Language for Granted

Automatic Evaluation of the Language used in Grant Applications

Master's thesis

Peeter Sällström Randsalu
Språkteknologiprogrammet
Department of Linguistics
Lund University, Sweden

November 23, 2007

Abstract

This thesis examines whether it is possible to ascertain a measurable difference between granted and refused grant applications with automatic methods. A corpus of project descriptions from the Swedish Science Council was examined with different classification techniques and using different linguistic features. A Naive Bayes classifier is shown to be a good predictor for this type of problem and the number of prepositional phrases in a document is shown to be a good attribute for classification. The results show that there does exist a statistically measurable linguistic difference between granted and refused applications.

Sammanfattning

Denna uppsats undersöker om det går att, med automatiska metoder, mäta en skillnad mellan godkända och avslagna bidragsansökningar. En korpus med projektbeskrivningar från Vetenskapsrådet undersöktes med olika klassifikationstekniker för olika lingvistiska särdrag. Det visar sig att en "Naive Bayes"-klassificerare fungerar bra för denna slags problem och också att antalet prepositionsfraser i ett dokument skulle kunna vara en bra utgångspunkt för klassificering. Resultaten visar slutligen att det finns en statistiskt mätbar språklig skillnad mellan godkända och avslagna ansökningar.

Supervisor: Caroline Willners http://www.ling.lu.se/persons/CarolineWillners.html

Contents

List of Tables

Introduction

In matters of grave importance, style, not sincerity, is the vital thing.
Oscar Wilde, *The Importance of Being Earnest, Act III*

Taking language for granted

Where it is trivially true that the way language is used is important in everyday communication situations, it may not be as self-evident that the way language is used is equally important when trying to communicate something where the subject is not primarily expressed through language.

An application for a grant for a scientific project can be thought of as a way of communicating a scientific thesis to a granting body, and one could easily imagine that the only thing of importance in such a situation would be *what* is said, not *how* one goes about saying it.

If all grant applications had only consisted of tables and charts which had all been designed in the same way all applications would have the same chances, as there would be nothing but data to build an assessment on. This is far from the case, though, and as a grant application is expressed through language, the way language is used becomes a factor in the assessment process, whether you like it or not.

There is a tendency to take the language of scientific texts for granted and not pay enough attention to how language is used. This is especially unfortunate when dealing with something as important as grant applications, where the fates of entire projects hang in the balance.

Purpose

What I want to examine in this thesis is whether the quality (linguistic, stylistic, etc.) of the language used in a grant application has any measurable impact on the chances of a project's application being granted and, subsequently, if this can be accurately predicted by purely automatic means.

The aim of this thesis is thus to see if there exists a way to automatically determine from the project description whether a research grant application will be granted or refused. This can be considered as a text categorization problem. The question is, then, whether a grant application document d can be classified into the category c_1, granted applications, or the category c_2, refused applications, according to the following definitions (Sebastiani, 2002, 2005):

> [Text Categorization] may be formalized as the task of approximating the unknown *target function* $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ (that describes how documents ought to be classified, according to a supposedly authoritative expert) by means of a function $\hat{\Phi} : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ called the *classifier*, where $\mathcal{C} = \{c_1, \ldots, c_{|c|}\}$ is

> a predefined set of categories and $\mathcal{D}$ is a (possibly infinite) set of documents. If $\Phi(d_j, c_i) = T$, then d_j is called a *positive example* (or a *member*) of c_i, while if $\Phi(d_j, c_i) = F$ it is called a *negative example* of c_i.

To see how different (semantic, stylistic, etc.) connections can be made between the texts I will use a number of statistical and machine learning techniques.

Outline of the thesis

The thesis consists of five main parts. Part 1 consists of background information and an overview of previous research in the field. In part 2 the tools and resources used are presented. Part 3 presents the methods I use to test my hypothesis. Part 4 presents the experimental results and part 5 concludes with a discussion of the results and suggestions for future work.

1 Background

It is very much more difficult to talk about a thing than to do it.
Oscar Wilde, *Intentions*

Sebastiani (2002, 2005) are good introductions to the field of automated text categorization where Sebastiani (2005) is a slightly less technical overview of the field. In it are listed several main applications of text categorization:

- Automatic indexing/automated metadata generation
- Document organization
- Text filtering
- Hierarchical categorization of web pages
- Word sense disambiguation
- Automated survey coding
- Automated authorship attribution and genre classification
- Spam filtering

None of these main categories really fit my project description. The problem of classifying applications into granted and refused categories is similar to other problems in the same area, however.

One classification problem not mentioned in the list above is automatic classification into readability levels. This is similar to the problem presented in this thesis, as it deals with the classification of texts into discrete levels or categories, where all texts are assigned a level or category.

