Toward an on-line preprocessor for Swedish

Mot en on-line preprocessor för svenska

Oscar Wemmert

Supervisor: Arne Jönsson
Examiner: Carine Sigoret
Upphovsrätt


Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances. The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/her own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Oscar Wemmert
Abstract

This bachelor thesis presents OPT (Open Parse Tool), a java program allowing for independent parsers/taggers to be run in sequence. For this thesis the existing java versions of Stagger and Maltparser has been adapted for use as modules in this program, and OPT’s performance has then been compared to an existing, in use, alternative (Språkbanken’s Korp Corpus Pipeline, henceforth KCP). Execution speed has been compared, and OPT’s accuracy has been coarsely tested as either comparable or divergent to that of KCP. The same collection of documents containing natural text has been fed through OPT and KCP in sequence, and execution time was recorded. The tagged output of OPT and KCP was then run through SCREAM (Sjöholm, 2012) and if SCREAM produced comparable results between the two, the accuracy of OPT was considered as comparable to KCP. The results show that OPT completes its tagging and parsing of the documents in around 35 minutes, while KCP took over four hours to complete. SCREAM performed almost exactly the same using the outputs of either program, except for one case in which OPT’s output gave better results than KCP’s. The accuracy of OPT was thus considered comparable to KCP. The one divergent example can not fully be understood or explained in this thesis, given that the thesis considers SCREAM’s internals as mostly that of a black box.
Acknowledgments

I would like to thank my supervisor Arne Jönsson for his guidance and patience, and Johan Falkenjack (previously Sjöholm) for providing me with a context in which to perform the work focused on in this thesis, access to his project SCREAM, as well as helpful advice. Finally I would like to thank my closest family, who has helped me stay motivated and acted as a steady pillar of support. Thank you, without your help this thesis would never have turned out the way it has.
Contents

Abstract ................................................................. iii
Acknowledgments .................................................. iv
Contents ................................................................. v
List of Figures ......................................................... vi
List of Tables ........................................................ vii

1 Introduction ......................................................... 1
   1.1 Background .................................................. 1
   1.2 Purpose ..................................................... 1
   1.3 Goal ........................................................... 2
   1.4 Limitations .................................................. 2

2 Theory ................................................................. 3
   2.1 Readability ................................................... 3
   2.2 Tagging and parsing ........................................... 3
   2.3 Corpus ......................................................... 4
   2.4 SCREAM ....................................................... 5
   2.5 Tool selection ............................................... 5

3 Method ................................................................. 7
   3.1 Overview ..................................................... 7
   3.2 Implementation .............................................. 7
   3.3 Run details .................................................. 8
   3.4 McNemar’s Test .............................................. 9
   3.5 Accuracy, Precision, and Recall ............................ 9

4 Results ................................................................. 11
   4.1 Preprocessing ................................................ 11
   4.2 Time taken ................................................... 11

5 Discussion ............................................................ 15
   5.1 Discussion of methods ....................................... 15
   5.2 Discussion of Results ....................................... 15

Bibliography ............................................................. 17
## List of Figures

2.1 Part-of-speech example of the sentence “Hunden jagade en liten katt på gården.” (The dog chased a small cat in the yard.) tagged using the SUC tagset. .......................... 4

2.2 Dependency tree example, using the sentence “Läkemedlet lurar bakterien genom att likna byggstenen.” (“The drug fools the bacteria by being similar to the building block.”). (a) shows the tokens, and (b) their dependency roles. .......................... 4

3.1 OPT architecture. An arbitrary number of analysis modules can be used, each using the previous module’s output as input. ................................................................. 7

3.2 OPT as implemented in this thesis. Two analysis modules were implemented for this run, one wrapping Stagger and one wrapping Maltparser. ................................. 8
List of Tables

3.1 Contingency table for McNemar’s Test .................................................. 9
3.2 Confusion matrix for classification ....................................................... 9

4.1 Preprocessing execution time, in seconds ............................................ 11
4.2 SCREAM’s classifier’s results for nominal ratio, in a contingency table for McNe-
mar’s Test ........................................................................................................ 12
4.3 SCREAM performance using the Nominal Ratio model .......................... 12
4.4 SCREAM’s classifier’s results for morpho-syntactic features, in a contingency table
for McNemar’s Test .......................................................................................... 13
4.5 SCREAM performance using the Morpho-syntactic model ...................... 13
4.6 SCREAM’s classifier’s results for syntactic features, in a contingency table for Mc-
Nemar’s Test ................................................................................................... 14
4.7 SCREAM performance using the Syntactic model .................................. 14
In the field of linguistic analysis, there exists a need to characterize different texts in respect to ease of reading. This need arises from the fact that some people have difficulties taking in complicated text. They could be new to the language or have some reading deficiency by birth, which makes it hard to access written information. For this reason, finding easy-to-read texts is important.

