Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Probability as readability
A new machine learning approach to readability assessment for written Swedish

by

Johan Sjöholm

LIU-IDA/LITH-EX-A--12/023--SE

2012-06-05

Linköping University
Department of Computer and Information Science

Final Thesis

Probability as readability
A new machine learning approach to readability assessment for written Swedish

by

Johan Sjöholm

LIU-IDA/LITH-EX-A--12/023--SE

2012-06-05

Supervisor: Katarina Heimann Mühlenbock (Språkbanken, University of Gothenburg)
Christian Smith (IDA, Linköping University)
Examiner: Arne Jönsson (IDA, Linköping University)

Avdelning, Institution Division, Department The Human Centered Systems Division Department of Computer and Information Science Linköpings universitet SE-581 83 Linköping, Sweden	**Datum** Date 2012-06-05

Språk Language ☐ Svenska/Swedish ☒ Engelska/English ☐ ______________	**Rapporttyp** Report category ☐ Licentiatavhandling ☒ Examensarbete ☐ C-uppsats ☐ D-uppsats ☐ Övrig rapport ☐ ______________	**ISBN** — **ISRN** LIU-IDA/LITH-EX-A–12/023–SE **Serietitel och serienummer** Title of series, numbering **ISSN** —
URL för elektronisk version		

Titel
Title

Sannolikhet som läsbarhet
En ny maskininlärningsansats till läsbarhetsmätning för skriven svenska

Probability as readability
A new machine learning approach to readability assessment for written Swedish

Författare
Author

Johan Sjöholm

Sammanfattning
Abstract

This thesis explores the possibility of assessing the degree of readability of written Swedish using machine learning. An application using four levels of linguistic analysis has been implemented and tested with four different established algorithms for machine learning. The new approach has then been compared to established readability metrics for Swedish. The results indicate that the new method works significantly better for readability classification of both sentences and documents. The system has also been tested with so called soft classification which returns a probability for the degree of readability of a given text. This probability can then be used to rank texts according to probable degree of readability.

Nyckelord
Keywords

Readability, Natural Language Processing, Computational Linguistics, Machine Learning, Swedish

Abstract

This thesis explores the possibility of assessing the degree of readability of written Swedish using machine learning. An application using four levels of linguistic analysis has been implemented and tested with four different established algorithms for machine learning. The new approach has then been compared to established readability metrics for Swedish. The results indicate that the new method works significantly better for readability classification of both sentences and documents. The system has also been tested with so called soft classification which returns a probability for the degree of readability of a given text. This probability can then be used to rank texts according to probable degree of readability.

Sammanfattning

Detta examensarbete utforskar möjligheterna att bedöma svenska texters läsbarhet med hjälp av maskininlärning. Ett system som använder fyra nivåer av lingvistisk analys har implementerats och testats med fyra olika etablerade algoritmer för maskininlärning. Det nya angreppssättet har sedan jämförts med etablerade läsbarhetsmått för svenska. Resultaten visar att den nya metoden fungerar markant bättre för läsbarhetsklassning av både meningar och hela dokument. Systemet har också testats med så kallad mjuk klassificering som ger ett sannolikhetsvärde för en given texts läsbarhetsgrad. Detta sannolikhetsvärde kan användas för rangordna texter baserad på sannolik läsbarhetsgrad.

Acknowledgments

I would like to thank my supervisors Katarina Mühlenbock and Christian Smith for their continuous support during the writing of this thesis, especially for helping me with tools, data, reusable code and proof reading. I would also like to thank my examiner Arne Jönsson for his help and enthusiasm as well as Åsa Wihlborg for support and patience. I would like to thank Kristoffer Mellberg for proof reading, Erik Prytz for support and inspiration and lastly my opponent Robin Keskisärkkä for lots of helpful comments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatically assessing the readability of written text has been a research area in computational linguistics as long as the field has existed. In 1949 Dale and Chall defined readability as "the sum total (including all the interactions) of all those elements within a given piece of printed material that affect the success a group of readers have with it" [Dale and Chall, 1949]. From this follows that readability is a function of both features of the text and the proficiency of the reader. The actual features of text which imply degree of readability are under debate and different features seem to be more or less relevant for different languages. For Swedish there are a number of formulas for calculating degree of readability but the most common among them, the LIX formula, is based on surface structure alone.

During the last few years machine learning has been applied to the field of readability assessment. Different kinds of classifiers have been experimented with and many different features have been used to train the models. However, most of these studies have used a relatively small subset of all possibly relevant features, often based on one of six levels of text structure: surface, lexical, morpho-syntactic, syntactic, semantic or discourse. The group of people with some kind of reading difficulties, about 25 % of the Swedish populace [OECD, 1994], is however very heterogeneous and features which might affect one type of reader's ability to read and understand a text might not matter at all to another type.

Studies have shown that the aforementioned LIX formula is overly simplistic and though it may give a hint about the degree of readability it is not sufficient to handle this heterogeneousness [Heimann Mühlenbock and Kokkinakis, 2010]. Therefore a system flexible enough to handle the different types of readers should be developed. Lacking Swedish corpora tagged for readability by different types of readers it is not possible to do this today. However, it is possible to use a general "easy-to-read" corpus to develop and test a system which will be easy to adapt whenever more reader type specific corpora is available.

This study aims to develop and test a proof-of-concept for such a system using machine learning, more specifically soft classification, and also to investigate the possibilities to integrate it in a search engine for the Webblättläst project, see the next section for background to the Webblättläst project.

1.1 Background

This thesis is part of a larger project requested and funded by Internetfonden.

Internetfonden is a Swedish foundation working to promote the use of the Internet among the Swedish populace. One of their goals is to make the Internet more accessible to people with reading disabilities and second language learners. In this vein Internetfonden is funding the Webblättläst project at the Department of Computer and Information Science at Linköping University. The project aims to create a search engine capable of ordering relevant websites based on their degree of readability.

Related to this are a number of sub-projects in usability, interface design and the development of new ways to assess readability. This thesis is part of this last sub-project about development of new readability metrics.

1.2 Purpose

The purpose of this thesis is to design and implement a classifier for readability assessment based on four levels of analysis. This will then be tested and measured against traditional readability assessment formulas for Swedish. A secondary purpose is to evaluate the possibility of using such a system to sort the results from a search engine by using soft classification. If this is not possible the system might, by applying feature selection among the extracted features, be able to provide a simpler formula which can outperform the ones in current use for general readability.

1.3 Goal

The goal of the thesis is to develop a Java module which can take a Swedish text as input, analyse it and provide a percentage score of how likely it is that the text is "easy-to-read". The system will then be evaluated and measured against current standard formulas such as LIX, OVIX and Nominal ratio.

1.4 Expectations and requirements

This thesis is mainly expected to test the feasibility of a machine learning approach to readability assessment for Swedish and whether it stands up to traditional metrics. It is also expected to lay the groundwork for a readability assessment system which can be trained for specific user groups with different reading disabilities. The prototype is not expected to be suitable for production use in a search engine because of the time consumption of the external pre-processing tools. However, the thesis should give an inkling about the feasibility of a machine learning based readability metric for search engines.

1.5 Limitations

As the time provided for this thesis is limited a large part of the pre-processing (POS-tagging, dependency parsing, etc.) will be done using an existing tool. This tool is designed to be as exact as possible for research purposes and is not optimized for speed, as a result of this the resulting system will not be fast enough to use in an actual search engine.

At the time of writing no readability tagged corpora for specific reader types exist, therefore a general "easy-to-read" corpus, together with a corpus with standard Swedish news texts will be used to test the feasibility of the approach.

1.6 Thesis outline

Chapter 1 : Introduction

This chapter will elaborate on the goal for this thesis and introduce its approach and limitations.

Chapter 2 : Background to Readability Assessment

This chapter will elaborate on the background of readability assessment. Some established metrics and recent developments will be covered.

Chapter 3 : Background to machine learning

This chapter will elaborate on the background of machine learning relevant for this thesis. Some different approaches and types of algorithms will be covered.

Chapter 4 : Approach

This chapter will elaborate on the approach to readability assessment underlying this thesis.

Chapter 5 : Implementation

This chapter will explain the inner workings of the actual system implemented for this thesis.

Chapter 6 : Results

This chapter will present the results from the evaluations as well as some analysis and discussion of those results.

Chapter 7 : Conclusion

The final chapter will summarize the thesis and the conclusions that can be drawn. Some future work will also be suggested.

Appendices

The appendices contain an example of the XML-code generated by the preprocessor (Appendix A) and some examples of erroneously classified documents (Appendix B).

Chapter 2

Background to readability assessment

This chapter will elaborate on the background of readability assessment. Some established metrics and recent developments will be covered.

2.1 The problem of readability assessment

The problem of readability assessment is the problem of mapping from a text to some unit representing it's degree of readability. This in itself consists of a number of problems.

Firstly, what features of a text are relevant for readability assessment? Chall [Chall, 1958] defined the four general properties vocabulary load, sentence structure, idea density and human interest but how are these represented in actual texts? Which features are independent and which are related and are related features redundant? Are all features as important for all readers?

Secondly, how are these features extracted from the text? Today a large number of lemmatizers, POS-taggers, grammar parsers and other analysis tools exist but which of them are suitable when degree of readability is assessed? Some tools are free, some can be freely licensed for research purposes, and some are purely commercial. The tools are implemented in a number of different languages using a number of different algorithms, but which ones fit well together and are easy to integrate? Are their computational complexity high and what degree of analysis is reasonable when computation time is taken into account?

Thirdly, how should these features be weighted to provide an actual metric? If a simple formula can be constructed, how does it relate to the degree of readability? If a simple formula is not feasible due to the amount of features, how can the degree of readability be modelled?

Lastly, how is degree of readability represented in an easily comprehensible way? A numerical value makes comparison simple but might be difficult for a reader to understand. A genre classification is more understandable for a reader but might make comparison hard.

2.2 History of readability assessment

Readability assessment has been a field of study since the 1920's [Feng et al., 2009]. In the U.S. during the 1920's and 1930's a number of studies were published about how to assess readability of texts for a number of different applications. Vogel and Washburne described a method for determining grade placement for books for children in 1928 [Vogel and Washburne, 1928], and as early as 1934 Dale and Tyler published a study on the text properties relevant for adults with limited reading ability [Dale and Tyler, 1934].

The field started to gain real momentum in the 1940's. Among other achievements, the article "The concept of readability" by Dale and Chall [1949] contained a definition of readability which is still used. The article defines readability as "the sum total (including all the interactions) of all those elements within a given piece of printed material that affect the success a group of readers have with it. The success is the extent to which they understand it, read it at optimal speed, and find it interesting."

Also in the 1940's the first readability metrics still in use were proposed, such as the Flesch Reading Ease test [Flesch, 1948]. The Flesch Reading Ease test grades texts on a scale from 0 to 100, the Flesch Reading Ease Score (FRES), where a lower score implies a more advanced text. In the Flesch Reading Ease test n(w) is the number of words, n(se) is the number of sentences and n(sy) is the number of syllables.

$$FRES = 206.835 - 1.015\frac{n(w)}{n(se)} - 84.6(\frac{n(sy)}{n(w)})$$

Figure 2.1 – The Flesch Reading Ease test.

The Flesch Reading Ease test was updated by Kincaid et al. [1975] to give a score corresponding to a grade level in the U.S. education system. The new formula is called the Flesch-Kincaid Grade Level.

The Flesch Reading Ease test was soon followed by the Dale-Chall formula which utilizes a list of "easy" words which is used in the assessment. The Dale-Chall Readability Score (DCRS) differs from the FRES in several ways. Firstly, a higher number indicates a more advanced text, secondly, the score is not limited and can, theoretically, have an infinitely high value (however, to reach infinity an infinitely long sentence is required). In the Dale-Chall formula n(dw) is the number of difficult words, n(w) is the number of words and n(s) is the number of sentences. Difficult words are any words which do not occur on the list of easy words.

$$DCRS = 0.1579\frac{n(dw)}{n(w)} + 0.0496(\frac{n(w)}{n(s)})$$

Figure 2.2 – The Dale-Chall formula.

The Dale-Chall formula was updated in 1995 with a new list of simple words [Chall and Dale, 1995].

Both these metrics spawned a number of related formulas utilizing different normalizations. Some of the most well known are the Gunning FOG index (1952), the Coleman-Liau index which was specifically designed for automated assessment of readability [Coleman and Liau, 1975], the SMOG formula [McLaughlin, 1969], and the Fry readability formula [Fry, 1968]. All of these scores correspond to the U.S. grade level thought necessary for full comprehension of the text.

During the 1950's linguistics started to turn it's attention to syntax under the influence of the Chomskyan revolution. At the time syntactic parsing was not an automated process but the possible influence of syntax on the degree of readability was investigated by Yngve [1960].

In 1958 Chall concluded that "only four types of elements seem to be significant for a readability criterion". These were vocabulary load, sentence structure, idea density and human interest [Chall, 1958]. However, these terms are quite abstract and not at all trivial to convert to numerical features.

The formulas mentioned above were widely accepted as good enough for practical use for a long time. Also, the scarcity and price of computational power might also have hampered the success of metrics utilizing more advanced analysis, leading to these traditional metrics still being used in most commercial applications for automatic readability assessment. However, by the early 1980's studies questioning the, by then established, approach of grading texts based on surface structure alone, were being published. Davison and Kantor [1982] showed that these measures did not always represent the actual complexities of written English.

2.3 Established metrics for Swedish

As mentioned above, the problem of readability assessment is the problem of mapping from a text to some value representing its degree of readability. This can be done in a number of different ways. One way is the traditional construction of formulas generating a numerical value. This is the way that the traditional Swedish metrics LIX, OVIX and Nominal ratio works. These numerical values can be used for ranking texts according to their degree of readability but on their own they might be harder to interpret. For the formulas there are tables representing a hierarchy of genres corresponding to ranges of values, however, these ranges might be shifted for different reader types.

Also, the traditional Swedish readability metrics manage to capture at most one of the four categories defined by Chall (see previous section) determining the degree of readability of a text. Vocabulary load in the case of OVIX and idea density in the case of Nominal ratio, while LIX does not really capture any of the categories except perhaps as a very rough metric of sentence structure. None of these metrics capture human interest or sentence structure in a deeper sense. Some research has been done combining and comparing these metrics and also approximating human interest and sentence structure by the average number of proper nouns and the average sentence length respectively [Heimann Mühlenbock and Kokkinakis, 2010].

2.3.1 LIX

The readability metric most commonly used in Swedish is the Läsbarhetsindex (abbreviated and commonly referred to as LIX and translated as Readability index, not to be confused with the metric of the same name mentioned below) formula introduced by Björnsson [1968]. The metric does not fit in any of Chall's categories but is related to international metrics such as Flesch-Kincaid readability test and the Coleman-Liau test which measure similar features. However, LIX differs from most other metrics in this family by counting the number of letters instead of the number of syllables when word length is considered. LIX can be interpreted as the percentage of "long" words added to the average number of words per sentence. In the LIX formula n(w) is the number of words, n(s) is the number of sentences and n(w $>$ 6) is the number of words longer than 6 characters.

$$LIX = \frac{n(w)}{n(s)} + (\frac{n(w > 6)}{n(w)} * 100)$$

Figure 2.3 – The LIX formula.

A number of variations of LIX has been proposed and tested, such as Readability index (RIX) [Anderson, 1983] and Karaktärsindex (KIX, translated as Character index, where character refers to the character of the text, not letters) [Larsson, 1987]. In Sweden LIX is used almost exclusively to the detriment of other metrics. However, as shown by recent research [Heimann Mühlenbock and Kokkinakis, 2010], it does not capture enough language features to be useful when the heterogeneity among groups with reading difficulties is considered.

2.3.2 OVIX

Another proposed readability metric for Swedish is the OVIX - Ordvariationsindex (translated as Word variation index) formula [Hultman and Westman, 1977]. As a metric the OVIX formula has not been able to challenge the near total dominance of the LIX metric and is not in common use. However, it has met some success in research.