- Larsson (2006) is an example which presents a solution for classification into readability levels for Swedish, using a number of classification models, which were induced by training a Support Vector Machines classifier on features established by previous research as good measurements of readability. The features were extracted from a corpus annotated with three readability levels. Empirical testings of different feature combinations were performed to optimize the classification models to render a good and stable classification, where the best model obtained an F-score of 89.88.
- Liu et al. (2004) is also a Support Vector Machines-based approach, which attempts to automatically recognize reading levels from user queries to a search engine. Results show that the proposed method, where a model was induced from authentic user queries using Support

Vector Machines, performed significantly better than standard readability indices and that it could achieve a recognition accuracy close to or well above 80% for both 2-category and 3-category cases.

- Collins-Thompson and Callan (2005) is an attempt to classify Web pages according to their reading difficulty level using multinomial Naive Bayes with a smoothed unigram model. By using a mixture model to interpolate evidence of a word's frequency across grades, a classifier was built that achieved an average root mean squared error of between one and two grade levels for 9 of 12 grades. Such classifiers have very efficient implementations and can be applied in many different scenarios. The models can be varied to focus on smaller or larger grade ranges or easily retrained for a variety of tasks or populations.

As all three examples of readability classification above use different methods and different evaluation measurements (F-score, accuracy and root mean squared error), it is difficult to say which one is the most successful, as it is not possible to directly compare the three. All three, however, show that automatic readability classification with good results is possible.

The differences between readability classification and my problem is that I use only two categories, granted and refused application, and that I try to classify something slightly less tangible than readability, as it is probably not readability alone (whatever measurement one chooses to use), which determines whether a grant application is granted or refused. It should also be said that all three models above in some way deal with the readability classification of Web pages or user queries to search systems which are typically quite short, whereas my work deals with complete written texts of some length.

Another example of analysis and scoring of open-ended written work similar to mine and not mentioned in the list above is automatic essay grading, where texts are automatically categorized into one of several grade levels or scores.

- Larkey (1998) is an example where several standard text-categorization techniques were applied to the problem of automated essay grading. Bayesian independence classifiers and k-nearest-neighbor classifiers were trained to assign scores to manually-graded essays. These scores were combined with several other summary text measures using linear regression. The classifiers and regression equations were then applied to a new set of essays. The classifiers worked very well and the agreement between the automated grader and the final manual grade was as good as the agreement between human graders.

- In Landauer et al. (2003) essays were graded by the so called Intelligent Essay Assessor (IEA) using Latent Semantic Analysis (LSA),

> a model which induces the semantic similarity of words and passages by analysis of large bodies of domain-relevant text. IEA's dominant variables are computed from comparisons with pre-scored essays of highly similar content as measured by LSA. The study shows that the scores of IEA correlated on average within two percentage points of the correlations between two human graders of the same essay.

Automatic essay grading is similar to my problem in that essay grading is a task using a number of variables, whereas readability classification only classifies using one variable, readability. As in readability classification and as opposed to my problem, however, the number of levels used are more than two when essays are not simply graded into passed or failed essays.

The different examples of techniques and results above show that automatic machine classification of texts in the manner I propose in this essay is a viable proposition, although none of the examples above use a binary classification such as mine, classifying into only two categories.

For background on methods and measurements for text classification for Swedish see Platzack (1974) on readability, Svensson (1993) and Nordman (1992) on LSP (Language for Specific Purposes) and Melin and Lange (2000) on stylistics.

2 Tools and Resources

Oh, I like tedious, practical subjects.
What I don't like are tedious, practical people.
There is a wide difference.
Oscar Wilde, *An Ideal Husband, Act I*

2.1 Programming

All scripting and programming is done in Python, a simple but powerful open source high level programming language available for free from www.python.org for a wide range of hardware and software configurations. The version used is Python 2.5.1, the latest as of this writing.

Python has a well established user base and has an extensive library of functions for many statistical and linguistic methods, e.g. NumPy for numeric processing, the PyMC package for Bayesian statistics, the LIBSVM package (Chang and Lin, 2001) for working with Support Vector Machines and the Natural Language Toolkit (NLTK) (Bird et al., 2007), which contains a number of modules for working with human language.[1]

Figure 1: Force-feeding a python

[1]See Bird et al. 2007, Appendix B for a comparison of Python versus other programming languages for natural language processing tasks.

2.2 The corpus

The corpus used consists of project descriptions from the project database of Vetenskapsrådet, the Swedish Research Council, collected from the Internet (www.vr.se).

When first embarked upon, my project was meant to create a template for EU-grant applications, to maximize the chances of an application being granted. There was no possibility, however, to access the refused applications from the EU archives. An attempt at categorization into two categories (granted/refused applications) based on data from only one category posed problems that did not seem worthwhile to grapple with in a thesis of this kind.