To aid in the task of characterization, different formulas to perform readability assessment have been developed as well as different machine learning approaches to automatically perform classification. Automated applications require reliable training material; large quantities of well annotated training texts. In the case of readability, this means both texts classified as either easy-to-read or not easy-to-read, and well tagged text to be able to extract the specific language features required to develop and use the formulas for assessing the readability.

In addition to this, a general important feature of all computation is execution speed, especially in on-line applications such as search engines.

1.1 Background

This thesis has close ties to SCREAM (Sjöholm, 2012), a project focused on development of a readability classifier, utilizing machine learning and soft classification, combining existing methods of readability assessment. The SCREAM project has shown promising results for future use, but the preprocessor used is optimized for large amounts of text, not for speed. The program would thus benefit from a faster preprocessor, performing the necessary part-of-speech (PoS) tagging and dependency parsing, as well as tokenization, sentence splitting, and lemmatization (these terms are defined in section 2.2).

1.2 Purpose

The purpose of this thesis is to produce and test the performance of a text preprocessor that performs tokenization, sentence splitting, lemmatization, PoS tagging, and dependency parsing. The work will be focused on building a modular architecture for feeding text through an arbitrary number of analysis modules. Existing taggers and parsers will be used for the text processing, forming the modules used. The performance of this chain will then be compared
with an existing preprocessor, Korp corpus pipeline (Borin, Forsberg, & Roxendal, 2012), referenced to in this thesis as KCP. The hypothesis is that selected chain will be able to perform the needed tagging in less time than KCP, and with approximately the same accuracy.

1.3 Goal

The goal is a completed Java module which given raw text, produces output divided into tokens, and tagged with the relevant part-of-speech and dependency information. This output should then be formatted as required by the user. The complete process should require less execution time than KCP.

1.4 Limitations

The established way to test the performance of a preprocessor would be to test it on a pre-tagged corpus, and then to look at accuracy, precision, and recall (these terms are defined in section 3.5). However, I am not writing new parsers, but instead an architecture for using existing parsers. As all the parsers involved have used the most complete corpuses of the Swedish language as training data, using other testing data would be difficult, and would require finding or manually tagging another corpus. Also, as these values have already been recorded for the individual parsers, this was deemed to be needlessly time consuming for little gain. Still, open parse tool’s (the program produced by this thesis, henceforth OPT) results needs to be evaluated in some way, to see that the architecture does not throw away data, or otherwise fail to work. Any test that is able to establish that OPT performs comparatively to an existing architecture should thus be acceptable. The test that was chosen was to run the output of both OPT and KCP through SCREAM, and see if SCREAM produced comparable results. Also, there exists many different tagging tools that could be tested, but this thesis will focus on just one combination due to time constraints.
2 Theory

2.1 Readability

Dale and Chall (1949) defined readability as how well a group of readers understand a given text. This definition contains two combined general elements, reader proficiency and text complexity. However, when given a text alone, with the goal to algorithmically assess its readability level, one normally only considers the text complexity by looking at its structure and contents.

The most common readability indices used for Swedish are LIX (Björnsson, 1968), OVIX and Nominal Ratio (Hultman & Westman, 1977). LIX rates long sentences and long words as more difficult to read than short ones. OVIX counts unique word tokens in the text. Few are rated as easy to read, many are rated as hard to read. Nominal Ratio works by counting and comparing the ratio of word classes to each other.

Modern machine analysis techniques allow for more advanced linguistic features to be extracted and used to find patterns. This facilitates the creation and use of readability indices that rely on more richly tagged text material.

2.2 Tagging and parsing

Tokenization

To be able to perform an analysis on natural text, it first needs to be tokenized, which means divided up into so called tokens. A token is normally one word, but also includes elements such as punctuation. Following tokenization more information can be extracted by applying different analysis steps.

