testing [6] framework in Python.

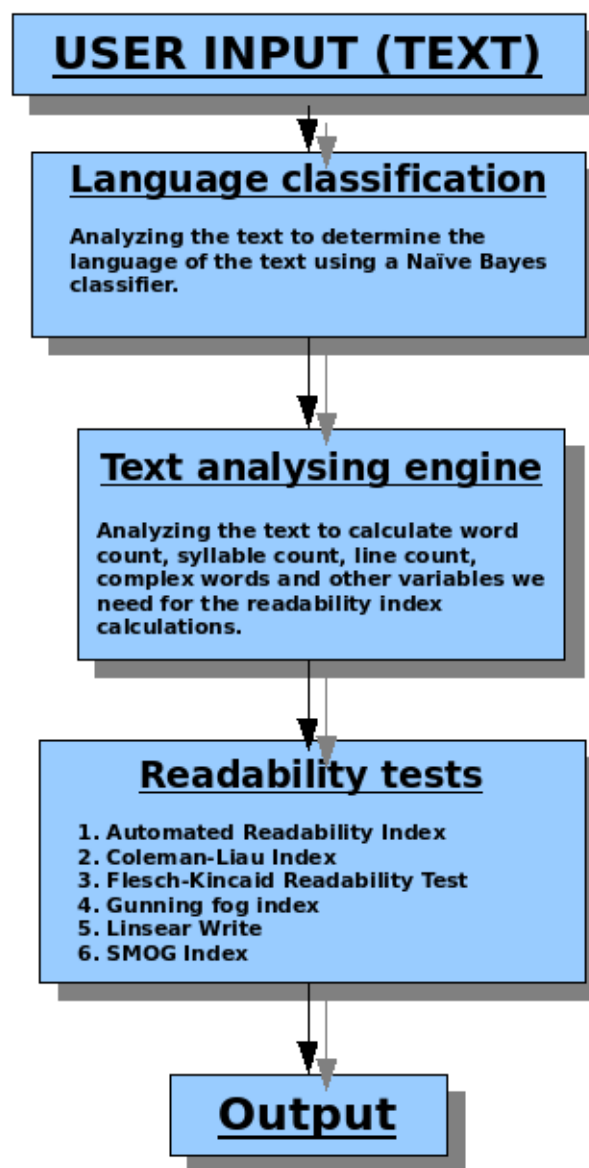
When all the readability methods have been implemented we will start to write unit-tests for each one of the readability tests. Then we will feed our test program with numerous texts of different length and complexity. We will make this readability plug-in as component based as possible. Our main readability engine will then connect and engage the desired components based on user-requests.

One of the crucial areas who requires in-depth testing will be text analyzing. We are dependent on a flawless engine to be able to make correct calculations in the readability tests. The engine needs to give a close-to-perfect response when analyzing the input text. Word count, syllables count, line count and detection of complex words needs to be calculated carefully and correct. An important part of this process will be language-classification. For all this to work flawless we will need a comprehensive test system. In this

process we will take advantage of the English online readability tests that exists and compare our local results to the online results.

Another important part is the language classifier. We will need numerous texts in both English and Norwegian to train it. We might need to make a miner to be able to gather enough texts of different complexities. Then we need solid tests to measure the accuracy of the language classifier.

Here is a flow-chart who illustrates the main flow of events:



## Appendices

### 2.4 References

- [1] \_\_\_\_\_ : \_\_\_\_\_ [<http://nltk.org>]
- [2] \_\_\_\_\_ : \_\_\_\_\_ [[http://en.wikipedia.org/wiki/Natural language processing](http://en.wikipedia.org/wiki/Natural_language_processing)]
- [3] \_\_\_\_\_ : \_\_\_\_\_ [[http://en.wikipedia.org/wiki/Category:Readability tests](http://en.wikipedia.org/wiki/Category:Readability_tests)]
- [4] \_\_\_\_\_ : \_\_\_\_\_ [[http://en.wikipedia.org/wiki/Naive bayes](http://en.wikipedia.org/wiki/Naive_bayes)]
- [5] \_\_\_\_\_ : \_\_\_\_\_ [[http://en.wikipedia.org/wiki/Bayes theorem](http://en.wikipedia.org/wiki/Bayes_theorem)]
- [6] \_\_\_\_\_ : \_\_\_\_\_ [<http://docs.python.org/lib/module-unittest.html>]
- [7] \_\_\_\_\_ : \_\_\_\_\_ [<http://pydev.sourceforge.net>]