The algorithm is similar to Honoré [1979] in that it is based on logarithmic variables. However, this has the weakness that if every word is unique a division by zero occurs. This is generally not a problem with documents but with single sentences it is very common. While LIX does not fit into any of Chall's categories OVIX is mainly a vocabulary load metric. Technically OVIX's approach to measuring vocabulary load is a matter of calculating the lexical variation. In the OVIX formula n(w) is the number of words and n(uw) is the number of unique words.

$$OVIX = \frac{log(n(w))}{log(2 - \frac{log(n(uw))}{log(n(w))})}$$

Figure 2.4 – The OVIX formula.

The OVIX formula measures something very different from LIX but still only captures one of Chall's properties. A recent study has also shown that LIX and OVIX do not, at least on the surface, correlate, and would produce different rankings if used individually to order texts according to their respective degree of readability [Heimann Mühlenbock and Kokkinakis, 2010].

2.3.3 Nominal ratio

Besides the LIX and OVIX formulas there is one other readability metric for Swedish which is used to some degree in research. Nominal ratio is mainly a measure of information density. In this case by comparing the number of nouns, prepositions and participles to the number of pronouns, adverbs and verbs [Hultman and Westman, 1977]. In the Nominal ratio formula n(noun) is the number of nouns, n(prep) is the number of prepositions, n(pro) is the number of pronouns, n(adv) is the number of adverbs and n(verb) is the number of verbs.

$$NR = \frac{n(noun) + n(prep) + n(part)}{n(pro) + n(adv) + n(verb)}$$

Figure 2.5 – The Nominal ratio formula.

Nominal ratio has the same weaknesses as LIX and OVIX in that it is only a measure of at most one of Chall's properties. However, it is also both a linguistically more involved and a computationally more expensive metric as part-of-speech tagging is necessary.

2.3.4 Combining traditional metrics

In the last years a few studies have experimented with combining classical metrics and adding some simple features not covered by these metrics. Heimann Mühlenbock and Kokkinakis [2010] showed that a combination of traditional Swedish metrics and such extra features could make it possible to assign better readability scores and better fit the grading of texts to different groups of readers.

2.4 Recent international developments in research

The years since 2000 have seen quite a few developments in the field of readability assessment. A large number of studies have taken place trying to find out which features might be, or are most, relevant for the degree of readability. These studies are often based on modern methods of text analysis such as automatic part-of-speech tagging, syntactic parsing and higher levels of analysis on semantic and discourse level. A number of studies on how these features might be used to train readability assessing systems have also been published.

The use of language models to assess the degree of readability was introduced by Collins-Thompson and Callan [2004]. By creating language models for texts suitable for different U.S grade levels, texts could be analysed for word frequencies and compared to these language models.

Phrase grammar features have also been used in a number of studies, most famously the average parse tree height, average number of verb phrases, noun phrases and SBARs (sub-ordinate clauses, SBAR is used by the Penn Treebank and a number of related treebanks) per sentence used by Schwarm and Ostendorf [2005]. Heilman et al. [2008] went further and used patterns of subtrees of different depths based on application of context-free grammar parsers as features. This could perhaps be considered a combination of a unigram language model and a syntactic structure approach, where the unigrams consist of tree patterns rather than words.

The development of new parsers, such as the dependency parsers generated by the MaltParser parser-generator [Nivre et al., 2006], makes new metrics based on dependency grammar analysis available for automated assessment. Liu [2008] proposed that dependency distance, which can be extracted from such parsed text, might be a good feature for readability assessment.

An Italian team designed and tested a system, READ-IT, which could identify easy-to-read texts in Italian using classification and advanced features. The system performed very well and demonstrated an accuracy of 98 % [Dell'Orletta et al., 2011]. See Section 6.6 on page 61 for a comparison between some of the READ-IT results and the results of this thesis. Heilman et al. [2007] have also done a number of studies using classification and advanced features. It should be noted that results have varied between studies when it comes to deciding how relevant grammatical features actually are.

When it comes to suitable U.S. grade level identification the work of Petersen is highly relevant. Using a combination of statistical language models (n-grams) and an SVM, detectors (single class classifiers) to detect a text's corresponding grade level were constructed [Petersen, 2007]. Feng did something similar but also included a number of higher level discourse features [Feng, 2010].

A number of recent studies have also analysed higher levels of features such as Feng's investigation of discourse features mentioned above. Dufty et al. [2004] tested a system analysing the cohesion of a text which showed that cohesion might be a good indication of the degree of readability. Feng et al. [2009] (the same Feng) also published a comprehensive study of various features for degree of readability assessment, yet again using suitable grade levels for comparison.

Researchers have also used regression to attempt to calculate the degree of readability of text by using feature vectors, similar to those used in classification, paired with the suitable U.S. grade level. However, the results have not yet measured up to the aforementioned detector approach when it comes to identifying grade level [Petersen and Ostendorf, 2009].

Chapter 3

Background to machine learning

This chapter will elaborate on the background of machine learning relevant for this thesis. Some different approaches and types of algorithms will be covered. Throughout this chapter, and the rest of the thesis, the term machine learning will be used rather than the more indistinct term data mining.

3.1 Relevant approaches to machine learning

For the problem at hand three different classes of machine learning algorithms might be relevant. Machine-learned ranking, regression analysis and classification. This thesis will focus mainly on classification, however, classification can be implemented by regression and in that "regression-under-the-hood" sense regression will be somewhat explored.

3.1.1 Ranking

Machine-learned ranking (MLR) is probably the most obvious approach when it comes to sorting search results. A MLR algorithm trained to assess readability could take all the search results, represented as a list of feature vectors, and return the list sorted on the degree of readability of the text. However, this approach has a number of drawbacks.

Firstly, the available corpora are not list ranked, which means that the most efficient class of ranking algorithms, the list based approach [Liu, 2009], is not available. This problem also exists for the pair based approach, which leaves only the point based approach.

Secondly, the aim of this thesis is to develop a metric which can score a text based on its degree of readability. MLR does not supply a score but rather a total ordering of the data points. Also this is not compatible with the Webblättläst interface which expects a numerical score.

Thirdly, MLR is a relatively new sub-field in machine learning and according to a recent study MLR might not be a "solved problem" [Chapelle et al., 2011] and not mature enough for use outside optimized domains.

Because of the reasons mentioned above MLR will not be explored in this thesis. However, if the aforementioned problems could be solved this might be a viable approach in future research.

3.1.2 Regression

Regression analysis is the attempt to establish how independent variables influence dependent variables. For the purposes of this thesis, this can be simplified as generating a mathematical formula based on the variables in the training data. This is done by treating data points as equations with the feature vector on the left side and a numerical value, representing the score of this data point, on the right side. The resulting formula could then be used for scoring new texts.

Regression on its own would generate a formula which could be used to score texts, however, since the training data consists only of two classes the training scores will not be distributed evenly. This is a problem for traditional linear regression, however, other regression algorithms constructed specifically for classification, such as logistic regression, performs significantly better under such circumstances [Witten et al., 2011]. Regression can, as mentioned, be used for classification purposes, either on its own or combined with other methods, and this is the main way it will be used in this thesis as the results will be more comparable with the other classification algorithms.

3.1.3 Classification

The classification problem is one of the oldest problems in machine learning. The goal is to decide what category a data point belongs to based on its features. A large number of different algorithms using different mathematical representations and processing have been developed for classification. This approach is tempting and has proven useful in similar studies [Dell'Orletta et al., 2011]. However, traditional, or hard, classification assigns a class and not a numerical value which can be used for scoring. One solution to this is to use a multiclass classifier with different score associated with each class, however, just as with regression, graded training corpora is needed to train such a model. A more feasible approach is soft classification.

3.1.4 Soft classification

Soft classification, also often called probabilistic or distribution classification, is a modern sub-field of traditional classification. Instead of returning a class to which the current data point belongs a soft classifier returns a mapping from each class to the probability that the data point belongs to that class. Using a soft classifier trained on "easy-to-read" texts and "normal" texts, the probability that a text is *not* "easy-to-read" can be represented as a value between 0 and 1. This value can then be used as a score where a higher score implies a more advanced text.

3.2 Relevant approaches to classification

There are a great number of algorithms in the machine learning field used for the classification problem. For the purposes of this thesis these can roughly be divided into four categories. These categories are debatable and do not correlate to any established taxonomy of classification algorithms. In all categories there exist algorithms which have versions designed for soft classification.

See Section 4.3 on page 29 for the actual algorithms used in this thesis.

3.2.1 Statistical approach

It should be noted that all machine learning could be considered to be statistical. However, for the purposes of this thesis this refers to models which use explicitly learned probabilities to calculate the compound probability of a specific class, and classifying based on this probability. Two of the most well known methods within the statistic approach are Naive Bayes and Bayesian Network algorithms.

Bayesian Networks require that there are some prior knowledge about the variables and their dependency relations. There are ways of discovering such relations using structure learning [Rebane and Pearl, 1987], but this lies outside the scope of this thesis.

Naive Bayes however is an approach which naively assumes independence of all variables. Any actual relationships among the variables are ignored in a way which could be considered blunt. As such, Naive Bayes is a comparatively simple algorithm but has been shown to work very well in real applications [Marsland, 2009], especially if feature selection is used [Witten et al., 2011].

Due to its simplicity, Naive Bayes is often used as a baseline when new classification schemes are evaluated.

3.2.2 Network approach

The network approach to classification utilizes a directed network of choice points to generate a classification.

Most obvious is the Decision Tree family of algorithms which constructs a tree structure where every branch node is a choice point, testing some feature and choosing a path based on the result, and every terminal node is a classification. Decision Tree is a traditional classification approach which actually existed in manually constructed, expert designed, versions before machine learning was applied in the generation of trees. It is also one of the most transparent approaches and is relatively easy to follow for a human user. In Figure 3.1 on page 20 is a simple example of a decision tree showing the most probable outcome for a Titanic passenger based on some simple attributes.

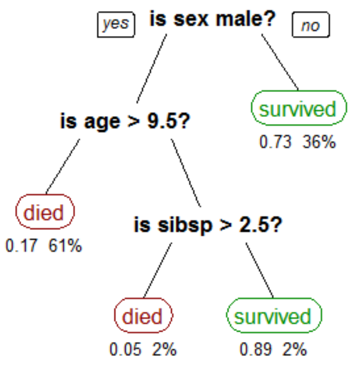

Figure 3.1 – A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of survival and the percentage of observations in the leaf. Created using the R programming language, Source: Wikimedia Commons, Author: Stephen Milborrow.

Another family of network algorithms is the Neural Network family which applies a connectionist perspective to classification [Rosenblatt, 1962]. By using networks of perceptrons, micro classifiers analysing singular or pairs of features, with propagation and feedback, relatively exact models can be constructed. However, unlike Decision Trees these Neural Networks are almost impossible to follow for a human user and has at times been referred to as a "black box" approach [Setino et al., 2000].

3.2.3 Algebraic approach

A large number of algebraic algorithms have been developed for classification. These are based on representing data as points or vectors in feature space and using matrix methods to calculate the classification. A problem for most algebraic algorithms is that badly normalized input might lower the precision of the classifier.

Most, if not all, forms of regression, such as linear regression using least squares, could be considered examples of algorithmic models. A system of equations, usually overdetermined, is constructed and regression is used to find a formula approximating the distribution of the data points. Some versions of regression can be used for classification by representing each class numerically and using these as the known constant side of the equations. While there are some problems with doing binary classification using traditional linear regression, such as the aforementioned least squares method, more advanced methods, such as logistic regression, have proved efficient for classification [Witten et al., 2011].

The Vector Space Model is a well established tool in computational linguistics being utilized in the field as early as 1975 [Salton et al., 1975]. By representing documents as vectors in feature space classification becomes a relatively simple problem of comparing vectors. The features used are often bag-of-word sets where the vectors represent frequencies of words [Eldén, 2007]. Usually, the distance is measured by calculating the cosine angular distance.

Related to the Vector Space Model is the traditional Nearest Neighbour (NN) algorithm. Instead of comparing vectors with the cosine angular distance the NN algorithm takes a new, unclassified, data point and classifies it according to the class of the closest classified data point. A popular version is the kNN algorithm which uses the k nearest data points and classifies according to the majority of these.

A relatively new approach is the Support Vector Machine (SVM) which has existed in its modern form since 1995 [Vapnik and Cortes, 1995]. Like the NN algorithm SVMs view data as points in feature space. In this space a hyperplane which separates the feature space into two parts, representing two different classes, is calculated. In Figure 3.2 on page 21 is simple two-dimensional example of a SVM.

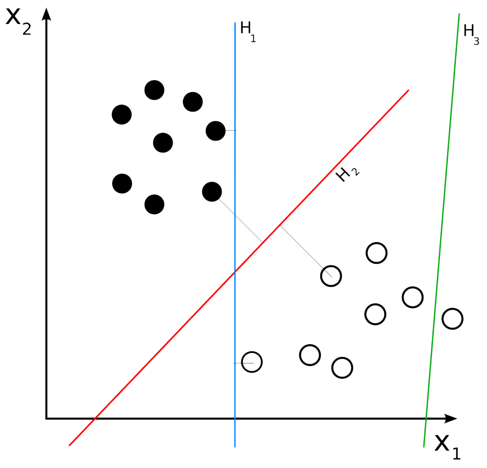

Figure 3.2 – Graphic showing three hyperplanes in 2D. The hyperplane H3 doesn't separate the two classes at all. H1 separates them with a small margin and H2 with a large margin. Created using vector graphics, Source: Wikimedia Commons, Author: Cyc.

3.2.4 Other approaches

Related to the aforementioned Decision Tree family of algorithms is the Classification Rules family of algorithms. While making classifications in Decision Trees consists of traversing a tree by choosing children based on tests, classification by rules is done by matching new data points against learned rules, represented as conjunctions of boolean variables. These variables are generally made up by the same kinds of tests as would be used in a binary Decision Tree. Whenever a rule-conjunction holds true the corresponding class is chosen.

Chapter 4

Approach

This chapter will elaborate on the approach to readability assessment underlying this thesis.

4.1 Probability as readability

As covered in Section 2.4 on page 14 many recent studies within degree of readability assessment in the U.S. have focused on identifying a suitable grade level. This is in part because many of the established metrics for American English, such as Flesch-Kincaid, output such a grade level. It is also in part because there exist an American corpus, Weekly Reader, tagged with precisely such a "suitable grade"-system.

This means that there are sets of texts ordered by degree of readability on which regression can be more effectively applied. It also means that detectors (single class classifiers) can be trained for each level. For Swedish nothing comparable exists. There are corpora which intuitively should have different degrees of readability relative to each other but nothing concrete.

As graded corpora does not exist for Swedish in any obvious and accessible form a substantially different approach will be examined. If the assumption is made that the degree of readability of a text is proportionate to the probability that the text is classified as easy-to-read by a, theoretically, perfect classifier the problem becomes one of constructing such a classifier. At this time there is no easy way of testing the assumption, but before the assumption can be tested a system which can at least calculate the probability that a text is easy-to-read must be constructed. Of course, the system will not constitute a perfect classifier but a good enough classifier should be able to calculate probabilities in such a way that texts within a limited span can be ranked by these probabilities. The goal of this thesis is to implement and test such a limited classifier for Swedish text.

4.2 Relevant features

As this is one of the first studies on machine learning based readability assessment for Swedish, possibly relevant features have been cherry-picked from four categories of features used in recent research on other languages. A large number of possible features, based on n-gram language models, phrase grammar syntax, compositional semantics and discourse level features etc., have been left out due to time constraints.

Besides Chall's four properties (see Section 2.1 on page 9) it is possible to divide text analysis into another dimension more suitable for describing data extraction. This dimension is defined by how the features are represented in the text and what kind of analysis is required to extract them. These different kinds of analyses can be divided into four levels of increasing linguistic involvement. The levels are shallow, or surface, structure, lexical composition, morpho-syntactic structure and syntactic structure. These categories, and many of the features, are influenced by prior research covered in Section 2.4 and in particular by Dell'Orletta et al. [2011].