Sentence splitting

Many analysis methods of natural language work on a sentence level, and require that these sentences are extracted. The information required for this can often be obtained by looking at punctuation tokens, but is complicated by things such as newspaper titles that end without any punctuation.
2.3. Corpuses

Part-of-Speech tagging

PoS tagging, is described by Östling (2012) as the task of syntactic disambiguation of natural language. PoS tagging aims to annotate each word token in a text with its part of speech and (often) morphological features. An example of this can be found in figure 2.1.

<table>
<thead>
<tr>
<th>Word</th>
<th>PoS</th>
<th>Tagset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hunden</td>
<td>nn ut sg def nom</td>
<td></td>
</tr>
<tr>
<td>jagade</td>
<td>vb prt akt</td>
<td></td>
</tr>
<tr>
<td>en</td>
<td>dt ut sg ind</td>
<td></td>
</tr>
<tr>
<td>liten</td>
<td>jj pos ut sg ind nom</td>
<td></td>
</tr>
<tr>
<td>katt</td>
<td>nn ut sg ind nom</td>
<td></td>
</tr>
<tr>
<td>på</td>
<td>pp</td>
<td></td>
</tr>
<tr>
<td>gården</td>
<td>nn ut sg def nom</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.1: Part-of-speech example of the sentence “Hunden jagade en liten katt på gården.” (The dog chased a small cat in the yard.) tagged using the SUC tagset.

Dependency parsing

Dependency parsing is a method for extracting the dependency structure in a given sentence. In this structure, words are linked, being either head or dependent (Hall, 2006). Arcs are created between these, and make up the dependency graph produced, as shown in figure 2.2.

Figure 2.2: Dependency tree example, using the sentence “Läkemedlet lurar bakterien genom att likna byggstenen.” (“The drug fools the bacteria by being similar to the building block.”). (a) shows the tokens, and (b) their dependency roles.

2.3 Corpuses

In this thesis, a number of corpuses and other kinds of annotated collections of text will be referenced, briefly described below.

SUC

SUC, the Stockholm-Umeå Corpus, is currently the most comprehensive corpus of written Swedish available. It contains a balanced collection of texts, covering various text types and stylistic levels. Texts are taken from a variety of sources, such as books, magazines, newspapers, prose and so on. SUC has been continuously developed since it was first released in 1997, and there exists three major versions. Version 2.0 was made available in 2006, and improved on 1.0 by adding extended annotations as well as bonus material. Version 3.0, released 2012, provides improved annotations and more bonus material. SUC contains roughly 1 million
tagged tokens, and is the most commonly used corpus for obtaining training data for general purpose taggers. (Gustafson-Capková & Hartmann, 2006)

LäSBarT

LäSBarT is a specialized corpus of easy-to-read texts, containing texts meant to be read by children, young adults, or people with reading difficulties. It comprises roughly 1.3 million tokens, taken from the fiction, information, and news categories (Heimann Mühlenbock, 2013, p.54).

GP2007

GP2007 is a corpus of text originating from the newspaper Göteborgs-Posten from the year 2007, available as a resource at Göteborgs Universitets Språkbanken. In this thesis, this corpus is used as reference material, acting as not easy-to-read texts. These texts can be considered to be normal, as opposed to easy-to-read.

2.4 SCREAM

SCREAM set out to create and investigate performance of different machine learning approaches to readability assessment compared to traditional methods (Sjöholm, 2012).

In this work, readability classifiers were constructed using soft classification. Soft classification is a statistical method for calculating the probability of an item being of type A or B. In SCREAM, this method is used to determine whether a text is easy-to-read or not. The analysis was based on four different levels; shallow structural information, lexical composition, morpho-syntactic structure and syntactic structure.

The shallow structural level analysis only measures surface information, counting word length both as number of characters and syllables, and sentence length as number of words contained in the sentence. Lexical composition is, in addition to information from the shallow level, also based on frequency counts of predefined Swedish words contained in SweVoc vocabulary (Mühlenbock & Kokkinakis, 2012), where these words have previously been shown to be easy to read. The morpho-syntactic level further adds morphological information, by measuring ratios of different PoS tags. The fourth, and most complex level, analyses the syntactic structure. Here the complexity of dependencies is measured, by applying many different metrics to this syntactic structure. Examples of metrics include average sentence depth, and the average length of dependency links.