The project descriptions in my corpus are abstracts, however, and not the entire grant applications, so I make the assumption that the style and quality of language in the abstract is an accurate reflection of the language in the actual grant application. Using the abstracts also has the advantage of not having to filter out extra-linguistic data such as figures, tables and equations from the documents.

The HTML documents collected from the Internet were stripped of all HTML tags and all extraneous information (names, dates, funds granted, etc.), and the resulting pure text documents, partitioned into granted and refused applications, were used as a base for further processing.

The pure text corpus obtained in this fashion, with documents not in Swedish and documents containing no text (of which there actually were a few) removed, contained 26,442 grant application abstracts from between 2002 and 2006. 6,024 (23 %) of the applications were granted and 20,418 (77 %) refused. For further statistics see Table 1.

Table 1: Corpus statistics

corpus	documents	sentences	words	sentences/ document	words/ sentence
entire	26,442	570,594	12,016,117	21.58	21.06
granted	6,024	132,323	2,796,210	21.97	21.13
refused	20,418	438,271	9,219,907	21.46	21.04

2.3 Part-of-speech tagging

Part-of-speech tagging is the labeling of words with tags containing information about the words' parts of speech, usually the word class and information about e.g. inflexion, gender or case. There are a number of automatic methods for part-of-speech tagging, e.g. rule-based, memory-based and statistical methods.

Thorsten Brants' Trigrams'n'Tags (TnT) tagger (Brants, 2000) is a language and tagset independent statistical tagger based on second order Markov models (Manning and Schütze, 1999, chapter 6). The TnT tagger uses trigrams, that is information about the tags for the two previous words to predict the probability for the tag for the current word, and smoothing is done by using linear interpolation of trigrams, bigrams and unigrams. TnT is optimized for speed and is robust, in that it provides every word with a tag. Average part-of-speech tagging accuracy is between 96% and 97%, depending on language and tagset (Brants, 2000). The TnT tagger has also been shown to have the highest overall accuracy when tagging both known and unknown words for Swedish (Megyesi, 2002; Sjöbergh, 2006), when trained on the Stockholm Umeå Corpus (SUC, 1997) .

I used the TnT tagger[2] with the Swedish language models[3] developed by Beata Megyesi (Megyesi, 2002) to tag the entire corpus. I used a version of the language models which uses the Parole tagset, to make the tags compatible with the SPARKchunk parser (see section 2.4).

TnT adds the part-of-speech tags to the words, separated with a slash (`/`), i.e.

```
Vi/PF@UPS@S kommer/V@IPAS sedan/RGOS att/CIS undersöka/V@NOAS
resultaten/NCNPN@DS ./FE
```

An explanation of the tags used in the example is shown below. `@` and `0` stand for features that are not applicable for the specific word. `S` in the end of the tag is short for standard, as opposed to `A` - abbreviation and `C` - compound. The entire Parole tagset is available (with Swedish descriptions) at spraakbanken.gu.se/parole/tags.phtml.

```
Vi/P(pronoun) F(personal) @ U(utrum) P(plural) S(subject) @ S
kommer/V(verb) @ I(indicative) P(present) A(active) S
sedan/RG(adverb) 0 S
att/CI(infinitive mark) S
undersöka/V(verb) @ N(infinitive) 0 A(active) S
resultaten/NC(noun) N(neutrum) P(plural) N(nominative) @ D(definite) S
./FE(punctuation)
```

TnT, the Swedish language models and SUC are free of charge for non-commercial research purposes.

[2] from www.coli.uni-saarland.de/∼thorsten/tnt/

[3] available at stp.lingfil.uu.se/∼bea/resources/tnt

2.4 Shallow parsing

Chunking is an efficient and robust method for identifying short phrases in text, or "chunks". Chunks are syntactically related non-overlapping groups of words. A chunk usually consists of a phrasal head word (such as a noun) and the adjacent modifiers and function words (such as adjectives and determiners).

In traditional chunking, however, the internal structure of the chunks is not analyzed, i.e. if a word belongs to an adjective phrase which in turn belongs to a noun phrase, the word is labeled as belonging to the noun phrase tag only, not marking any other lower nodes in the tree, as it were. This is not optimal, for instance, if we are interested in counting all types of phrases in a text. What is required here is something more than a chunker, but not a full parser with all the complexity that entails.