All these features might not be necessary for classifying easy-to-read texts. For instance, Dell'Orletta et al. [2011] showed that a model only using the first three levels of analysis, that is, excluding the syntactic structure, actually can perform better then a full model on Italian texts. However, the features relevant for easy-to-read classification might differ from the features relevant for deciding degree of readability for some specific reader group. Therefore, feature selection, based on these features, should perhaps be performed before training with reader type specific corpora in future developments.

4.2.1 Shallow text features

The shallow text features are the main features traditionally used for simple readability metrics. They occur in the "shallow" surface structure of the text and can be extracted after tokenization by simply counting tokens and characters. They include:

- Word length calculated as the average number of characters per word.
- Word length calculated as the average number of syllables. The number of syllables is approximated by counting the number of vowels.
- Sentence length calculated as the number of words in the sentence.

Longer sentences, as well as longer words, tend to predict a more difficult text as exemplified by the success of the LIX metric and related metrics for English. These types of features have been used in a number of readability studies based on machine learning [Feng, 2010] and as baseline when evaluating new features [Pitler and Nenkova, 2008].

4.2.2 Lexical Features

The lexical features are those features based on lexicology, in this case, categorical word frequencies. The word frequencies can be extracted after lemmatization and

50	DI	någon	S
51	RG	också	C
52	S	vid	C
53	V	säga	C
54	S	under	C
55	NCN	år	C, H
57	V	se	C, H
58	V	gå	C, D
60	RG	mycket	C
61	V	ta	C
62	RG	här	C
63	RG	nu	C
64	PH	vad	C
65	CC	hur	C
65	RG	hur	C
65	RH	hur	C
66	S	mot	C
67	S	efter	C

Figure 4.1 – SweVoc: Entries 50 to 67.

are calculated using a basic Swedish vocabulary developed by Heimann Mühlenbock [forthcoming] named SweVoc. SweVoc is comparable to the list used in the classic Dale-Chall formula [Dale and Chall, 1949] for English and developed for similar purposes, however special sub-categories have been added. See Figure 4.1 on page 25.

The total vocabulary comprises ≈ 8,000 word lemmas, discounting a large number of supplementary words (category S) leaves ≈ 4,000. These ≈ 4,000 word lemmas are subdivided into a core vocabulary of ≈ 2,000 words (category C), ≈ 500 words denoting everyday objects and phenomena (category D), 1,000 words highly frequent and dispersed in a balanced corpus (category H) and additionally ≈ 500 words highly frequent in a corpus from the web (category K, not seen in Figure 4.1). Some lemmas belong to more than one category, such as word 57 (gå) in 4.1. In this thesis only the lemmas in the C, D and H categories are treated individually due to uneven distribution of K and S lemmas in previous research [1]. The following ratios are considered:

- SweVoc lemmas fundamental for communication (category C)
- SweVoc lemmas for everyday use (category D)
- SweVoc other highly frequent lemmas (category H)

[1]Based on personal correspondence with my advisor Katarina Heimann Mühlenbock who is the creator of SweVoc

- Unique, per lemma, SweVoc words in the sentence.

A high ratio of SweVoc words should indicate a more easy-to-read text. The Dale-Chall metric [Chall and Dale, 1995] has been used as a similar feature in a number of machine learning based studies of text readability for English [Feng, 2010; Pitler and Nenkova, 2008]. The SweVoc metrics are also related to the language model features introduced by Schwarm and Ostendorf [2005] and used in a number of studies since [Heilman et al., 2008].

4.2.3 Morpho-syntactic features

The morpho-syntactic features relate to a morphology based analysis of text. For the purposes of this thesis this analysis consists of part-of-speech tagging. This part-of-speech tagging is then used as the basis for a number of features. They consist of:

- Unigram probabilities for the 26 different part-of-speech tags in the document, that is, the ratio of each part-of-speech, on a per token basis, as individual features. Such a unigram language model based on part-of-speech, and and similar metrics, has shown to be a relevant feature for readability assessment [Heilman et al., 2007] [Petersen, 2007] [Dell'Orletta et al., 2011].
- Lexical density, calculated as the ratio of content words, on a per token basis, in the text. Such a metric has been used in a number of related studies [Alusio et al., 2010]. A related metric is the Nominal ratio metric which is an established metric for Swedish [Hultman and Westman, 1977].

4.2.4 Syntactic features

The most linguistically advanced category of features are the syntactic features. These features depend on a syntactic parsing of the text. The syntactic analysis in this thesis, as in Dell'Orletta et al. [2011], is based on a dependency grammar view of syntax and extracted by dependency parsing using a MaltParser [Nivre et al., 2006] generated parser. An example of a sentence parsed with this parser is found in Figure 4.2 on page 28.

A phrase grammar approach, with a phrase grammar parser, such as the one proposed in Nenkova et al. [2010], might be also be relevant for future research as it has been shown to be relevant in many studies [Heilman et al., 2007; Feng et al., 2010] but is not used in this thesis.

The syntactic feature set is extracted after dependency parsing. They consist of the following features:

- The average dependency distance in the document, that is, the length of a dependency link between a dependent token and its head, calculated as the difference between their positions in a sentence. A longer average dependency distance could indicate a more complex text. In the sentence in Figure 4.2 the dependency distance between "gick" and "på" is 2. This has been proposed as a readability metric by Liu [2008]

- The average total length of dependency links in all sentences in the document, also based on Liu [2008]. In the sentence in Figure 4.2 the total length of dependency links is 5.

- The ratio of right dependencies to total number of dependencies in the document. This is calculated by counting the dependency links in which the head word occur after the dependent word in their sentence. A high ratio of right dependencies could indicate a more complex text. In the sentence in Figure 4.2 "De" represents a right dependency as it's head word, "gick", occur later in the sentence.

- The average sentence depth. A sentence's depth is calculated as the depth of the tree consisting of all dependency links in the sentence. Sentences with deeper dependency trees could be indicative of a more complex text in the same way as phrase grammar trees has been shown to be [Petersen and Ostendorf, 2009]. Parse tree depth was proposed as a feature influencing degree of readability by Yngve [1960].

- The unigram probabilities for the 63 dependency types resulting from the dependency parsing, on a per token basis. These unigram probabilities are extracted by calculating the ratios of the different syntactic dependency types, such as predicate, direct object, etc. to the total number of tokens. In the sentence in Figure 4.2 the dependency types ROOT, SS, NA, 0A and PA each have a ratio of 0.2, every other dependency type have a ratio of 0. This feature is related to the phrase type rate used by for instance Nenkova et al. [2010].

- The ratio of sentences with a verbal root, that is, the ratio of sentences where the root word is a verb to the total number of sentences. The sentence in Figure 4.2 has a verbal root. This is, however, not obvious from the figure as part-of-speech is not represented. This feature was proposed by Dell'Orletta et al. [2011].

- The average arity of verbs in the document, calculated as the average number of dependents per verb. This is calculated by counting the number of dependency relations where the head word is a verb. In the sentence in Figure 4.2 the average verbal arity is 3 as the root word, "gick", is the only verb. This feature was proposed by Dell'Orletta et al. [2011].

- The ratios of verbs with an arity of 0-7, that is, the ratio of verbs with an arity of 0 as one feature, the ratio of verbs with an arity of 1 as another feature and so on. This feature was proposed by Dell'Orletta et al. [2011].

- The ratio of subordinated clauses to total number of clauses in the document. Subordinated clauses are identified by the UA dependency relation. The sentence in Figure 4.2 has no subordinated clause. The frequency of "SBAR"s (subordinated clauses in phrase grammar treebanks such as the Penn Treebank) has been used as a metric by a number of studies utilising phrase grammar analysis [Schwarm and Ostendorf, 2005] [Petersen, 2007],

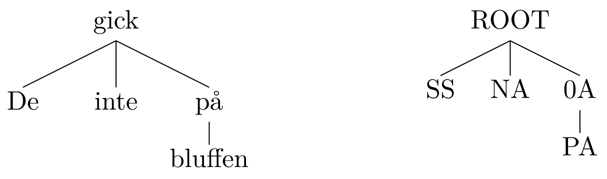

Figure 4.2 – A dependency tree for the sentence "De gick inte på bluffen." ("They were not fooled by the bluff."), to the left are the tokens and to the right their dependency grammar roles.

this feature can be considered a dependency based counterpart to that feature.

- The ratio of subordinated clauses occurring after the main clause in their sentences to the total number of subordinated clauses in the document. Sentences with post-main clause subordinated clauses could indicate a more easy-to-read text [Dell'Orletta et al., 2011].

- The average number of tokens per clause in the document. This is related to the shallow feature average number of tokens per sentence.

- The average number of nominal pre-modifiers and the average number of nominal post-modifiers per sentence in the document, as two distinct features. Pre- and post-modifiers are identified by the AT and ET dependency relations respectively.[2]

- The average number of prepositional complements per sentence in the document. Prepositional complements are identified by the PA dependency relation. The sentence in Figure 4.2 has 1 prepositional complement.[3]

4.2.5 Necessary preprocessing

Generally, the preprocessing steps necessary to extract the relevant features are tokenization, lemmatization, part-of-speech tagging and dependency parsing. A custom preprocessor performing these steps one at a time would be possible and was done, for instance, in the the CogFLUX project [Rybing and Smith, 2010]. However, for this thesis a tool, not made publicly available at the time of writing[4], from Språkbanken at University of Gothenburg is used. The Korp corpus import tool performs all preprocessing up to and including dependency parsing and generation of sentence trees in a custom XML-format. It also does some preprocessing not relevant for this thesis.

[2]Introduced after personal correspondence with my advisor Katarina Heimann Mühlenbock.
[3]Ibid
[4]I have been allowed to use it as my advisor, Katarina Heimann Mühlenbock, work at Språkbanken.

4.3 Classification algorithms in Weka

For the Classification task the Waikato Environment for Knowledge Analysis, or Weka, is used. Weka is a suite of machine learning and data mining tools and algorithms implemented in Java [Hall et al., 2009]. Weka has been developed at Waikato University, New Zealand, since 1993 and is distributed under the GNU General Public Licence. Weka consists of two main parts, a suite of GUI applications for different applications machine learning and data mining tasks, and a library of Java classes for including Weka directly into Java applications. Through this API Weka provides access to classes implementing a large number of popular machine learning schemes and algorithms as well as filters and other preprocessing tools. Only the Java API has been used in this thesis.

As the time frame for this study is limited only some cursory research into the different available algorithms and which of them might be most suited has been done. For this reason the fact that it is easy to swap classifier in Weka, as they all share a common interface, is utilized. Four different schemes has been selected and tested. All schemes below can handle missing values, something that is necessary as the data extraction at a few points might generate infinity or NaN (not a number, a value generated by Java when the result of a computation is not considered an actual number, such as arithmetically undefined values such as 0 divided by 0 or the square root of a negative number) due to bad grammar in the corpus data. As bad grammar is unavoidable in real applications these bad examples are not filtered from the training and test sets.

4.3.1 J48 - Decision Tree

There are a number of different tree learning algorithms in Weka. The one used in this research, the J48 decision tree learner, is a Java implementation of Quinlan's C4.5 tree learning algorithm which is one of the most popular algorithms today.

The C4.5 algorithm generates decision trees by finding the attribute with the highest information gain, creating a decision point for it and splitting the training data based on that attribute and then recursively doing the same thing for the resulting sets [Quinlan, 1993]. The leaf nodes of the resulting tree represents classifications.

4.3.2 ClassificationViaRegression - Regression

Regression is, as mentioned above, not mainly a classification scheme but rather an analytic tool to find relations among variables. However, Weka contains a number of so-called meta-learning tools where classifiers are combined with other tools to create more powerful schemes. The method used is a combination of decision trees and linear regression called Model Trees, realised in the class ClassificationViaRegression. By placing linear regression functions on the leaf nodes of a traditional decision tree, a very powerful classifier can be constructed [Frank et al., 1998].

4.3.3 NaiveBayes - Naive Bayes

Naive Bayes is a relatively simple probabilistic algorithm. The naiveté in the name comes from the Naive Bayes algorithms implicit assumption that all variables are independent. In the case of readability assessment it is obvious that some variables are, in fact, conditionally dependent on each other, for instance the two word length metrics (number of characters and number of syllables). However, the Naive Bayes algorithm has been shown to provide a good approximation for most applications [Witten et al., 2011].

Naive Bayes works by training a probability distribution for each feature. When a new sentence is classified the probabilities for each of its feature values is calculated, and the average result is used as a basis for classification.

4.3.4 SMO - Support Vector Machine

Support Vector Machines (SVM) is an algebraic approach to machine learning. Objects with known class is represented as points in a n-dimensional space, where n is the number of features. The algorithm then attempts to find a maximum margin hyperplane separating the objects by their class [Witten et al., 2011]. New objects are classified by calculating on which side of this hyperplane the object's corresponding point occurs. Support Vector Machines has been increasingly popular in Computational Linguistics in recent years and a number of SVM implementations are available. While Weka has a wrapper for the popular LibSVM C/C++ module this thesis uses the SMO, Sequential Minimal Optimization, algorithm [Platt, 1998] which is a Java based SVM learner included in the standard Weka toolkit.

Chapter 5

Implementation

This chapter will explain the inner workings of the actual system implemented for this thesis.

5.1 Modularity

The system can roughly be divided into two different modules: the analysis module and the classification module. These have a single class in common, the FeatureVector class, but are otherwise decoupled. The majority of the work for this thesis cover the interface between the modules, especially on the analysis side. The analysis module can further be broken down into a preprocessor, the Korp corpus import tool, which does all linguistic analysis, and an XML-parser and feature vector generator which converts the result from this linguistic analysis to statistical data and organises this data into a FeatureVector object. The classification module can be broken down into the Model module, which controls which features to use and serves as a data wrapper for Weka, and the ClassifierFacade module which is just a functional wrapper for Weka.

In Figure 5.2 on page 35 the chain of data transformations through the system is visualised and in Figure 5.1 the general structure of the system is visualised.

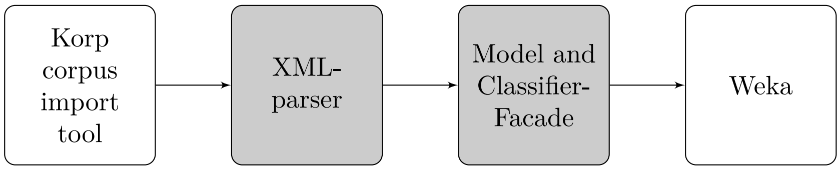

Figure 5.1 – Rough system structure, the grey modules are constructed for this thesis.

5.2 Text analysis

The analysis module extracts the data relevant for classification. This module is at this time dependent on third party preprocessing which is not optimized for speed but accuracy. As such this module will have to be redesigned before use in production. Like the system in general this module is not highly coupled and is split into two packages, the Preprocessor package and the XMLHandler package.

5.2.1 Preprocessing

As the preprocessing is performed by an external application (described below) the preprocessor portion of the implemented system will consist only of a class handling communication with that application. However, due to the infeasibility of using that application in a production environment and the limited time available, efficient wrapping will not be a high priority and might not be implemented as it does not affect the result of the primary study but only the ease-of-use of the test environment.

The preprocessor generates a lemma- and part-of-speech tagged as well as dependency parsed version of the text represented as an XML-document.

5.2.2 The Korp corpus import tool

The Korp corpus import tool is used at Språkbanken, University of Gothenburg, to generate their corpora. It performs a number of preprocessing steps of which lemmatisation, part-of-speech-tagging and dependency parsing are the ones relevant for this thesis. The result can be output in a number of formats, an XML representation is used in this thesis. See Appendix A for an example of the XML-format.

5.2.3 XML-parsing

This section describes the data extraction for the document based analysis. The sentence based analysis is done by treating each sentence as a one-sentence document.

Data extraction from XML is generally performed either by using a SAX-parser or a DOM-object representing the XML-document. A SAX-parser approach is used in this thesis.