2.5 Tool selection

Since a large part of this thesis has been of a practical nature, with a focus on producing a program that is agnostic to the tools it uses for tagging and parsing, the inner workings of these tools were not considered as important as they otherwise might have been. However, some criteria for selection of suitable analysis tools were still important. These criteria included three elements: the analysis tools should be readily available for research purposes, contain java interfaces, and should have shown good results for the Swedish language. Stagger and Malt-parser fulfill these requirements, and were thus selected. Stagger performs the tasks of tokenization, sentence splitting, lemmatization, and PoS tagging. Malt-parser does dependency parsing. A brief description of each of these can be found below.
2.5. Tool selection

Stagger - The Stockholm Tagger

This software was developed by Östling (2012), and is a Swedish language PoS tagger performing tokenization, sentence splitting, lemmatization and PoS tagging. The main difference from previous PoS taggers for the Swedish language is that Stagger uses a feature-rich model; combining a large number of history-dependent and history-independent linguistic features. It is based on the Averaged Perceptron algorithm by Collins (2002). The performance when trained and tested on SUC 3.0 (Gustafson-Capková & Hartmann, 2006) lies around 96.6%, which compares favourably to other PoS taggers for the Swedish language. In this thesis the training data used was the Swedish language model supplied by Stagger. Stagger is open source, and has a java interface.

Maltparser

Maltparser, written by Nivre, Hall, and Nilsson (2006), was used to perform the required dependency parsing. It generates a parser using deterministic algorithms, coupled with history based feature models and discriminative learning. It can be applied to different languages, by using a treebank of that particular language. I have used a pre-trained Swedish parser model based on Swedish Treebank, which is a combination of Talbanken (Nivre, Nilsson, & Hall, 2006) and SUC.

Korp

Korp consists of three main parts: Korp frontend, Corpus pipeline and Korp backend. The purpose is to annotate and export corpora in the standardized corpus format of Språkbanken. It was specifically developed to handle large corpora (Borin et al., 2012). This thesis only concerns itself with the Corpus pipeline, which performs tokenization, sentence splitting, links to the lexical persistent identifiers, lemmatization, compound analysis, PoS tagging, and syntactic dependency trees. This thesis only concerns itself with the corpus pipeline, referred to as KCP The pipeline consists of Natural Language Toolkit (NLTK) (Bird, 2006), HunPoS (Halácsy, Kornai, & Oravecz, 2007), and Maltparser. In this pipeline NLTK performs tokenization, sentence splitting and lemmatization. HunPoS performs PoS tagging, and Maltparser does dependency parsing. HunPoS was trained on SUC2 and Maltparser on Swedish Treebank.
3 Method

3.1 Overview

This thesis project has to a large extent been practical. I have written a text preprocessor, a Java program (OPT, Open Parse-Tool), which parses text, and with the help of analysis modules produces tagged text as output. The analysis modules used in my implementation contain and wrap Stagger and Maltparser.

Suitable text collections were run through OPT, and XML containing tagged data was output. The same collections were then run and tagged by KCP. KCP was considered to be my baseline. Execution time taken by each was noted.

For the above, two collections of text were selected, with one of them being considered easy-to-read, and the other being considered not easy-to-read. The reason for this is that the SCREAM classifier requires this configuration.

The next step was to use the processed texts from OPT and KCP, as input for SCREAM. The resulting classification data was then used as an approximation of tagging accuracy. If SCREAM performed comparably when using data tagged by OPT and KCP, OPT’s accuracy could be considered comparable to that of KCP.

3.2 Implementation

OPT is designed to be used with an arbitrary number of analysis tools. For each analysis tool there is a module, containing a bridge which formats input from OPT’s internal representation, and a parser which performs the analysis. OPT then takes the output of the parser and updates its internal representation. When this is done for all modules the output is formatted as specified by the user, as seen in figure 3.1.

![OPT architecture diagram](image)

Figure 3.1: OPT architecture. An arbitrary number of analysis modules can be used, each using the previous module’s output as input.
3.3 Run details

I have used two text compilations containing natural text, following the procedure outlined in Sjöholm (2012). One from the easy-to-read corpus LäSBarT (Heimann Mühlenbock, 2013, p. 54), and the other from a corpus based on the newspaper Göteborgsposten from 2007, which in this thesis is considered not to be easy-to-read. Each text compilation consists of 700 separate documents, of various lengths.