SPARKchunk by Beata Megyesi (Megyesi, 2002) is a shallow parser based on a context-free-grammar for Swedish. SPARKchunk uses the Earley algorithm implemented in Python by John Aycock (Aycock, 1998) to parse PoS-tagged data into a number of different phrase types that represent the hierarchical structure of a sentence. Here follows a brief description of the phrases used in SPARKchunk, taken from Megyesi (2002):

- Adverb Phrase (`ADVP`) – adverbs that can modify adjectives, numerical expressions or verbs. e.g. "very"
- Minimal Adjective Phrase (`APMIN`) – the adjectival head and its possible modifiers, e.g. "very interesting"
- Maximal Adjective Phrase (`APMAX`) – more than one AP with a delimiter or a conjunction in between, e.g. "very interesting and nice"
- Noun Phrase (`NP`) – the head noun and its modifiers to the left, e.g. "Pilger's very interesting and nice book"
- Prepositional Phrase (`PP`) – one or several prepositions delimited by a conjunction and one or several NPs, or in elliptical expressions an AP only. e.g. "about politics"
- Verb Cluster (`VC`) – a continuous verb group belonging to the same verb phrase without any intervening constituents, e.g. "would have been"
- Infinitive Phrase (`INFP`) – an infinite verb together with the infinite particle and may contain AdvP and/or verbal particles. e.g. "to go out"
- Numeral Expression (`NUMP`) – numerals with their possible modifiers, e.g. "several thousands"

The PoS-tagged corpus output from TnT was subsequently tagged with SPARKchunk. The version used adds the phrase tags to the ends of the PoS-tagged words, separating them with an underscore (_), i.e.

```
Vi/PF@UPS@S_NPB kommer/V@IPAS_VCB sedan/RG0S_ADVPB att/CIS_INFPB
undersöka/V@N0AS_INFPI resultaten/NCNPN@DS_NPB ./FE
```

Each phrase type is also represented with an additional tag marking position information, where `XB` marks the initial word of the phrase `X`, and `XI` marks a non-initial word inside the phrase `X`. There is also a tag `O` for words outside any phrase.

SPARKchunk is available free of charge for non-commercial research purposes from stp.lingfil.uu.se/∼bea/resources/spark.

2.5 Data analysis

WEKA (Waikato Environment for Knowledge Analysis) is a collection of different models, techniques and algorithms for machine learning and data mining, developed at the University of Waikato, New Zeeland (Witten and Frank, 2005). WEKA can be used for machine learning in a number of different contexts, not just in linguistics. It is also possible for the user to add and try out their own machine learning models and algorithms.

WEKA can be run using a command line interface, but also has a graphical interface, consisting of three parts: Explorer, Experimenter and Knowledge Flow. In the Explorer the machine learning algorithms are run, and there are also tools for preprocessing data (discretisation, normalisation, etc.) and the possibility to build your own classifiers. In the Experimenter, which is the part of WEKA I will mainly be using, you can compare different machine learning algorithms and see how well they perform on the same data. Knowledge Flow is a visual interface where you can build your own algorithms graphically by combining different components (data, filters, classifiers, etc.).

WEKA's input by default is flat ARFF-files (Attribute-Relation File Format), which basically is a list of comma-separated values with one instance per row, but WEKA can also import data from a number of other formats, such as normal comma-separated value lists and a number of different database formats.

The machine learning algorithms in WEKA can be applied to a number of fields in language technology. WEKA also contains tools for preprocessing, classification, regression, clustering, association and visualisation of data. It is a good collection of machine learning tools, and a good way to try out and understand different machine learning techniques applicable to linguistic data. Being so comprehensive makes for quite a steep learning curve, but once you have understood its basic workings, WEKA is a powerful tool.

WEKA is written in Java, which makes it platform independent and able to be built into the user's own Java (or, for that matter, other language) programs. Weka is open source software issued under the GNU General Public License.[1]

I have used WEKA version 3.5.6, the latest as of this writing.

[1]WEKA was recently (september 2006) bought by Pentaho Corp, an American company specialising in "Open Source Business Intelligence". Hopefully WEKA will remain open and free in the future...

3 Method

To test Reality we must see it on the tight-rope.
When the Verities become acrobats, we can judge them.
Oscar Wilde, *The Picture of Dorian Gray*

3.1 Feature selection

The analysis part of the process consisted of training machine learning algorithms on features extracted from the texts in the processed corpus, to find successful models for classification into granted and refused applications.

I used 6,000 granted and 6,000 refused applications taken at random from the corpus (section 2.2) and the following features, taken from Melin and Lange (2000) and Platzack (1974):

Sentences The number of sentences in a document.

Words The number of words in a document.

Letters The average word length in letters.

PPS The number of prepositional phrases in a document.

NPS The number of noun phrases in a document.

NQ Nominal Quotient, a measure of the information content of a document, measured as follows:

$$NQ = \frac{nouns + participles + prepositions}{verbs + pronouns + adverbs}$$

LIX Readability index for Swedish (LäsbarhetsIndeX), a measure of the complexity of a document, measured as follows:

$$LIX = words/sentences + (100 * long\ words/words)$$

where "long words" are words with more than 6 letters

3.2 Classifiers

I used WEKA (section 2.5) with six different machine learning algorithms to try to classify the corpus into granted and refused applications using the different features from section 3.1.