SAX, or Simple API for XML, is an event-based sequential access parser API. A SAX-parser traverses a XML-document sequentially and calls a specific handler method for every "event", that is, when a certain element type, such as a start tag or an end tag, is reached.

The SAX parser in itself is stateless and unlike DOM, or Document Object Model, does not need to keep the document in memory. If the parsing can be constructed in such a way that a single linear traversal of the XML-code is sufficient, SAX should be both faster and cheaper, memory wise, than DOM.

While both SAX and DOM could be used to directly generate a feature vector by creating methods for extracting and calculating features one by one this is not very efficient. SAX would need to traverse the XML-code at least once, and sometimes more, for each feature, and DOM would need generation of a DOM object in which lookup could be done.

Instead a specially designed Document object holding data about the document is generated by the parser. This Document object (currentDocument) holds all information about the document relevant for calculating the feature vector but not the actual values for the feature vector. For example: instead of holding the average length of sentences currentDocument holds the total number of words and the total number of sentences. The calculations converting these absolute numbers to averages and ratios are done by the FeatureVector constructor. The parser also keeps a Sentence object (currentSentence) holding data about the current sentence which cannot directly be inserted into the Document object.

For each word the parser reads, currentDocument and currentSentence is updated. The total number of words and the number of occurrences of the word's part-of-speech and dependency type are incremented. The lemma is added to a map of lemmas and the number of occurrences for each lemma for the Document. If applicable, that is, if the word is not the root of its sentence, the total number of dependencies is incremented, and the dependency distance is added to the total dependency distance. Also, a WordNode object is generated and added to a bag of words in currentSentence. Lastly, the word is checked against SweVoc, see Section 4.2.2 on page 24.

When the end of a sentence is reached the sentence is analysed and currentDocument is updated with sentence specific data. The total number of sentences is incremented and the total sentence depth and the verb arities are calculated. Also, if the sentence has a verbal root the total number of verbal roots is incremented.

This process effectively creates a new representation of the document, however with relatively large loss of information as it only contains statistics about the document and not any meaningful part of its semantics.

5.2.4 Computation of features

The Document object created during the XML-parsing is used to construct a FeatureVector object. The FeatureVector class represents yet another representation of the document where the raw numbers extracted from the text are converted to a number of averages and ratios. The raw numbers could have been used directly, but that could have made the system dependent on document length, which is not desirable.

The FeatureVector object is designed for experimental use and as such all features are included. In a production version of the system, a class hierarchy of FeatureVector classes for the different models could be designed to speed up the process for smaller models.

As FeatureVector is a custom class its objects can not be used directly with Weka. Weka instead use the class Instance and its specialised container class Instances to handle data points and data sets. The system thus includes a number of custom model classes which serves as an interface between the system's representation of data and Weka's. Each model class corresponds to one of the models introduced in Section 6.2 on page 37 and thus contains only a subset of all features in the FeatureVector object. The model objects contain methods for creating Instance and Instances objects and are used throughout the system whenever the structure of the data needs to be referred to.

Just as Weka uses Instances to handle data sets the system uses a custom TestSet class which contains a data set of FeatureVector objects. This is, among other things, used for the custom n-fold evaluations, where the training and testing sets need to be regenerated from FeatureVector objects a number of times.

5.3 Classification

Training and classification is done through the ClassifierFacade class. If the aforementioned model classes can be said to wrap Weka's data types, ClassifierFacade wraps its functionality. This class is instantiated with a Classifier object from Weka, for instance J48, a FeatureVector data set (represented by a Java Vector with FeatureVector objects), and a model object. The constructor stores these three essential components of the system and also uses the FeatureVectors to train the Classifier. The ClassifierFacade object is called whenever a data point is to be classified. There are methods both for hard and soft classification.

For evaluation there is a special static Evaluator class. This class has methods both for singular test runs as well as n-fold evaluations for both hard and soft classification. For the n-fold evaluations a new Classifier and ClassifierFacade is created for each run to make sure that the test runs are independent of one another. The Evaluator class works with a special TestResults class which handles both singular test runs as well as n-fold evaluations. TestResults keep track of both true positives and false positives for both possible classes in order to calculate precision and recall for each set.

The ClassifierFacade works as an interface to Weka, however, other parts of the system, such as XML-parsing and FeatureVector generation, are still explicitly used. A comprehensive API for the full system, as well as pre-trained classifiers should be included in a production version of the system.

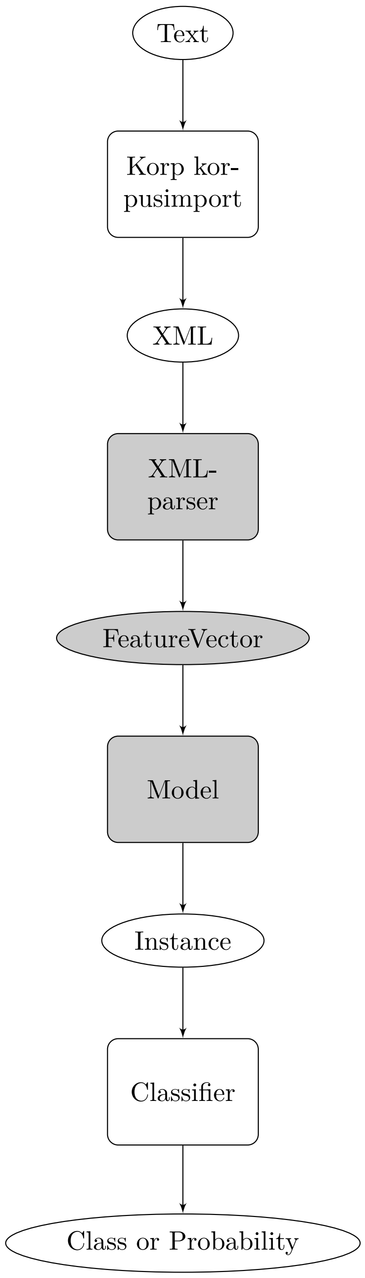

Figure 5.2 – The import and classification chain, the grey steps are constructed for this thesis.

Chapter 6

Results

This chapter will present the results from the evaluations as well as some analysis and discussion of those results.

6.1 Overview

This chapter will give an account of the results from testing the system with different models and the four algorithms described in Section 4.3 on page 29. Hard classification is evaluated on both sentence and document level. When it comes to ranking only documents are relevant, soft classification will therefore only be evaluated on a document level, and only with the best performing models from the hard classification evaluation. The hard classification can be considered a test run to find the most suitable models for soft classification as well as an evaluation of the new and the established approaches to degree of readability assessment.

6.2 Models

A number of different *models*, sets of features used for classification, has been used to evaluate the system. Firstly, the three aforementioned established metrics for Swedish, LIX, OVIX and Nominal ratio, have one single feature model each. Also, a model combining them as a three feature model is used to test whether the new more involved linguistic analysis captures something outside the scope of the established metrics.

When it comes to the new linguistic analysis models, four different models have been constructed. The first model contains only shallow text features and is referred to as the Shallow model (3 features). The second model contains shallow text and lexical features and is referred to as the Lexical model (7 features). The third model contains shallow text, lexical and morpho-syntactic features and is referred to as the Morpho-Syntactic model (34 features). The fourth model contains shallow text, lexical, morpho-syntactic and syntactic features and is referred to as the Syntactic model (116 features). The rule is that the models grow incrementally, that is, each model includes all features from the less complex models. This approach has been chosen to illustrate to what extent each level of linguistic analysis increase the accuracy of the system.

Also, two additional models were included. The Super model, no pun intended, is a combination of the Syntactic model and all three established Swedish metrics, that is, every feature extracted by the system. The NoDep model (No Dependency) is a model combining the Morpho-Syntactic model with all three established Swedish metrics, that is, every feature extracted by the system except those requiring dependency parsing.

The term *configuration* will refer to a combination of an algorithm and a model. For instance, the SMO algorithm combined with the Super model.

The term *feature set* will refer to a set of features strictly resulting from a certain level of linguistic analysis. For instance, the Lexical model consist of the shallow and the lexical feature sets. While model names are capitalized feature sets will be written in lower case letters only.

6.3 Results for hard sentence evaluation

The four different algorithms were evaluated with a data set of 7000 randomly selected sentences. Half of the sentences were selected from the Easy-to-read Swedish corpus LäSBarT (translates as "Readable"). The other half were selected from the news text corpus GP2006, which contain all articles from the Swedish newspaper Göteborgsposten from 2006. It should be noted that these corpora are not sentence based but document based, which means that a small number of relatively complex sentences might appear among the easy-to-read sentences and vice versa.

The tests have been performed with 7-fold cross-validation to smooth the result and hopefully eliminate statistical anomalies arising in individual test runs. That is, the training data has been split into 7 parts, each of which has been used once to test a classifier trained on the other 6 parts. The average results follow below.

Precision and recall are calculated individually for LäSBarT and GP2006. Two of the algorithm names are abbreviated in the tables, these abbreviations are CvR for ClassificationViaRegression and NB for NaiveBayes. The algorithm and corresponding results with the highest accuracy is highlighted. The accuracy of a configuration is total percentage of correctly classified data points for both corpora. The numbers represent percentages which have been rounded off to one decimal.

6.3.1 LIX

To read about the LIX metric, see Section 2.3.1 on page 12. The results of using LIX for classification can be viewed in Table 6.1.

		LäSBarT		GP2006	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	**61.2**	**61.0**	**63.1**	**61.6**	**59.3**
CvR	60.7	60.0	64.5	61.6	56.9
NB	59.7	57.0	79.1	65.8	40.2
SMO	57.3	56.5	63.7	58.4	50.9

Table 6.1 – LIX for hard sentence classification.

The difference between the best and the worst performing algorithm is 3.9 percentage points of accuracy and the average accuracy is approximately 59.7 %.

As LIX is the most dominant metric in use in Swedish today this result is less than could have been expected. However, as LIX is not designed for sentences and some normal difficulty texts might contain easy-to-read-sentences, and vice versa, the result is understandable.

6.3.2 OVIX

As the logarithmic version of OVIX, presented in Figure 2.4 on page 13, is very error prone when applied to sentences, a simplified version is used. It is defined as

$$OVIX = \frac{n(uw)}{n(w)}$$

Figure 6.1 – The simplified OVIX formula.

where n(w) is the number of words and n(uw) is the number of unique words. To read more about the OVIX metric, see Section 2.3.2 on page 12. The results of using OVIX for classification can be viewed in Table 6.2.

		LäSBarT		GP2006	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	50.4	50.2	99.9	93.8	0.9
CvR	50.5	50.3	84.7	51.4	16.3
NB	50.1	50.1	84.1	50.4	16.2
SMO	**51.0**	**51.1**	**45.8**	**50.9**	**56.3**

Table 6.2 – OVIX for hard sentence classification.

The difference between the best and the worst performing algorithm is 0.9 percentage points of accuracy and the average accuracy is approximately 50.5 %.

The OVIX-metric seems to be a very weak metric for sentence assessment as it barely outperforms pure chance. However, this is perhaps to be expected when it is applied at the sentence level. The high recalls of LäSBarT and low recalls of GP2006 implies that generally the algorithms tend to classify a majority of sentences as easy-to-read. The exception is the SMO algorithm which seems to have a slight opposite tendency, this is also the best performing algorithm, although not significantly so.

6.3.3 Nominal ratio

To read about the Nominal ratio metric, see Section 2.3.3 on page 13. The results of using Nominal ratio for classification can be viewed in Table 6.3.

		LäSBarT		GP2006	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	54.7	53.4	73.7	57.6	35.8
CvR	**62.5**	**60.7**	**70.8**	**65.0**	**54.2**
NB	52.7	51.6	86.6	58.4	18.8
SMO	52.5	51.4	88.6	58.9	16.3

Table 6.3 – Nominal ratio for hard sentence classification.

The difference between the best and the worst performing algorithm is 10.0 percentage points of accuracy and the average accuracy is approximately 55.6 %.

Considering sentence level assessment, Nominal ratio seems to perform best among the established metrics when it comes to best algorithm accuracy. If the average is considered though, LIX perform slightly better. However, the same tendency for over-classifying sentences as easy-to-read noted for OVIX seems to exist for Nominal ratio as well. Again, though, the best performing algorithm does not seem to fit this pattern, however, the opposite is not a large problem either.

6.3.4 Combination of established metrics

This model combines the three established metrics LIX, OVIX and Nominal ratio, read more about these in Section 2.3 on page 11. The result of using the combination of established metrics for classification can be viewed in Table 6.4.

		LäSBarT		GP2006	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	61.1	59.5	69.6	63.3	52.6
CvR	**65.2**	**63.6**	**70.9**	**67.1**	**59.5**
NB	60.4	57.4	80.1	67.1	40.7
SMO	56.9	54.7	79.5	62.5	34.2

Table 6.4 – Combination of established metrics for hard sentence classification.

The difference between the best and the worst performing algorithm is 8.3 percentage points of accuracy and the average accuracy is approximately 60.9 %.

The combination is slightly better than Nominal ratio, which is the best performing among the established metrics when best algorithm accuracy is considered. The combination also outperforms LIX when the average is considered. NaiveBayes and SMO seem to have the same tendency to over-classify as easy-to-read as seen above, while ClassificationViaRegression and J48 are more balanced. The fact that ClassificationViaRegression is the best performing algorithm again might be explained by the fact that it was the algorithm in the best performing configuration among the one feature models. The two extra features might be considered an augmentation of this configuration.

6.3.5 The Shallow model

See Section 4.2.1 on page 24 for more information about the shallow feature set. The result of using the Shallow model for classification can be viewed in Table 6.5.

		LäSBarT		GP2006	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	63.1	60.8	73.7	66.6	52.6
CvR	**63.4**	**60.2**	**78.6**	**69.2**	**48.1**
NB	59.9	56.5	86.0	70.7	33.9
SMO	59.8	56.6	83.4	68.5	36.1

Table 6.5 – The Shallow model for hard sentence classification.

The difference between the best and the worst performing algorithm is 3.6 percentage points of accuracy and the average accuracy is approximately 61.6 %.

Like with the established metrics there is a tendency to over-classify data points as easy-to-read. The Shallow model analyses the same properties as the LIX metrics but it seems that the features perform slightly better in raw form than when they are mathematically compounded by the LIX formula. Interestingly, the average result is better than for the combination of established metrics, however, best algorithm accuracy is not better than for the combination.

6.3.6 The Lexical model

The Lexical model includes all features from the Shallow model as well as the features in the lexical feature set. See section Section 4.2.2 on page 24 for more information about the lexical feature set. The result of using the Lexical model for classification can be viewed in Table 6.6.

		LäSBarT		GP2006	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	68.3	65.0	79.1	73.4	57.5
CvR	**68.3**	**65.5**	**77.6**	**72.5**	**59.1**
NB	61.8	57.8	87.5	74.3	36.0
SMO	66.7	63.1	79.9	72.7	53.4

Table 6.6 – The Lexical model for hard sentence classification.

The difference between the best and the worst performing algorithm is 6.5 percentage points of accuracy and the average accuracy is approximately 66.3 %.

Note that ClassificationViaRegression was slightly better than J48 before rounding off.

As expected the addition of lexical features increase the accuracy of all algorithms. The ClassificationViaRegression algorithm performs best again which, as for the combination of established metrics, might be viewed as an augmentation of the prior best performing configuration.

6.3.7 The Morpho-Syntactic model

The Morpho-Syntactic model includes all features from the Lexical model as well as the features in the morpho-syntactic feature set. See section Section 4.2.3 on page 26 for more information about the morpho-syntactic feature set. The result of using the Morpho-Syntactic model for classification can be viewed in Table 6.7.

		LäSBarT		GP2006	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	77.4	75.8	80.4	79.1	74.4
CvR	**79.0**	**76.4**	**83.8**	**82.1**	**74.1**
NB	71.8	65.4	92.5	87.3	51.1
SMO	78.2	75.2	84.1	82.0	72.3

Table 6.7 – The Morpho-Syntactic model for hard sentence classification.