Stagger was run with the supplied swedish model available at http://www.ling.su.se/english/nlp/tools/stagger/stagger-the-stockholm-tagger-1.98986 and Maltparser using the swemalt-1.7.2.mco configuration available at http://www.maltparser.org/mco/swedishparser/swemalt.html.

First the 700 LäSBarT documents were run through OPT, and the execution time was noted. Then the same was done with the 700 GP2007 documents. The same text compilations were processed with KCP, to perform the same analysis. The program was run as supplied. Execution time was again noted.

Data from OPT and KCP was then fed into SCREAM, and results compared.

I have chosen to analyse the texts at the document level, rather than at the sentence level, since it generally produced better results in Sjöholm, 2012 and would be more meaningful for classifying web pages in the future. SCREAM was run using the Sequential Minimal Optimization (SMO) algorithm with “test all models” setting, SMO being a java based Support Vector Machine (SVM) learner. SMO was chosen as it generally produced the best results for document level classification (Sjöholm, 2012).

SCREAM contains models that work on several different levels of analysis, from very simple to more complex. This thesis has been focused on the models that use and depend on information generated by OPT and KCP. While all models can be said to depend on tokenization, simple models working only on a shallow level, like word length and sentence length, were considered less interesting than models that also depend on information generated by the main components of OPT and KCP. These components being PoS-tagging and dependency parsing.

The models in SCREAM have not been modified, and exactly correspond to those found in Sjöholm, 2012. Stagger performs PoS tagging, so any model that depends on this was automatically included. Maltparser does dependency parsing, so any model that uses this information was also included. Three models were thus included, namely Nominal Ratio, Morpho-syntactic Features and Syntactic Features.
SCREAM trains a number of classifiers (one per model), on the 1400 documents, and then tests these classifiers’ abilities to classify texts as easy-to-read or not. A classifier trained using the same model, but differing input data, also produces slightly differently trained classifiers. One can then regard these classifiers as being separate, and test if they differ statistically. This test was performed using McNemar’s Test (section 3.4, see below).

### 3.4 McNemar’s Test

McNemar’s test was proposed as a method to judge the significance of the difference between correlated proportions (McNemar, 1947). It uses a 2x2 contingency table, as shown in table 3.1. McNemar’s was chosen for this thesis since it was a recommended test in Dietterich, 1998.

<table>
<thead>
<tr>
<th></th>
<th>Test 1 YES</th>
<th>Test 1 NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 2 YES</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Test 2 NO</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>A+C</td>
<td>B+D</td>
<td>N</td>
</tr>
</tbody>
</table>

Referencing table 3.1, the cell containing A denotes the number of instances where both tests gave a positive result, D where both gave a negative result, and B and C where the two tests differed. The null hypothesis states that $p_b = p_c$, meaning that the probability of B and C is equal, indicating no significant difference between the two tests.

The original formula for McNemar’s Test is:

$$\chi^2 = \frac{(B - C)^2}{B + C}$$

There also exists a continuity corrected version, shown below. This corrects for the fact that the $\chi^2$ distribution is continuous, and not discreet (Dietterich, 1998, p.7).

$$\chi^2 = \frac{(|B - C| - 1)^2}{B + C}$$

In this thesis McNemar’s Test will be used with the continuity correction to test if the two classifiers trained by SCREAM on the same model can be proven to differ statistically.

### 3.5 Accuracy, Precision, and Recall

<table>
<thead>
<tr>
<th></th>
<th>Prediction positive</th>
<th>Prediction negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition Positive</td>
<td>True Positive (TP)</td>
<td>False Negative (FN)</td>
</tr>
<tr>
<td>Condition Negative</td>
<td>False Positive (FP)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

Referencing table 3.2, the cell containing TP denotes the number of instances where the prediction correctly returned positive, TN where prediction correctly returned negative, FN where prediction falsely returned negative, and FP where prediction falsely returned positive. In this thesis, precision and recall will be provided for both GP2007 and LäSBarT. For LäSBarT, TP means the times the classifier guessed that a text was easy-to-read, and was correct. For GP2007, TP means the times the classifier guessed that a text was not easy-to-read, and was correct. The other alternatives follow from the definition above.