The classifiers I used were chosen to give a varied mix of classifiers using different techniques (Witten and Frank, 2005). Descriptions of the classifiers used follow below.

ZeroR (Zero Rules) Predicts the test data's average value. The accuracy with two classes (yes/no) is always 50%. This is used as the baseline against which the other classifiers are evaluated.

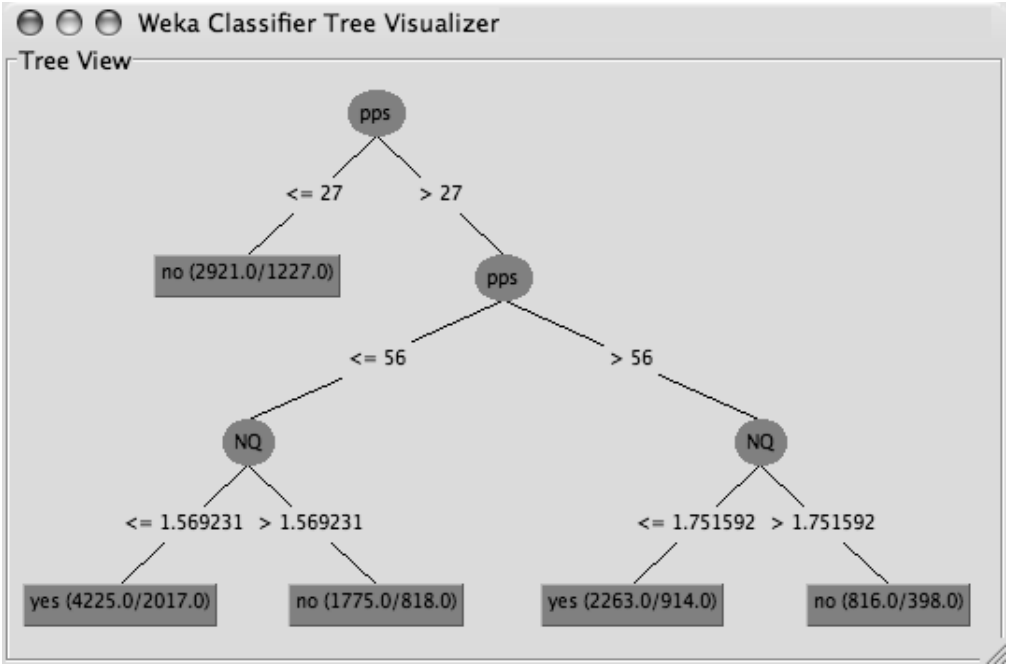
DTable (Decision Table) Builds a simple decision table majority classifier. For an example of the type of table used by the classifier, see Table 2. The Decision Table classifier evaluates feature subsets using best-first search and can use cross-validation for evaluation (Kohavi, 1995).

Table 2: An example of rules from a decision table classifier

Sentences	Letters	PPS	NPS	Granted
'(10.5-20.5]'	'(3094.5-inf)'	'(56.5-inf)'	'(216.5-inf)'	yes
'(20.5-inf)'	'(3094.5-inf)'	'(56.5-inf)'	'(216.5-inf)'	yes
'(20.5-inf)'	'(1571.5-3094.5]'	'(56.5-inf)'	'(216.5-inf)'	no
'(10.5-20.5]'	'(3094.5-inf)'	'(27.5-56.5]'	'(216.5-inf)'	no
'(20.5-inf)'	'(3094.5-inf)'	'(27.5-56.5]'	'(216.5-inf)'	yes
'(-inf-10.5]'	'(3094.5-inf)'	'(56.5-inf)'	'(95.5-216.5]'	yes
'(10.5-20.5]'	'(3094.5-inf)'	'(56.5-inf)'	'(95.5-216.5]'	yes
'(20.5-inf)'	'(1571.5-3094.5]'	'(27.5-56.5]'	'(216.5-inf)'	no
...				

ADTree (Alternating Decision Tree) A decision tree is an arrangement of tests that prescribe an appropriate test at every step in an analysis. Nodes in a decision tree involve testing a particular attribute. An unknown instance is classified by starting at the root (the top, as a decision tree is a tree standing on its head) node and travelling down the tree according to the values of the attributes tested at successive nodes down the tree until one reaches a leaf node, see Figure 2. In the algorithm used, based on Freund and Mason (1999), the induction of the trees has been optimized, and heuristic search methods have been introduced to speed learning.