The difference between the best and the worst performing algorithm is 7.2 percentage points of accuracy and the average accuracy is approximately 76.6 %.

Again the ClassificationViaRegression is the best performing algorithm.

6.3.8 The Syntactic model

The Syntactic model includes all features from the Morpho-Syntactic model as well as the features from the extensive syntactic feature set. See section Section 4.2.4 on page 26 for more information about the syntactic feature set. The result of using the Syntactic model for classification can be viewed in Table 6.8.

		LäSBarT		GP2006	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	78.3	77.0	80.8	79.8	75.9
CvR	82.1	79.2	87.1	85.7	77.1
NB	73.7	67.1	93.1	88.7	54.3
SMO	**82.7**	**80.0**	**87.3**	**86.0**	**78.1**

Table 6.8 – The Syntactic model for hard sentence classification.

The difference between the best and the worst performing algorithm is 9.0 percentage points of accuracy and the average accuracy is approximately 79.2 %.

This full new model performs considerably better than the established metrics. There is still a small tendency for over-classifying sentences as easy-to-read, especially for NaiveBayes. However, it is small enough that the assumption can be made that the relatively low accuracy for the established metrics is caused to a large extent by the models themselves. SMO is now the best performing model, which is not consistent with the earlier apparent augmentation behaviour, however, as SMO is the algorithm most increasing it's accuracy by addition of morpho-syntactic features, SMO might actually be the most fitting algorithm for the syntactic and morpho-syntactic feature sets.

6.3.9 The Super model

The Super model consists of all features from all models above. That is, the Syntactic model with LIX, OVIX and Nominal ratio added. The result of using the Super model for classification can be viewed in Table 6.9.

		LäSBarT		GP2006	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	77.6	76.6	79.5	78.7	75.7
CvR	82.4	79.6	87.0	85.7	77.7
NB	73.7	67.1	92.9	88.4	54.5
SMO	**83.0**	**80.1**	**87.7**	**86.4**	**78.2**

Table 6.9 – The Super model for hard sentence classification.

The difference between the best and the worst performing algorithm is 9.3 percentage points of accuracy and the average accuracy is approximately 79.2 %.

The Super model slightly improves best algorithm accuracy, however, the average result does not improve as J48 slightly decrease in accuracy. The increase in best algorithm accuracy is however enough to declare the Super model as the best model for sentence level readability assessment.

6.3.10 The NoDep model

The NoDep model consists of all features from all models above except those from the syntactic feature set, which require dependency parsing. That is, the Morpho-Syntactic model with LIX, OVIX and Nominal ratio added. The result of using the NoDep model for classification can be viewed in Table 6.10.

		LäSBarT		GP2006	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	77.6	76.6	79.4	78.7	75.8
CvR	**79.4**	**77.2**	**73.7**	**82.2**	**75.2**
NB	72.1	65.7	92.1	86.9	52.0
SMO	78.7	76.1	83.8	82.0	73.6

Table 6.10 – The NoDep model for hard sentence classification.

The difference between the best and the worst performing algorithm is 6.6 percentage points of accuracy and the average accuracy is approximately 77.0 %.

The NoDep model relates to the Morpho-Syntactic model in the same way as the Super model relates to the Syntactic model. The addition of the three established metrics only slightly increase accuracy but the increase is big enough to declare the NoDep model as the best model for sentence level readability assessment not requiring dependency parsing.

6.3.11 Comments on the sentence level evaluation

As expected, different algorithms seem to be suitable for different models. This is one of the reasons more than one algorithm was evaluated and why the best algorithm accuracy was used together with the average instead of comparing each algorithm individually. Some models seem to be more sensitive to choice of algorithm than others. See Figure 6.3 on page 54 for a comparison of the differences in algorithm accuracy for each model and a comparison between the sentence level and the document level. See also Section 6.4.11 on page 53 for a discussion of these differences.

The fact that many configurations seem to favour easy-to-read classifications might be explained somewhat by the higher probability of an easy-to-read sentence in a normal document than vice versa. Realistically a news text should be more heterogeneous when it comes to sentence readability, since it can be assumed that no great effort has been put into purging easy-to-read sentences which appear naturally in the writing process. This implies a possible slight overlap in degree of readability of sentences between the two corpora, however, this should not be a problem in the document case due to a natural smoothing effect when full documents are considered.

Interestingly, ClassificationViaRegression is the best performing algorithm for 6 out of 10 models, however, SMO is the algorithm in the best performing configura-

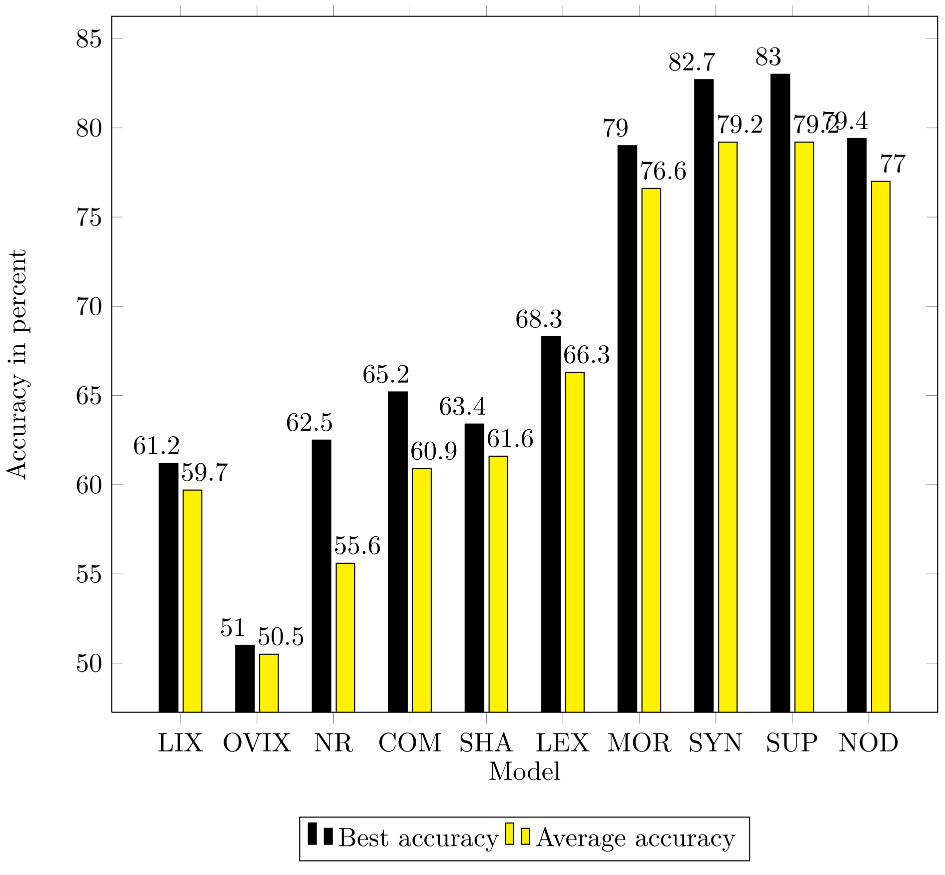

Figure 6.2 – The accuracies for the different models used in the sentence level evaluation.

tion. This might be explained by SMO being most fitting for the Morpho-Syntactic and Syntactic features. J48 is the best algorithm only in one case and NaiveBayes is never the best algorithm.

The accuracy increase between the Lexical and Morpho-Syntactic models is the largest one in the sentence level evaluation both when it comes to best algorithm accuracy and average accuracy. This implies that morpho-syntactic features are the most relevant features when it comes to sentence level assessment. Feature selection shows that 11 of 13 features selected from the Morpho-Syntactic model and 9 of 14 selected from the Syntactic model belong to the morpho-syntactic feature set.

A chart showing all best algorithm and average accuracies can be found in Figure 6.2 on page 46.

6.4 Results for hard document classification

The same 7-fold cross validation method was used for the document level evaluation as for the sentence level evaluation. Instead of GP2006 the slightly newer GP2007 was used as a source for news texts. The main difference between the evaluations is the size of the data set. Whereas the sentence level evaluation used a data set of 7000 data points, the document level evaluation use only 1400 data points. However, as each data point represent a document instead of a sentence the actual amount of text processed to generate the data points is in fact larger.

6.4.1 LIX

To read about the LIX metric, see Section 2.3.1 on page 12. The results of using LIX for classification can be viewed in Table 6.11.

		LäSBarT		GP2007	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	**79.0**	**74.8**	**87.7**	**85.1**	**70.4**
CvR	78.6	72.5	92.0	89.1	65.1
NB	77.1	70.8	92.3	88.9	61.9
SMO	77.4	81.3	71.0	74.3	83.7

Table 6.11 – LIX for hard document classification.

The difference between the best and the worst performing algorithms is 1.9 percentage points of accuracy and the average accuracy is approximately 78.0 %.

Compared to the sentence level evaluation this is a considerably better result for the LIX metric. But, as LIX is designed for document assessment rather than sentence assessment this is not surprising. There is still a tendency to over-classify news texts as easy-to-read. As in the sentence level evaluation J48 seem to be the most suitable algorithm for use with LIX.

6.4.2 OVIX

For the document level evaluation the logarithmic version of OVIX, presented in Figure 2.4 on page 13, has been used. To read about the OVIX metric, see Section 2.3.2 on page 12. The results of using OVIX for classification can be viewed in Table 6.12.

		LäSBarT		GP2007	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	83.6	83.9	83.1	83.3	84.0
CvR	**84.6**	**83.8**	**85.9**	**85.5**	**83.4**
NB	84.1	81.3	88.7	87.6	79.6
SMO	84.5	82.0	88.4	87.4	80.6

Table 6.12 – OVIX for hard document classification.

The difference between the best and the worst performing algorithms is 1.0 percentage points of accuracy and the average accuracy is approximately 84.2 %.

OVIX was the worst performing metric when it came to sentence assessment barely outperforming pure chance. When it comes to document assessment the OVIX metric performs significantly better. As with the results for LIX, the improvement as such is not very surprising.

6.4.3 Nominal ratio

To read about the Nominal ratio metric, see Section 2.3.3 on page 13. The results of using Nominal ratio for classification can be viewed in Table 6.13.

		LäSBarT		GP2007	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	72.0	71.4	73.3	72.6	70.7
CvR	**73.1**	**71.2**	**77.4**	**75.3**	**68.7**
NB	61.1	74.8	33.6	57.2	88.7
SMO	53.0	51.6	98.9	86.2	7.1

Table 6.13 – Nominal ratio for hard document classification.

The difference between the best and the worst performing algorithms is 20.1 percentage points of accuracy and the average accuracy is approximately 64.8 %.

Both LIX and OVIX experienced significant accuracy increases between the sentence level and document level analyses for their respective top results. Nominal ratio on the other hand only experience an improvement of about 8 percentage points. Nominal ratio thus goes from being the best performing established metric in the sentence level evaluation to the worst performing established metric for documents.

The especially bad results for NaiveBayes and SMO should also be noted. Apparently NaiveBayes has a tendency to underestimate the degree of readability while SMO has the opposite tendency. Exactly why this happens is hard to explain due to the black box nature of Weka's inner workings.

As with LIX the same algorithm perform best for both sentence assessment and for document assessment.

6.4.4 Combination of established metrics

This model combines the three established metrics LIX, OVIX and Nominal ratio, read more about these in Section 2.3 on page 11. The result of using the combination of established metrics for classification can be viewed in Table 6.14.

		LäSBarT		GP2007	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	**90.6**	**91.7**	**89.4**	**89.7**	**91.9**
CvR	90.1	90.5	89.6	89.7	90.6
NB	88.1	85.9	91.1	90.6	85.0
SMO	89.3	87.6	91.6	91.2	87.0

Table 6.14 – Combination of established metrics for hard document classification.

The difference between the best and the worst performing algorithms is 2.5 percentage points of accuracy and the average accuracy is approximately 89.5 %.

As in the sentence level evaluation the combination of the three established metrics do perform slightly better than any of them on their own. The best single value configuration was OVIX and ClassificationViaRegression, however, this is not the configuration with the highest accuracy using the combination model. This is especially interesting as ClassificationViaRegression was the best performing algorithm for both OVIX and Nominal ratio, only LIX performed best with the J48 algorithm.

6.4.5 The Shallow model

See Section 4.2.1 on page 24 for more information about the shallow feature set. The result of using the Shallow model for classification can be viewed in Table 6.15.

		LäSBarT		GP2007	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	84.4	84.1	84.9	84.7	84.0
CvR	**87.9**	**86.6**	**89.7**	**89.3**	**86.1**
NB	77.4	71.2	92.1	88.9	62.7
SMO	72.6	76.7	64.9	69.6	80.3

Table 6.15 – The Shallow model for hard document classification.

The difference between the best and the worst performing algorithms is 15.3 percentage points of accuracy and the average accuracy is approximately 80.6 %.

The features used in the Shallow model are more or less the same features as those used to calculate LIX. However, the mathematical mangling performed by the LIX formula apparently misses something relevant, perhaps the relation between word length and sentence length. This model actually outperform each one of the established metrics used on their own and only a combination of them manages to outperform a simple model such as this one. However, considering the average accuracy of the algorithms, OVIX performs slightly better.

6.4.6 The Lexical model

The Lexical model includes all features from the Shallow model as well as the features in the lexical feature set. See section Section 4.2.2 on page 24 for more information about the lexical feature set. The result of using the Lexical model for classification can be viewed in Table 6.16.

		LäSBarT		GP2007	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	89.1	88.1	90.3	90.0	87.9
CvR	**92.1**	**90.3**	**94.3**	**94.0**	**89.9**
NB	85.4	79.2	96.1	95.1	74.7
SMO	81.4	81.9	80.7	81.0	82.1

Table 6.16 – The Lexical model for hard document classification.

The difference between the best and the worst performing algorithms is 10.7 percentage points of accuracy and the average accuracy is approximately 87.0 %.

As expected the Lexical model performs better than the Shallow model. The Lexical model also manages to outperform the combination of established metrics when considering best algorithm accuracy. Considering average accuracy the combination of established metrics still performs better though.

6.4.7 The Morpho-Syntactic model

The Morpho-Syntactic model includes all features from the Lexical model as well as the features in the morpho-syntactic feature set. See section Section 4.2.3 on page 26 for more information about the morpho-syntactic feature set. The result of using the Morpho-Syntactic model for classification can be viewed in Table 6.17.

		LäSBarT		GP2007	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	94.0	93.4	94.7	94.6	93.3
CvR	94.7	94.1	95.4	95.3	94.0
NB	90.0	86.0	95.6	95.0	84.4
SMO	**95.6**	**94.8**	**96.6**	**96.5**	**94.7**

Table 6.17 – The Morpho-Syntactic model for hard document classification.

The difference between the best and the worst performing algorithms is 5.6 percentage points of accuracy and the average accuracy is approximately 93.6 %.

In the case of the Morpho-Syntactic model the addition of features have pushed another algorithm to the top. This could mean that the morpho-syntactic feature set work better with SMO, which the sentence level results also hinted at. If this is the case, the new features might increase the accuracy of SMO to such a degree that SMO surpasses ClassificationViaRegression. A feature selection on the data set does imply that the features in the morpho-syntactic feature set are the ones most indicative of degree of readability as 12 of 15 features selected from the Morpho-Syntactic model are from the morpho-syntactic feature set. As this is the case the other feature sets can be viewed as augmentations of this set keeping SMO as the most suitable algorithm.

This model is the best performing so far outperforming all earlier models both when it comes to average and best algorithm accuracy.

6.4.8 The Syntactic model

The Syntactic model includes all features from the Morpho-Syntactic model as well as the features from the extensive syntactic feature set. See section Section 4.2.4 on page 26 for more information about the syntactic feature set. The result of using the Syntactic model for classification can be viewed in Table 6.18.