Given this definition, Accuracy, Precision, and Recall are defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
3.5. Accuracy, Precision, and Recall

Precision = \frac{TP}{TP + FP}

Recall = \frac{TP}{TP + FN}
4 Results

4.1 Preprocessing

4.2 Time taken

Run time for KCP and OPT to finish their tagging processes on each document collection was recorded and is shown in table 4.1.

Table 4.1: Preprocessing execution time, in seconds.

<table>
<thead>
<tr>
<th></th>
<th>GP2007</th>
<th>LäSBarT</th>
<th>sum total</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCP</td>
<td>8400</td>
<td>6600</td>
<td>15000</td>
</tr>
<tr>
<td>OPT</td>
<td>1200</td>
<td>900</td>
<td>2100</td>
</tr>
</tbody>
</table>

As can be seen OPT performs its preprocessing faster than KCP, as expected. The difference is large, however, and this is a step in the right direction for OPT’s use in a real-time applications.
Nominal Ratio

Nominal Ratio requires and used PoS tags, therefore Nominal Ratio is tested.

Table 4.2: SCREAM’s classifier’s results for nominal ratio, in a contingency table for McNe- mar’s Test.

<table>
<thead>
<tr>
<th></th>
<th>OPT Correct</th>
<th>OPT Incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCP correct</td>
<td>725</td>
<td>37</td>
</tr>
<tr>
<td>KCP incorrect</td>
<td>284</td>
<td>354</td>
</tr>
<tr>
<td>total</td>
<td>1009</td>
<td>391</td>
</tr>
</tbody>
</table>

Statistically significant differences in SCREAM’s performance when using OPT or KCP as preprocessor were observed using McNemar’s test for the nominal ratio model ($\chi^2 = 188.523, p = 0.000, p < 0.001$) (table 4.2).

Table 4.3: SCREAM performance using the Nominal Ratio model.

<table>
<thead>
<tr>
<th></th>
<th>LaSBarT</th>
<th>GP2007</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>KCP</td>
<td>54.3</td>
<td>53.0</td>
</tr>
<tr>
<td>OPT</td>
<td>72.0</td>
<td>65.1</td>
</tr>
</tbody>
</table>

As seen in table 4.3, SCREAM performs better with SMO when the material is preprocessed by OPT. Results are closer to that of other algorithms used in Probability as readability (Sjöholm, 2012). The accuracy of KCP at around 50% is not much better than random guessing, or always choosing answer A over B. The near 100% Recall on LaSBarT, and lower Recall on GP2007 indicated a heavy bias towards classifying texts as easy-to-read. OPT seems to perform better here, but the Recall on GP2007 is still much lower than that on LaSBarT, indicating that this bias still exists, although to a lesser degree.
Morpho-syntactic Features

The morpho-syntactic model is also relevant to test because it uses information from the PoS tags.

Table 4.4: SCREAM’s classifier’s results for morpho-syntactic features, in a contingency table for McNemar’s Test.

<table>
<thead>
<tr>
<th></th>
<th>OPT Correct</th>
<th>OPT Incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCP correct</td>
<td>1347</td>
<td>9</td>
</tr>
<tr>
<td>KCP incorrect</td>
<td>10</td>
<td>34</td>
</tr>
<tr>
<td>total</td>
<td>1357</td>
<td>43</td>
</tr>
</tbody>
</table>

No statistically significant differences in SCREAM’s performance when using OPT or KCP as preprocessor were observed using McNemar’s Test for the morpho-syntactic model ($\chi^2 = 0.000, p = 1.000, p > 0.05$) (table 4.4).

Table 4.5: SCREAM performance using the Morpho-syntactic model.

<table>
<thead>
<tr>
<th></th>
<th>LäSBarT</th>
<th>GP2007</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Precision</td>
</tr>
<tr>
<td>KCP</td>
<td>96.9</td>
<td>96.9</td>
</tr>
<tr>
<td>OPT</td>
<td>96.9</td>
<td>96.5</td>
</tr>
</tbody>
</table>

As seen in table 4.5, results seem to be very close. This would indicate that the PoS tagging done by Stagger is up to par with that of KCP.
4.2. Time taken

Syntactic Features

The syntactic model is also relevant because dependency parsing is used.