Figure 2: An example of a decision tree



J48 Another decision tree algorithm, an implementation in Java of the C4.5 algorithm from Quinlan (1993). The algorithm grows the tree by splitting the nodes in a way that maximises information gain, which is defined as the difference of the entropy of the mother node and the weighted sum of the entropies of the child nodes (Manning and Schütze, 1999). The algorithm can be explained in pseudo-code as follows:

```
(1) Check for base cases
(2) For each attribute (a)
        Find the information gain from splitting on (a)
(3) Let (a_best) be the attribute with the highest
    information gain
(4) Create a decision node that splits on (a_best)
(5) Recurse on the sublists obtained by splitting on (a_best)
    and add those nodes as children of the decision node
```

NBTree (Naive Bayes Tree) A hybrid between decision trees and Naive Bayes (see below), that creates trees whose leaves are Naive Bayes classifiers for the instances that reach the leaf. Cross-validation is used to decide whether a node should be split further or a Naive Bayes model should be used instead.

Naive Bayes An implementation of the probabilistic Naive Bayes classifier

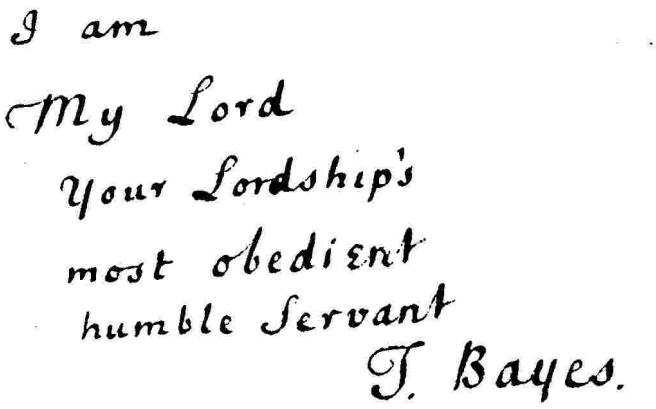
using estimator classes. Numeric estimator precision values are chosen based on analysis of the training data. A Naive Bayes classifier uses Bayes' Theorem (Bayes, 1764):

$$P(A|B) = \frac{P(A) \times P(B|A)}{P(B)}$$

The probability of A occurring given that B has occurred ($P(A|B)$) is the probability of A occurring ($P(A)$) times the probability of B occurring if A has occurred ($P(B|A)$) divided by the probability of B occurring ($P(B)$).

together with the "naive" assumption that the attributes used are conditionally independent, which they usually are not in real life. The Naive Bayes classifier has, however, been shown to do surprisingly well in comparison to other classifiers (Zhang, 2004).

Figure 3: A greeting from Rev. Mr. Bayes



3.3 Evaluation

The features from section 3.1 were compared in WEKA's Experimenter interface (see section 2.5) using the different classifiers from section 3.2. All tests were carried out using a 20-fold cross validation repeated ten times, which means that the data is first split randomly into 20 parts which are each held out from training in turn, and testing is performed on the held-out part. The results from this procedure are then averaged to give the overall result. This procedure is repeated ten times and again the average is used (Witten and Frank, 2005).

In this way the classifiers were evaluated for accuracy, i.e. the proportion of correctly classified instances, defined as

$$accuracy = \frac{a+d}{a+b+c+d}$$

using a contingency table such as Table 3 (Manning and Schütze, 1999).

Table 3: Contingency table for evaluating a binary classifier

	YES is correct	NO is correct
YES was assigned	a	b
NO was assigned	c	d

Using a paired t-test we can also see whether there exists a significant difference between the outcomes of different classifiers on the same data. A significance level of 0.05 (5%) is used and a version of the test known as the corrected resampled t-test (Witten and Frank, 2005).

4 Results and Discussion

Success is a science; if you have the conditions, you get the result.
Oscar Wilde, *Letters*

My first results are shown in Table 4, which shows the results of testing the classifiers on all seven separate features and on the combination of all features taken together (henceforth referred to as the 'All' feature), sorted by the highest average accuracy. The best classifier for each separate feature has the result shown in **bold**.

Table 4: Accuracy (%) for each feature separately in descending order, along with the averages for each feature and for each classifier

Dataset	Dtable	ADTree	J48	NBTree	Bayes	Average
All	53.90 ∘	**54.15** ∘	53.62 ∘	53.81 ∘	53.81 ∘	53.86
PPS	53.41 ∘	54.01 ∘	53.37 ∘	53.41 ∘	**54.22** ∘	53.68
Words	**53.71** ∘	53.67 ∘	**53.71** ∘	**53.71** ∘	53.57 ∘	53.67
Letters	**53.75** ∘	53.61 ∘	53.62 ∘	53.65 ∘	53.53 ∘	53.63
NPS	53.45 ∘	53.11 ∘	53.45 ∘	53.45 ∘	**53.47** ∘	53.39
Sentences	51.71 ∘	52.26 ∘	52.83 ∘	52.00 ∘	**52.85** ∘	52.33
NQ	49.95	**52.43** ∘	51.06 ∘	50.56 ∘	51.45 ∘	51.09
LIX	50.00	50.41	49.78	49.99	**50.96** ∘	50.23
Average	52.48	52.96	52.68	52.57	**52.99**	

∘ = statistically significant improvement from the baseline (50%)

The ZeroR classifier will not be included in these tables as the results are invariably 50% (see section 3.2).