		LäSBarT		GP2007	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	93.6	93.7	93.4	93.4	93.7
CvR	93.4	94.3	94.4	94.4	94.3
NB	90.4	86.4	95.9	95.3	84.9
SMO	**97.1**	**97.3**	**96.9**	**96.9**	**97.3**

Table 6.18 – The Syntactic model for hard document classification.

The difference between the best and the worst performing algorithms is 6.7 percentage points of accuracy and the average accuracy is approximately 93.6 %.

As with other relations between feature sets the fact that SMO performs best again might result from the syntactic feature set augmenting the morpho-syntactic set.

Interestingly, J48 and ClassificationViaRegression actually has lower accuracy for the Syntactic model than for the Morpho-Syntactic model. This is however not totally surprising as similar results have been reached before, however for the morphologically more complex Italian language [Dell'Orletta et al., 2011].

6.4.9 The Super model

The Super model consists of all features from all models above. The result of using the Super model for classification can be viewed in Table 6.19.

		LäSBarT		GP2007	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	94.1	92.6	96.0	95.8	92.3
CvR	95.4	95.3	95.4	95.4	95.3
NB	90.8	87.0	95.9	95.4	85.7
SMO	**97.6**	**97.7**	**97.6**	**97.6**	**97.7**

Table 6.19 – The Super model for hard document classification.

The difference between the best and the worst performing algorithms is 6.8 percentage points of accuracy and the average accuracy is approximately 94.5 %.

The Super model manages to slightly outperform even the Syntactic model both in best algorithm accuracy and average accuracy. While the averages did not vary between the Syntactic and the Super model in the sentence level evaluation the difference here is larger than the difference in best algorithm accuracy. Both the best algorithm accuracy and the average accuracy here are the highest in the hard document evaluation.

6.4.10 The NoDep model

The NoDep model consists of all features from all models above except the those from the syntactic feature set, which require dependency parsing. The result of using the NoDep model for classification can be viewed in Table 6.20.

		LäSBarT		GP2007	
Algorithm	Accuracy	Precision	Recall	Precision	Recall
J48	94.2	93.8	94.9	94.8	93.7
CvR	95.1	94.5	95.9	95.8	94.4
NB	91.4	87.9	95.9	95.4	86.9
SMO	**97.0**	**95.7**	**98.4**	**98.4**	**95.6**

Table 6.20 – The NoDep model for hard document classification.

The difference between the best and the worst performing algorithms is 5.6 percentage points of accuracy and the average accuracy is approximately 94.4 %.

As expected the NoDep model is slightly less accurate than the Super model. Interestingly, the average accuracy of the NoDep model is higher than the average accuracy of the Syntactic model, and the best algorithm accuracy is only 0.1 percentage point lower.

6.4.11 Comments on the document level evaluation

As with the sentence level evaluation some models seem to be more sensitive to choice of algorithm than others, see Figure 6.3 on page 54 for a comparison of the difference in algorithm accuracy for each model and a comparison between the sentence level and the document level.

There does not seem to be any obvious patterns in the comparisons of best and worst algorithm accuracy, other than that Nominal ratio has the largest difference between best and worst performing algorithm in both levels of assessment. Interestingly, the level, sentence or document, on which the assessment is done does not in general seem to affect the difference in accuracy between best and worst algorithm accuracy. For example, for the model combining established metrics the difference is more than three times as large for the sentence level evaluation as for the document level, while for the Shallow model the difference is four times as large for the document level evaluation as for the sentence level.

The trend in the sentence level evaluation is that many configurations seem to favour easy-to-read classifications. In the document level evaluation this trend is not as prominent except for the Nominal ratio. This might be explained by the fact that the errors in sentence level assessment resulting from incidental easy-to-read sentences in otherwise standard news text should be eliminated by the data point smoothing resulting from a document level assessment. The overlap between easy-to-read texts and news texts seem instead to lead to a more general confusion where the error rates are more evenly distributed between the two data sets. However, as the total error rate is generally much smaller in the document level evaluation this is a sign of improvement.

As in the sentence level evaluation, NaiveBayes is never the best performing algorithm. ClassificationViaRegression perform best in 6 of 10 cases in the sentence level evaluation but not in the best performing configurations where SMO has the highest accuracy. In the document level evaluation ClassificationViaRe-

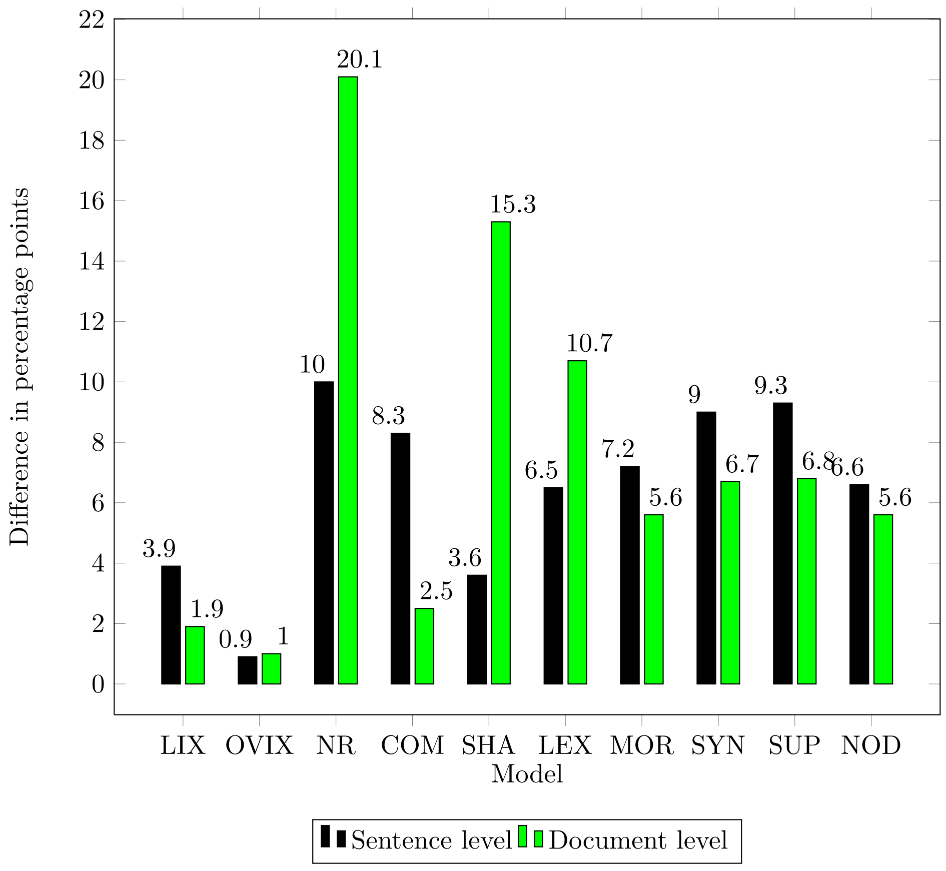

Figure 6.3 – Difference in algorithm performance in percentage points for each model.

gression only wins out in 4 of 10 cases putting it on a par with SMO in number of best configurations. However, SMO again wins out in the best performing configurations.

While OVIX was practically useless for sentence assessment it is the best performing among the established metrics when it comes to document assessment. This large difference is not surprising as the non-logarithmic OVIX is very sensitive to sample size. The logarithmic version on the other hand breaks down in a large number of cases when applied to sentences, as division by zero occurs when all words in a text unit are unique, something very common in sentences while very rare in full documents. These problems are both eliminated by applying the logarithmic OVIX to full documents.

The system is implemented in such a way that a document can be interpreted as the average of its constituent sentences. This results in a positive smoothing where a few non typical sentences, for instance, easy-to-read sentences in news text, have a relatively small impact on the total degree of readability of the document.

See Appendix B for some examples of erroneously classified documents. The documents are written in Swedish.

A chart showing all best algorithm and average accuracies can be found in Figure 6.4.

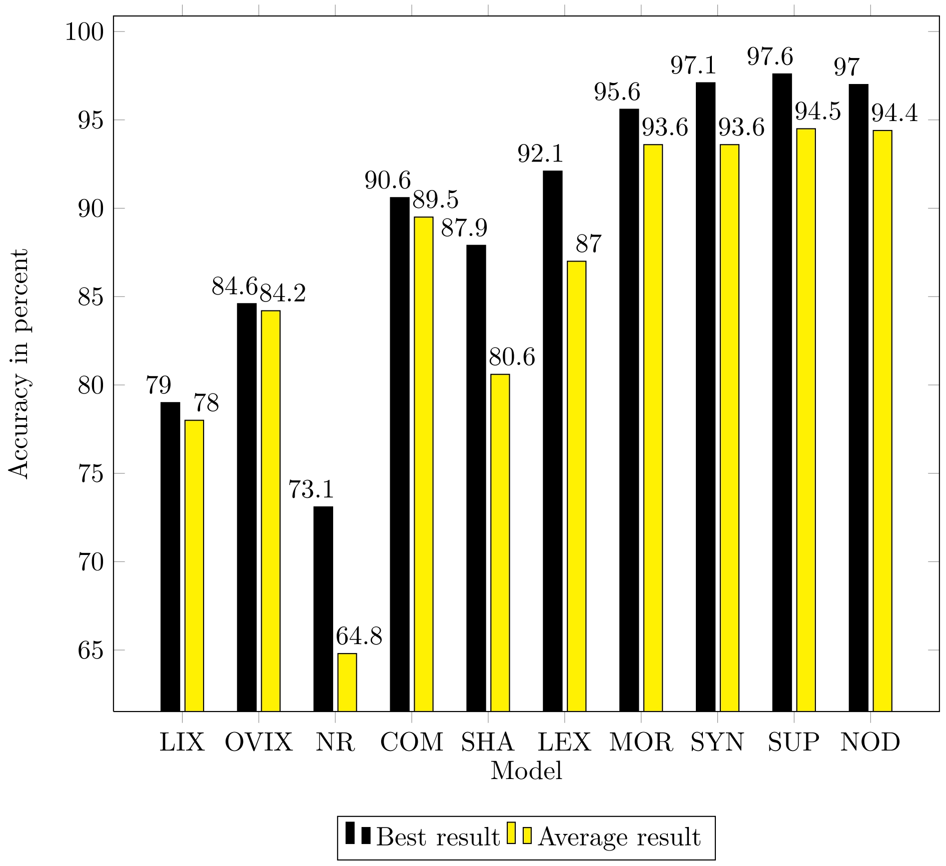

Figure 6.4 – The accuracies for the different models used in a document level evaluation.

6.5 Results for the soft document evaluation

The evaluation of soft document classification used the same data set as the hard document evaluation. Instead of counting precision and recall, the result is split into 4 parts, Correct, AmbCorr (ambiguous but correct), AmbErr (ambiguous but erroneous) and Erroneous. As the training data is hard classified, documents from GP2007 are considered to be 100 % "hard-to-read" and documents from LäSBarT are considered to be 0 % "hard-to-read". This is a bit of an oversimplification but a necessary one due to the available training data.

Any soft classification which is at most 25 percentage points off is considered to be Correct. That is, documents from LäSBarT classified to < 25 % or documents from GP2007 classified to > 75 % are considered to be correctly classified. Classifications between 25 and 50 percentage points off are considered to be AmbCorr, classifications between 50 and 75 are considered AmbErr and classifications with errors above 75 percentage points are considered Erroneous. There is also the sub-category perfect which is calculated as the ratio of perfect, 100 % correct, classifications. Any perfect classification is obviously also a correct classification.

Lastly, to evaluate the feasibility of using the probability of readability for ranking, the number of equivalence classes (# ECs in the tables), where two documents are considered equivalent if they have the same degree of readability, is also counted. This is counted as the total number for all 7 runs.

Based on the results from the evaluation of hard document classification and the fact that dependency parsing is infeasible for use in search engines today the two models to be evaluated with soft classification are the Super model and the NoDep model. The Super model as it was the best performing model overall, and the NoDep model as it was the best performing model that did not require dependency parsing.

The algorithms used below are modified versions of the earlier used algorithms. Without going into to much detail these modifications are the following: Laplace smoothing has been activated in J48, kernel estimation has been activated in NaiveBayes and logistic models has been added to SMO. The last modification is the only one strictly necessary for soft classification.

6.5.1 The Super model

The Super model consists of all features from all models above. The result of using the Super model for soft classification can be viewed in Table 6.21.

Algorithm	Correct	AmbCorr	AmbErr	Erroneous	Perfect	# ECs
J48	94.0	0.3	0.6	5.1	0.0	71
CvR	90.5	4.6	2.2	2.6	35.7	901
NB	92.9	1.0	1.1	5.1	29.4	964
SMO	**93.6**	**3.4**	**1.5**	**1.4**	**0.2**	**1398**

Table 6.21 – The Super model for soft document classification.

These results are more difficult to interpret than the results from the hard classifications but some conclusions can be drawn.

The algorithms J48 and NaiveBayes are generally sure of themselves giving either very high probabilities or very low probabilities even when the result is incorrect. This is demonstrated by the fact that most classifications are either within 25 percent margin or outside the 75 percent margin.

ClassificationViaRegression and SMO have a somewhat better curve where an unambiguous but erroneous classification is less common than an ambiguous but correct classification. ClassificationViaRegression instead has the problem that 35.7 % of the classifications are perfect. Perfect classifications can not be ordered into more than two unordered sets, perfectly "easy-to-read", and perfectly not "easy-to-read".

The reason that J48 seems to have no perfect classifications is that it runs with Laplace smoothing, with Laplace smoothing inactivated 39.4 % of the classifications are perfect. This means that even though no perfect classifications exist at least 39.4 % of the data points cannot be ordered. The fact that there are only 71 equivalence classes is also a considerable problem when it comes to ranking. See Figure 6.5 on page 59 for a comparison of the accuracies of all configurations used in the soft document evaluation.

SMO, however, have only 0.2 % perfect classifications as well as a high accuracy and no large tendency to be too sure of itself. With 1398 equivalence classes and 0.2 % perfect classifications it can be inferred that except for three perfect classifications all documents can be ordered. This implies that SMO is the most fitting algorithm for soft classification in this case. See Figure 6.6 on page 60 for a comparison of the number of equivalence classes in all configurations used in the soft document evaluation.

6.5.2 The NoDep model

The NoDep model consists of all features from all models except the those from the syntactic feature set, which require dependency parsing. The result of using the NoDep model for soft classification can be viewed in Table 6.22.

Algorithm	Correct	AmbCorr	AmbErr	Erroneous	Perfect	# ECs
J48	93.8	0.4	0.4	5.5	0.0	83
CvR	91.7	3.6	2.4	2.3	32.7	942
NB	93.4	0.3	0.4	5.9	10.1	1252
SMO	**95.8**	**1.9**	**1.1**	**1.3**	**0.2**	**1398**

Table 6.22 – The NoDep model for soft document classification.

The algorithms seem to have more or less the same behaviour for the NoDep model as for the Super model. Interestingly SMO actually perform better with the NoDep model than with the Super model with 97.7 % of documents classified within the 50 percentage point margin (Correct plus AmbCorr). This is actually a better result than the best result from the hard document evaluation. Also, SMO still has only 3 equivalent classifications. Again SMO is the most fitting algorithm. See Figure 6.5 on page 59 for a comparison of the accuracies of, and Figure 6.6 on page 60 for a comparison of the number of equivalence classes in, all configurations used in the soft document evaluation.

6.5.3 Comments on soft classification

The results above are indicative of a strong potential for using soft classification as a basis for ranking texts according to degree of readability. Especially SMO, and it might be assumed, SVM in general, given logistic models for soft classification, needs to be explored further.