Table 4.6: SCREAM’s classifier’s results for syntactic features, in a contingency table for McNemar’s Test.

<table>
<thead>
<tr>
<th></th>
<th>OPT Correct</th>
<th>OPT Incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCP correct</td>
<td>1356</td>
<td>16</td>
</tr>
<tr>
<td>KCP incorrect</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>total</td>
<td>1360</td>
<td>40</td>
</tr>
</tbody>
</table>

No statistically significant differences in SCREAM’s performance when using OPT or KCP as preprocessor were observed using McNemar’s Test for the syntactic model (\( \chi^2 = 3.682, p = 0.055, p > 0.05 \)) (table 4.6).

Table 4.7: SCREAM performance using the Syntactic model.

<table>
<thead>
<tr>
<th></th>
<th>LaSBarT</th>
<th>GP2007</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Precision</td>
</tr>
<tr>
<td>KCP</td>
<td>97.9</td>
<td>97.5</td>
</tr>
<tr>
<td>OPT</td>
<td>97.1</td>
<td>96.8</td>
</tr>
</tbody>
</table>

As seen in table 4.7, the results are similar. While one could expect identical results as Maltparser is used by both programs for this line of analysis, they receive different data as input, so a small divergence would be understandable.
5 Discussion

5.1 Discussion of methods

Using a method like looking at the output of SCREAM for checking accuracy may seem unusual and maybe even confusing. The normal way to go about this would be to test the parsers on well tagged training data, and this way get reliable and comparable results. However, the analysis tools used have already been rigorously tested in this way and have not been modified for this thesis.

Both OPT’s and KCP’s PoS taggers (Stagger and HunPoS) have both been trained on versions of the same data (SUC), and there exist benchmarks for both on SUC2.0 training data. Stagger attains an accuracy of 96.4% (Östling, 2013, p. 11), while HunPoS attains 95.9% (Megyesi, 2009, p. 3).

The version of Stagger used in this thesis was trained on SUC3.0, and while this offers an increase in accuracy for Stagger, it is minor, around 0.18% (Östling, 2013, p. 11).

Maltparser is used by both OPT and KCP, so given the same PoS input they should produce the same output. Since all these perform very similarly, one can assume that SCREAM’s results would also be very similar when used on their output. Any larger difference found can thus be seen as unexpected, and something worth investigating.

5.2 Discussion of Results

OPT completes the tagging in much less time than Korp when run on a single computer. The fact that OPT is faster is by itself not unexpected, as OPT was meant for this usage. Korp, however, has been designed to handle large corpora by dividing the data into chunks that can be processed in parallel. This process involves a large amount of writing to disk, which is slower than keeping data in memory. OPT was designed to be able to do the entire process in memory, with no disk access required other than initializing the language models. When the text volume increases over a certain level, OPT can optionally read input from disk, and write output to disk, however large corpora are not the main target of the program. When operating on a document level, this is never a problem. Korp also performs more lines of analysis, in the form of links to the lexical persistent identifiers, and compound analysis and these extra steps will consume some time.
This thesis has not been able to determine how large a part of the difference this represents, as testing this in isolation has not been possible. It is however unlikely that these extra steps could account for the entire time difference. Therefore, while this may not be an entirely fair comparison, it is still valid for the purpose of producing a more efficient preprocessor for applications that do not require these extra steps.

After analysis of SCREAM run data, one noticeable difference was found. Using Nominal Ratio as readability classifier the result showed 18\% higher accuracy (significant ...) when OPT was used as the preprocessor instead of KCP. The performance of Nominal Ratio with KCP’s tagged data combined with SMO does not seem to work very well. This was also true in Probability as Readability (Sjöholm, 2012, p. 48). However, when using OPT as preprocessor, SCREAM produces results that comes closer to that of the other algorithms in his results. The reasons for this are difficult to get at, since I have neither tested the actual accuracy of the tagged data, or have done any analysis of the inner workings of SCREAM. Although a more detailed investigation is outside the scope of this thesis, some possible explanations seem reasonable. Perhaps the accuracy of tagging on the selected texts varies more than the benchmarks would have us believe, despite both being trained on SUC. Or if the taggers have tagging biases, and some of these may be better for readability assessment than others. Or there may simply be a problem with OPT or SCREAM, and/or SMO for this task.
Bibliography