We can see that the feature with the highest average accuracy (53.86% correct) was the 'All' feature, followed by the number of prepositional phrases (PPS, item 3.1 in section 3.1). When looking at the separate results, the two highest ranked features have changed places, with the PPS feature having the highest ranking of all (54.22%), followed by the 'All' feature (54.15%).

The worst single feature was LIX (item 3.1 in section 3.1), which only shows a significant improvement using one classifier out of six, so we can already conclude that LIX (at least in itself) is not a good enough predictor attribute, which could be discarded from future analyses.

Table 5: Accuracy (%) for all combinations of two features in descending order

Dataset	Dtable	ADTree	J48	NBTree	Bayes	Average
Words+NQ	53.69 ∘	53.83 ∘	**54.07** ∘	53.69 ∘	53.92 ∘	53.84
PPS+NQ	53.39 ∘	54.31 ∘	53.53 ∘	53.40 ∘	**54.50** ∘	53.83
Letters+NQ	53.69 ∘	**53.94** ∘	53.86 ∘	53.64 ∘	53.84 ∘	53.79
Letters+LIX	53.75 ∘	53.82 ∘	53.62 ∘	53.65 ∘	**54.06** ∘	53.78
Words+PPS	53.85 ∘	53.78 ∘	53.35 ∘	**53.89** ∘	53.88 ∘	53.75
Sents+Letters	**54.25** ∘	53.78 ∘	53.76 ∘	53.53 ∘	53.43 ∘	53.75
Words+LIX	53.71 ∘	53.81 ∘	53.25 ∘	53.71 ∘	**53.95** ∘	53.69
Sents+Words	53.76 ∘	**53.87** ∘	53.73 ∘	53.64 ∘	53.34 ∘	53.67
PPS+NPS	53.68 ∘	53.50 ∘	53.36 ∘	53.69 ∘	**53.88** ∘	53.62
NPS+NQ	53.41 ∘	53.59 ∘	53.60 ∘	53.43 ∘	**53.98** ∘	53.60
PPS+LIX	53.41 ∘	53.63 ∘	53.37 ∘	53.41 ∘	**54.13** ∘	53.59
NPS+LIX	53.45 ∘	53.66 ∘	53.38 ∘	53.45 ∘	**54.00** ∘	53.59
Words+NPS	53.64 ∘	53.35 ∘	53.57 ∘	**53.70** ∘	53.51 ∘	53.55
Sents+PPS	53.68 ∘	**53.71** ∘	53.18 ∘	53.52 ∘	53.60 ∘	53.54
Sents+NPS	53.40 ∘	**53.77** ∘	53.67 ∘	53.46 ∘	53.36 ∘	53.53
Words+Letters	53.38 ∘	**53.57** ∘	53.39 ∘	53.35 ∘	53.54 ∘	53.45
Letters+PPS	53.67 ∘	53.06 ∘	53.26 ∘	53.35 ∘	**53.77** ∘	53.42
Letters+NPS	53.35 ∘	53.30 ∘	53.48 ∘	53.38 ∘	**53.55** ∘	53.41
Sents+LIX	51.71 ∘	52.77 ∘	52.67 ∘	52.00 ∘	**52.97** ∘	52.42
Sents+NQ	51.67 ∘	52.47 ∘	52.32 ∘	51.99 ∘	**53.28** ∘	52.35
NQ+LIX	49.95	51.79 ∘	51.56 ∘	50.46	**52.31** ∘	51.21
Average	53.26	53.49	53.33	53.26	**53.66**	

∘ = statistically significant improvement from the baseline (50%)

Table 6: Accuracy (%) for the best ten features or two-feature combinations in descending order

Dataset	Dtable	ADTree	J48	NBTree	Bayes	Average
All	53.90 ∘	**54.15** ∘	53.62 ∘	53.81 ∘	53.81 ∘	53.86
Words+NQ	53.69 ∘	53.83 ∘	**54.07** ∘	53.69 ∘	53.92 ∘	53.84
PPS+NQ	53.39 ∘	54.31 ∘	53.53 ∘	53.40 ∘	**54.50** ∘	53.83
Letters+NQ	53.69 ∘	**53.94** ∘	53.86 ∘	53.64 ∘	53.84 ∘	53.79
Letters+LIX	53.75 ∘	53.82 ∘	53.62 ∘	53.65 ∘	**54.06** ∘	53.78
Words+PPS	53.85 ∘	53.78 ∘	53.35 ∘	**53.89** ∘	53.88 ∘	53.75
Sents+Letters	**54.25** ∘	53.78 ∘	53.76 ∘	53.53 ∘	53.43 ∘	53.75
Words+LIX	53.71 ∘	53.81 ∘	53.25 ∘	53.71 ∘	**53.95** ∘	53.69
PPS	53.41 ∘	54.01 ∘	53.37 ∘	53.41 ∘	**54.22** ∘	53.68
Sents+Words	53.76 ∘	**53.87** ∘	53.73 ∘	53.64 ∘	53.34 ∘	53.67
Average (top ten)	53.74	53.93	53.62	53.64	**53.90**	