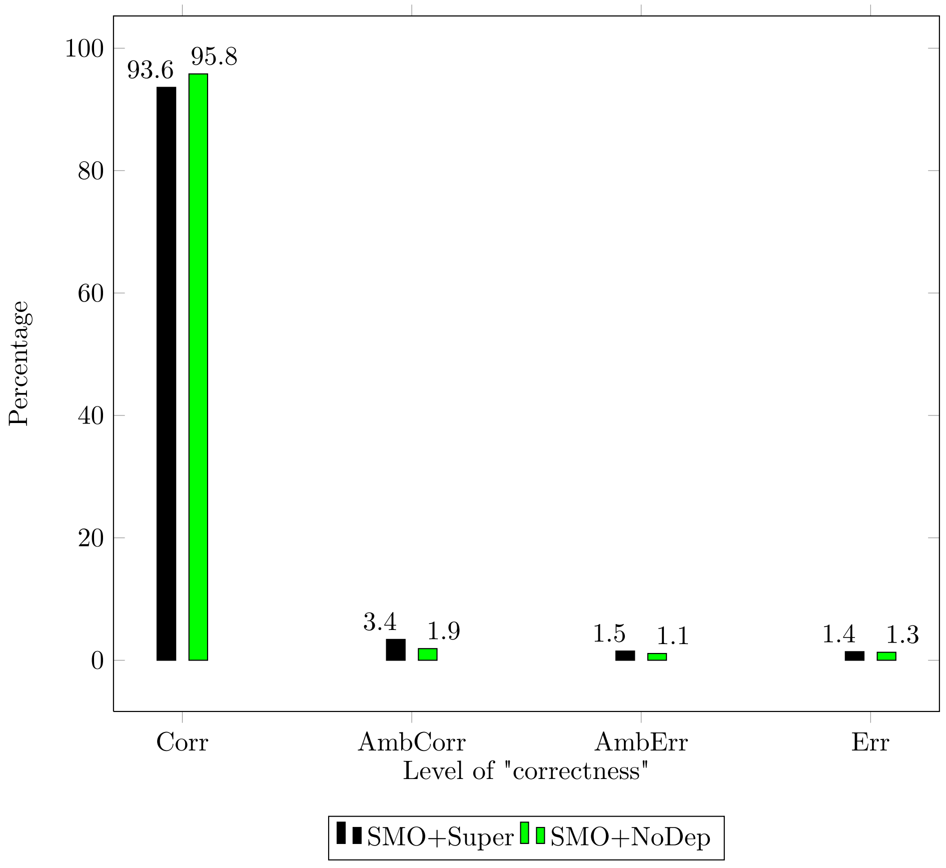

Figure 6.5 – The distribution of accuracies of each configuration in the soft document evaluation.

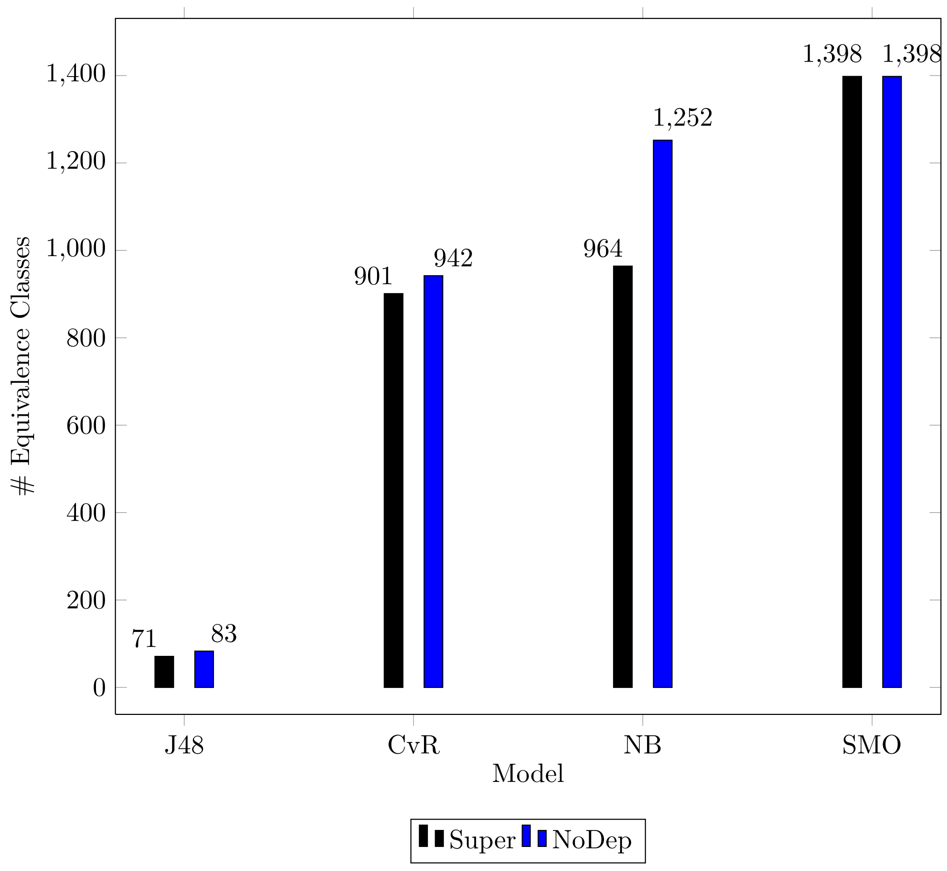

Figure 6.6 – The number of equivalence classes resulting from each configuration in the soft document evaluation.

6.6 Comparison to prior research

In general it is difficult to compare the results from the soft classification with much of the existing research on readability assessment. This is due to the fact that much of the publicised research has been done in America for American English. Unlike for Swedish there exist American corpora tagged with degree of readability (suitable grade in the U.S. education system). The existence of such corpora have resulted in much of the research being based on designing detectors (single class classifiers) for each such grade [Feng, 2010; Petersen, 2007; Heilman et al., 2008].

As far as the author of this thesis is aware, no comparable research into probability of easy-to-read as degree of readability has been done.

When it comes to hard classification of easy-to-read text READ-IT [Dell'Orletta et al., 2011] is the most relevant project for comparison. The features used are similar and so are the methods. The largest differences are the choice of languages and algorithms. While READ-IT works with Italian the system designed for this thesis works with Swedish. Also, while READ-IT only uses LibSVM as a classifier four different algorithms were used here. However, as SMO and LibSVM both are Support Vector Machines they are at least theoretically comparable.

6.6.1 Sentence classification

READ-IT's ability to classify sentences was evaluated with 7000 sentences. Half of the sentences were selected from the newspaper La Repubblica and half from the easy-to-read newspaper Due Parole. As with the evaluation in this thesis 1000 sentences, 50 % from each corpus, were used for testing. However, no cross-validation was done with READ-IT on sentence level [Dell'Orletta et al., 2011].

In Table 6.23 is a comparison between the accuracy of the system implemented in this thesis and READ-IT.

Model	READ-IT	This thesis (SMO)	This thesis (best)
Base/Shallow	59.6	59.8	63.4 (CvR)
Lexical	61.6	66.7	68.3 (CvR)
MorphoS/NoDep	76.1	78.7	79.4 (CvR)
Syntactic/Super	78.2	**83.0**	-

Table 6.23 – Comparison of sentence classification accuracy between READ-IT and this thesis.

6.6.2 Document classification

READ-IT's ability to classify documents was evaluated with 638 documents gathered evenly from the same sources as for the sentence analysis. A 5-fold cross-validation was done to smooth out the results [Dell'Orletta et al., 2011].

In Table 6.24 is a comparison between the accuracy of the system implemented in this thesis and READ-IT.

Model	READ-IT	This thesis (SMO)	This thesis (best)
Base/Shallow	76.65	72.6	87.9 (CvR)
Lexical	95.45	81.4	92.1 (CvR)
MorphoS/NoDep	**98.12**	97.0	-
Syntactic/Super	97.02	97.6	-

Table 6.24 – Comparison of document classification accuracy between READ-IT and this thesis.

6.7 Discussion and remaining problems

Based on the results above a number of conclusions can be drawn.

6.7.1 The data set

The data set is not perfectly suited for the experiments done in this thesis. A news text might very well be easy-to-read without the explicit intent of the author. The LäSBarT corpus also includes some easy-to-read but actual news texts. This means that there is probably a degree of readability overlap between the corpora and some similar texts might in reality be both news texts and easy-to-read. This is reflected in the result and might imply that a 100 % accuracy is practically impossible.

Another problem is the relatively small span of covered degrees of readability. One corpus is easy-to-read, the other is news texts. However, on-line there are a lot of texts with a lower degree of readability, that is, that are harder to read, than news texts, such as technical manuals, scientific articles, philosophical treatises, political editorials and so on. This system only cover the span between easy-to-read and news text, texts more difficult to read would probably all get a score of 100 (100 % assured news text), making them impossible to order among themselves. This could probably be resolved by using more extreme training data.

6.7.2 The models and feature relevance

The evaluations have shown that not all levels of linguistic analysis are equally relevant. For instance, there is only a small increase in accuracy when the the Syntactic feature set is added to the NoDep model. This implies that the dependency grammar features are not as relevant when it comes to easy-to-read classification as would have been expected.

However, a further analysis implies that this interpretation is overly simplistic. A feature selection on the Super model on the document level shows that 20 of 29 selected features belong to the syntactic feature set. This implies that the small increase of accuracy might rather result from a practical maximum performance based on the imperfection of the data set discussed above.

The feature selection selected only 29 of 119 features existing in the Super model. However, this does not mean that extracting and including the other 90 features is a waste of computation time and memory space. Firstly, the feature selected model does not perform as well as the full model, the selected features are just the *most* relevant features, not the *only* relevant features. Secondly, the results only show that these features are the most relevant features when it comes to distinguishing the general categories easy-to-read and news text. Distinguishing more general texts with high degree of readability from texts with low degree of readability for a certain reader type might call for a totally different subset of features. The same can be said for distinguishing texts of other genres with lower degrees of readability than either easy-to-read or news text, such as technical manuals or academic papers.

However, the results do imply that in this case an adequate result can be achieved without dependency parsing. This means that a significantly faster preprocessor, quite possibly fast enough to include in a search engine, can be built with existing tools.

6.7.3 Probability of classification and degree of readability

There is no way of knowing if any ranking returned by the system is actually a ranking based on degree of readability. These results only show that the soft classifier with fairly high accuracy can differentiate between two, internally relatively homogeneous, sets with different degrees of readability, and, using SMO, that almost all documents have slightly different probabilities of belonging to either set.

The system might, however, be a blunt tool and there is no way of knowing that a document, correctly classified as easy-to-read with 99 % certainty, in reality has a higher degree of readability than another document, correctly classified as easy-to-read with 98 % certainty. Intuitively there should be a correlation between the probability of easy-to-read classification and degree of readability, but with the data available for this thesis it is not possible to test this.

To evaluate this there would have to be a number of ordered sets of documents which could be experimentally ranked by the system. This ranking could then be compared to the correct order.

Chapter 7

Conclusion

This final chapter will summarize the thesis and the conclusions that can be drawn. Some future work will also be suggested.

7.1 Conclusions

There are a number of conclusions which can be drawn from the results in the previous chapter. The first is perhaps that for sentence level assessment existing metrics are lacking. For purposes such as text simplification, where assessment of the degree of readability of individual sentences is an essential component, the approach presented in this thesis might work significantly better. The new approach obviously captures a number of text features which affect degree of readability that traditional metrics misses.

While the existing metrics do seem to be a lot more relevant for document level assessment than for sentence level, their individual performances do not measure up to the new approach. Neither does a combination of traditional metrics.

However, as the performance of the NoDep model demonstrated it might not be necessary to analyse syntactic features of text to better assess the degree of readability. The loss of accuracy by excluding syntactic features is only about 1 percentage point. This means that more lightweight preprocessing is sufficient which makes web page ranking in a search engine, based on degree of readability, a lot more feasible.

Syntactic analysis is however not useless as shown by the feature selection mentioned in Section 6.7.2 on page 62. For comparative applications of machine generated, or manipulated, text, such as evaluation of text simplification systems it might be absolutely essential to analyse syntax. Syntax might also be a lot more relevant for some reader groups.

SMO, a version of the Support Vector Machine approach to machine-learning, was the best performing algorithm both when it came to hard classification of sentences and documents and when it came to soft classification of documents. The exception is the NoDep model for sentences which works slightly better with ClassificationViaRegression.

Using soft classification for ranking would probably be more practical if the training corpora consisted of very easy-to-read texts and very difficult to read texts. A problem with using easy-to-read versus news text corpora for both training and testing is that most soft classifications are clustered close to 100 % and 0 %. A more normal distribution would probably have been reached with more extreme training data and more diverse testing data.

7.2 Future work

A system with built in, and more lightweight, preprocessing should be implemented and tested. A phrase grammar parser and features based on a phrase grammar approach to syntax might also be relevant. The performance and execution speed of such a system is highly relevant for future use for web page ranking.

Feature selection from each of the feature sets could result in a relatively small set of very relevant features. These features could be used with regression to generate a formula for degree of readability scoring. This approach has the advantage that it is not bounded in the same way as soft classification which only covers the span on which it is trained.

Discourse level features of text such as measures of cohesion and coherence might also be relevant to degree of readability when it comes to documents and should be experimented with. For instance, the average cosine distance between pairs of consecutive sentences might be calculated using vector space models.

The soft classification approach presented in this thesis should be further tested by evaluating it with sets of readability ranked documents. However, no readability ranked or tagged corpora exist for Swedish at the time of writing. Automated ranking is an approach that is very relevant and should be explored as soon as readability ranked corpora exist.

New reader type specific corpora are also necessary for future research into making readability ranking more adaptive. Perhaps a calibrating set of texts can be developed which could be used to generate user specific classifiers. However, this requires a large amount of work with different reader groups.

The algorithms used in this thesis represent only a small part of all algorithms available for classification. Also, the classification approach most popular for these purposes, Support Vector Machine, have a lot of variables and training methods which can be tweaked to increase accuracy further. There are also other ways to further increase the accuracy of classifiers. One way is to use voting among a number of classifiers. Algorithm choice and optimization should therefore be further investigated.

A system trained with more extreme and less clustered training data should also be evaluated.

Bibliography

Sandra Alusio, Lucia Specia, Caroline Gasperin, and Carolina Scarton. Readability assessment for text simplification. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–9, 2010.

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

Carl Hugo Björnsson. *Läsbarhet*. Liber, Stockholm, 1968.

Jeanne S. Chall. *Readability: An appraisal of research and application.* Ohio State University Press, Columbus, OH, 1958.

Jeanne S. Chall and Edgar Dale. *Readability revisited: The new Dale–Chall readability formula*. Brookline Books, Cambride, MA, 1995.

Olivier Chapelle, Yi Chang, and Tie-Yan Liu. Future directions in learning to rank. In *JMLR: Workshop and Conference Proceedings*, volume 14, 2011.

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

Kevyn Collins-Thompson and Jamie Callan. A language modeling approach to predicting reading difficulty. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 2004.

Edgar Dale and Jeanne S. Chall. The concept of readability. *Elementary English*, 26(23), 1949.

Edgar Dale and Ralph W. Tyler. A study of the factors influencing the difficulty of reading materials for adults of limited reading ability. *The Library Quarterly*, 4(3):384–412, July 1934.

Alice Davison and Robert N. Kantor. On the failure of readability formulas to define readable texts: A case study from adaptations. *Reading Research Quarterly*, 17(2):187–209, 1982.

Felice Dell'Orletta, Simonetta Montemagni, and Giulia Venturi. Read-it: Assessing readability of italian texts with a view to text simplification. In *Proceedings of the 2nd Workshop on Speech and Language Processing for Assistive Technologies*, pages 73–83, July 2011.

David F. Dufty, Danielle McNamara, Max Louwerse, Ziqiang Cai, and Arthur C. Graesser. Automatic evaluation of aspects of document quality. In *Proceedings of the 22nd annual international conference on Design of communication: The engineering of quality documentation*, pages 14–16, 2004.

Lars Eldén. *Matrix Methods in Data Mining and Pattern Recognition*. The SIAM series on Fundamentals of Algorithms. Society for Industrial and Applied Mathematics, 2007.

Lijun Feng. *Automatic Readability Assessment*. PhD thesis, City University of New York, 2010.

Lijun Feng, Noémie Elhadad, and Matt Huenerfauth. Cognitively motivated features for readability assessment. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, 2009.

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noémie Elhadad. A comparison of features for automatic readability assessment. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010.

Rudolph Flesch. A new readibility yardstick. *Journal of Applied Psychology*, 32 (3):221–233, June 1948.

Eibe Frank, Y. Wang, S. Inglis, G. Holmes, and Ian H. Witten. Using model trees for classification. *Machine Learning*, 32(1):63–76, 1998.

Edwad B. Fry. A readability formula that saves time. *Journal of Reading*, 11: 513–516, 1968.

Mark A. Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

Michael J. Heilman, Kevyn Collins-Thompson, Jamie Callan, and Maxine Eskenazi. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Proceedings of NAACL HLT 2007*, pages 460–467, 2007.

Michael J. Heilman, Kevyn Collins-Thompson, and Maxine Eskenazi. An analysis of statistical models and features for reading difficulty prediction. In *Proceedings of the Third ACL Workshop on Innovative Use of NLP for Building Educational Applications*, pages 71–79, June 2008.

Katarina Heimann Mühlenbock. *Matching text to readers. Assessing readability for specific target groups*. Dissertation, Språkbanken, forthcoming.

Katarina Heimann Mühlenbock and Sofie Johansson Kokkinakis. Lix 68 revisited. an extended readability measure. Technical report, Department of Swedish, Gothenburg University, 2010.

A. Honoré. Some simple measures of richness of vocabulary. *Association of Literary and Linguistic Computing Bulletin*, pages 172–179, 1979.

Tor G. Hultman and Margareta Westman. *Gymnasistsvenska*. LiberLäromedel, Lund, 1977.

J. P. Kincaid, R. P. Fishburne, R. L. Rogers, and B. S. Chissom. Derivation of new readability formulas (automated readability index, fog count, and flesch reading ease formula) for navy enlisted personnel. Technical report, U.S. Naval Air Station, Millington, TN, 1975.

Roland Larsson. *Läsbarhetsprogram KIX för IBM PC, XT och AT*. Scandinavian PC Systems AB, Växjö, 1987.

Haitao Liu. Dependency distance as a metric of language comprehension difficulty. *Journal of Cognitive Science*, 9(2):169–191, 2008.

Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3), March 2009.

Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. Machine Learning & Pattern Recognition Series. CRC Press, 2009. ISBN 978-1-4200-6718-7.

G. H. McLaughlin. Smog grading - a new readability formula. *Journal of Reading*, 22:639–646, 1969.

Ani Nenkova, Jieun Chae, Annie Louis, and Emily Pitler. *Structural Features for Predicting the Linguistic Quality of Text Applications to Machine Translation, Automatic Summarization and Human–Authored Text.*, pages 222–241. Empirical Methods in NLG. Springer-Verlag, 2010.

Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006)*, pages 2216–2219, May 2006.

OECD. International adult literacy survey, 1994.

Sarah Petersen. *Natural language processing tools for reading level assessment and text simplification for bilingual education*. PhD thesis, University of Washington, Seattle, WA, 2007.

Sarah Petersen and Mari Ostendorf. A machine learning approach toreading level assessment. *Computer Speech and Language*, 23:89–106, 2009.

Emily Pitler and Ani Nenkova. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 186–195, Honolulu, HI, October 2008.

John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, April 1998.

J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Francisco, 1993.

George Rebane and Judea Pearl. The recovery of causal poly-trees from statistical data. In *Proceedings, 3rd Workshop on Uncertainty in AI*, pages 222–228, 1987.

Frank Rosenblatt. *Principles of Neurodynamics: Perceptrons and the theory of brain mechanisms*. Spartan Books, New York, 1962.

Jonas Rybing and Christian Smith. Cogflux: Grunden till ett automatiskt textförenklingssystem för svenska. Bachelor's Thesis, Linköping University, August 2010.

Gerard M. Salton, Andrew K. C. Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, November 1975.

Sarah E. Schwarm and Mari Ostendorf. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005.

Rudy Setino, Wee Kheng Leow, and James Y. L. Thong. Opening the neural network black box: an algorithm for extracting rules from function approximating artificial neural networks. In *ICIS '00 Proceedings of the twenty first international conference on Information systems*, pages 176–186, 2000.

Vladimir Vapnik and Corinna Cortes. Support-vector networks. *Machine Learning*, (20):273–297, 1995.

Mabel Vogel and Carleton Washburne. An objective method of determining grade placement of children's reading material. *Elementary School Journal*, 28:373–381, 1928.

Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann series in data management system. Morgan Kaufmann Publishers, third edition, 2011. ISBN 978-0-120374856-0.

Victor H. A. Yngve. A model and an hypothesis for language structure. In *Proceedings of the American Philosophical Society*, pages 444–466, 1960.

Appendix A

XML-example

Preprocessing using the Korp corpus import tool generates XML-code in the format exemplified below. The sentence preprocessed is "En politiker i kristdemokraterna, kd, har försökt gå med i det socialdemokratiska ungdomsförbundet SSU."

```
<corpus>
<sentence id="0172755-01725d0">
<w pos="DT" msd="DT.UTR.SIN.IND" lemma="|en|" lex="|en..al
    .1|" saldo="|den..1|en..2|" prefix="|" suffix="|" ref="
    01" dephead="02" deprel="DT">En</w>
<w pos="NN" msd="NN.UTR.SIN.IND.NOM" lemma="|politiker|"
    lex="|politiker..nn.1|" saldo="|politiker..1|" prefix="
    |" suffix="|" ref="02" dephead="08" deprel="SS">
    politiker</w>
<w pos="PP" msd="PP" lemma="|i|" lex="|i..pp.1|" saldo="|i
    ..2|" prefix="|" suffix="|" ref="03" dephead="02"
    deprel="ET">i</w>
<w pos="NN" msd="NN.UTR.PLU.DEF.NOM" lemma="|kristdemokrat|
    " lex="|kristdemokrat..nn.1|" saldo="|kristdemokrat..1|
    " prefix="|" suffix="|" ref="04" dephead="03" deprel="
    PA">kristdemokraterna</w>
<w pos="MID" msd="MID" lemma="|" lex="|" saldo="|" prefix="
    |" suffix="|" ref="05" dephead="04" deprel="IK">,</w>
<w pos="NN" msd="NN.UTR.SIN.IND.NOM" lemma="|" lex="|"
    saldo="|" prefix="|" suffix="|" ref="06" dephead="04"
    deprel="AN">kd</w>
<w pos="MID" msd="MID" lemma="|" lex="|" saldo="|" prefix="
    |" suffix="|" ref="07" dephead="04" deprel="IK">,</w>
<w pos="VB" msd="VB.PRS.AKT" lemma="|ha|" lex="|ha..vb.1|"
    saldo="|ha..1|" prefix="|" suffix="|" ref="08" dephead=
    "" deprel="ROOT">har</w>
<w pos="VB" msd="VB.SUP.AKT" lemma="|försöka|" lex="|
    försöka..vb.1|" saldo="|försöka..1|" prefix="|för..ab
    .2|för..nn.1|föra..vb.1|" suffix="|söka..vb.1|" ref="09
    " dephead="08" deprel="VG">försökt</w>
<w pos="VB" msd="VB.INF.AKT" lemma="|gå|gå␣med|" lex="|gå..
    vb.1|gå_med..vbm.1|" saldo="|gå..1|gå..10|gå..2|gå..3|
    gå..4|gå..5|gå..6|gå..7|gå..8|gå..9|gå_med..1|" prefix=
    "|" suffix="|" ref="10" dephead="09" deprel="OO">gå</w>
<w pos="PL" msd="PL" lemma="|med|gå␣med:10|" lex="|med..ab
    .2|med..ab.1|gå_med..vbm.1:10|" saldo="|med..5|med..4|
    gå_med..1:10|" prefix="|" suffix="|" ref="11" dephead="
    10" deprel="PL">med</w>
<w pos="PP" msd="PP" lemma="|i|" lex="|i..pp.1|" saldo="|i
    ..2|" prefix="|" suffix="|" ref="12" dephead="10"
    deprel="RA">i</w>
<w pos="DT" msd="DT.NEU.SIN.DEF" lemma="|en|" lex="|en..al
    .1|" saldo="|den..1|en..2|" prefix="|" suffix="|" ref="
    13" dephead="15" deprel="DT">det</w>
<w pos="JJ" msd="JJ.POS.UTR+NEU.SIN.DEF.NOM" lemma="|
    socialdemokratisk|" lex="|socialdemokratisk..av.1|"
    saldo="|socialdemokratisk..1|" prefix="|social..av.1|"
    suffix="|demokratisk..av.1|" ref="14" dephead="15"
    deprel="AT">socialdemokratiska</w>
<w pos="NN" msd="NN.NEU.SIN.DEF.NOM" lemma="|ungdomsförbund
    |" lex="|ungdomsförbund..nn.1|" saldo="|ungdomsförbund
    ..1|" prefix="|ungdom..nn.1|" suffix="|förbund..nn.1|"
    ref="15" dephead="16" deprel="DT">ungdomsförbundet</w>
<w pos="PM" msd="PM.NOM" lemma="|" lex="|" saldo="|" prefix
    ="|" suffix="|" ref="16" dephead="12" deprel="PA">SSU</
    w>
<w pos="MAD" msd="MAD" lemma="|" lex="|" saldo="|" prefix="
    |" suffix="|" ref="17" dephead="08" deprel="IP">.</w>
</sentence>
```

Appendix B

Examples of wrongly classified documents

B.1 Documents

The examples below consist of four erroneously classified documents from a test run of hard document classification using the NoDep model and the SMO algorithm. The documents are written in Swedish and formatting irrelevant for this thesis have been introduced for the sake of readability.

B.1.1 Erroneously classified news texts

In total 31 news text documents were erroneously classified as easy-to-read, in figures B.1 and B.2 are two of them.

Det går att förbereda sig för ekonomisk kris. Men en del månader är svårare än andra. Ann-Sofie Magnusson, familjeekonom på Ikanobanken, varnar för augusti. Januari har länge varit en typisk fattigmånad. I regel får man lön innan jul, därefter följer en rad utgifter och det dröjer till saldot får sig ett lyft. Men även augusti kan bli en kritisk månad.
- Tidigare kom skatteåterbäringen då. Men de som deklarerar på internet får sina pengar redan vid midsommar och då är risken stor att semesterpengarna är slut redan i augusti.
Hur förbereder man sig för en kort ekonomisk kris?
- Man ska göra en budget och utreda hur mycket man har och var man lägger sina pengar. Då ser man var det går att dra in och vilka utgifter som är onödiga. Sen får man prioritera; är det viktigt att äta chips en gång i veckan ja, då får man avstå från läsken.
Vad ska man göra om man plötsligt inte har några pengar?
- Tömma alla skåp; kylen, frysen, skafferiet. Andra tips är att sälja saker man inte längre behöver. På internet eller genom att sätta upp lappar. I stället för att gå ut kan man ha knytkalas hemma. Åker man bil eller buss till jobbet är det smart att börja samåka, cykla eller gå.
HELENE ROTHSTEIN SYLVESTEN 031-62 40 00 ekonomired@gp.se

Figure B.1 – A news text erroneously classified as easy-to-read.

Vad är sjukersättning?
Hette tidigare förtidspension. Gäller personer mellan 30 och 64 år. För personer under 30 år är den alltid tillfällig och kallas aktivitetsersättning. Prövning ska ske senast efter 12 månaders sjukskrivning och görs sällan tidigare. Bedöms man då inte kunna återgå till arbete inom det närmsta året ska man ha sjukersättning i stället för sjukpenning
Hur stor är sjukersättningen?
I grunden 64 procent av en antagen inkomst de senaste fem åren. Till det kommer bostadstillägg som varierar från person till person.
Hur många får sjukersättning efter 12 månaders sjukskrivning?
Det finns ingen statistik på det. Inför reformen granskade man ett mindre urval, vilket visade att bara 37 procent bedömdes ha grund för sjukersättning efter 12 månaders sjukskrivning.
Vad innebär en sänkning av sjukpenningen från 80 till 75 procent efter 12 månader?
En sjuk medelinkomsttagare får 12000 kronor mindre per år att leva för. I dag finns 73 000 personer som skulle drabbas. Får man förlängd sjukpenning istället för sjukersättning kan reformen innebära högre ersättning.
Vad har arbetsgivaren för rätt att säga upp sjuka i dag?
Huvudregeln är att man inte kan sägas upp på grund av sjukdom. Om arbetsgivaren fullföljt sitt rehabiliteringsansvar och ändå inte har arbetsuppgifter för den anställde kan det vara skäl för uppsägning. När sådana fall prövas i Arbetsdomstolen brukar arbetsgivaren få rätt i något fler fall än den uppsagda, enligt Svenskt Näringslivs arbetsrättsjurist Lars Gellner.
Har arbetsgivaren rätt att avskeda mig efter 6 månaders sjukskrivning, med regeringens nya förslag?
Nej, men de fackliga organisationerna oroar sig för att det kan bli en effekt av förslaget.
Måste jag gå med på att byta arbetsuppgifter efter tre månaders sjukskrivning?
Ja.
Linus Hugo 031-62 43 76 linus.hugo@gp.se

Figure B.2 – Another news text erroneously classified as easy-to-read.

B.1.2 Erroneously classified easy-to-read texts

In total 12 easy-to-read text documents were erroneously classified as news text, in figures B.3 and B.4 are two of them.

Sveriges herrlag ställde till med en riktig knall i skidstafetten i OS. Sverige slutade trea och knep bronsmedaljerna. Bronset kom som en stor överraskning. Det var länge sedan Sverige lyckades ta medalj i stafett och i vinter har laget inte alls lyckats bra. Det är faktiskt arton år sedan Sverige tog medalj i en OS-stafett. Men i den här tävlingen fick alla fyra killarna till det perfekt.
Den som åkte allra bäst var kanske ändå Anders Södergren på den tredje sträckan. Tillsammans med italienaren Piller Cottrer ryckte Anders ifrån alla de andra lagen. Han kunde skicka ut Mathias Fredriksson som tvåa i spåret. Mathias hade en hård kamp om silvret med Tysklands Tobias Angerer. De kampen förlorade Mathias. Men vad gjorde det? Sverige hade tagit sin första stafett-medalj på arton år. De fyra skidhjältarna var glada och nöjda.
I laget åkte Mats Larsson, Johan Olsson, Anders Södergren och Mathias Fredriksson. Det svenska damlaget åkte också mycket bra i stafetten. Laget var väldigt nära att ta en medalj. Men Anna Karin Strömstedt orkade inte riktigt i spurten. Italiens Sabina Valbusa tog sig förbi och Sverige slutade på fjäde plats. Ryssland vann damstafetten.

Figure B.3 – An easy-to-read text erroneously classified as news text.

Svårt att klara skolans krav. I höstas fick eleverna i åttonde klass nya betyg. Många blev inte godkända i ett eller flera ämnen. Det är extra svårt för invandrarelever som inte kan tillräckligt mycket svenska. I Fittjaskolan i Botkyrka kommun går många invandrarelever. Cirka 60 procent av eleverna har inte fått godkänt i matematik och 40 procent har inte fått godkänt i svenska. På skolor runt om i landet är det också många som inte blivit godkända i bild, slöjd, musik, idrott och hemkunskap. För att få börja gymnasiet måste man ha godkänt i matematik, engelska och svenska, när man slutar nionde klass. Den som inte klarar det, har rätt till särskilt stöd. Skolorna får inga extra pengar till det. - Vi kanske kan ordna sommarkurser eller ha undervisning på sportlovet, säger Marianne Hellström, utbildningsledare på Fittjaskolan. Man kan också gå om ett år. - Skolan ska se till att eleverna blir godkända. Men tyvärr tror jag inte att vi kommer att klara det. Många elever som redan går på gymnasiet har också problem att bli godkända på kurserna. Därför ska Botvidsgymnasiet i Botkyrka börja med ett fjärde år i gymnasiet till hösten.

Figure B.4 – Another easy-to-read text erroneously classified as news text.