∘ = statistically significant improvement from the baseline (50%)

However, it could still be the case that the LIX feature taken together with some other feature could yield good results, especially as the 'All' feature comes in first in the averages. So for that reason I have also performed tests with combinations of two features. Table 5 shows the results for all combinations of two features in descending order by classifier average.

In Table 5 we can see that combinations with LIX do get fairly high results, and that the best LIX combination scores better even than the PPS + Words combination, which Table 4 could lead one to believe would be one of the highest scorers. The combination of the two worst single features from Table 4, LIX and NQ, does however get the bottom place as a combination as well.

In Table 6, which shows the features or two-feature combinations with the best ten average accuracies, we can see that the 'All' feature from Table 4 still has the highest ranking.

The above tests have shown the average results when combining all classifiers. For the rest of the tests I have chosen just one classifier to test the rest of the combinations, as similar tests to the ones above would be prohibitively resource-heavy. I have chosen the Naive Bayes classifier as it was the one with the best average results in Tables 4 and 5 (see Table 7). Research (Zhang, 2004) has also shown that Naive Bayes is a good classifier for these types of tasks. These tests were also carried out using a 20-fold cross validation.

Table 7: Classifiers sorted by average accuracy using the averages from Tables 4 and 5

Classifier	Accuracy (%)
Naive Bayes	53.32
Alternating Decision Tree	53.23
J48	53.01
Naive Bayes Tree	52.92
Decision Table	52.87

Tables 8, 9 and 10 show the best ten combinations of three, four and five features respectively, and Table 11 shows the results for all combinations of six features, i.e. all features minus one, in descending order by accuracy.

Table 8: Accuracy for the best ten combinations of three features using the Naive Bayes classifier (for all results see Table 13)

Features	Accuracy (%)
PPS+NQ+LIX	54.46
NPS+NQ+LIX	54.37
Words+NQ+LIX	54.36
Letters+NQ+LIX	54.32
PPS+NPS+NQ	54.27
Words+PPS+LIX	54.21
PPS+NPS+LIX	54.13
Words+PPS+NQ	54.08
Letters+PPS+NQ	53.98
Letters+PPS+LIX	53.98

As we can see in Table 8 the features NQ and LIX together with another feature score the highest among the combinations of three features, although the combination of just NQ and LIX score the lowest of all combinations in Table 5.

Table 9: Accuracy for the best ten combinations of four features using the Naive Bayes classifier (for all results see Table 14)

Features	Accuracy (%)
Words+PPS+NQ+LIX	54.37
PPS+NPS+NQ+LIX	54.26
Sentences+PPS+NQ+LIX	54.10
Words+NPS+NQ+LIX	54.03
Letters+NPS+NQ+LIX	54.03
Words+PPS+NPS+NQ	54.00
Sentences+PPS+NPS+LIX	53.98
Words+PPS+NPS+LIX	53.98
Letters+PPS+NPS+NQ	53.95
Sentences+Words+PPS+LIX	53.93

In Table 9 we can see that the best three four-feature combinations all have the features PPS, NQ and LIX in common, and that the NQ+LIX combination is prominent in the top ten for five-feature combinations as well (Table 10).

Table 10: Accuracy for the best ten combinations of five features using the Naive Bayes classifier (for all results see Table 15)

Features	Accuracy (%)
Words+PPS+NPS+NQ+LIX	54.09
Words+Letters+PPS+NQ+LIX	54.05
Words+Letters+NPS+NQ+LIX	53.96
Sentences+PPS+NPS+NQ+LIX	53.93
Words+Letters+PPS+NPS+LIX	53.88
Sentences+Words+PPS+NPS+LIX	53.87
Letters+PPS+NPS+NQ+LIX	53.84
Sentences+Letters+PPS+NPS+LIX	53.81
Words+Letters+PPS+NPS+NQ	53.80
Sentences+Words+PPS+NQ+LIX	53.79

Table 11 shows that the features PPS, NQ and LIX have an impact on the scoring here as well, as the three worst combinations are the ones where these features are removed.

Table 11: Accuracy for all combinations of six features (all minus one) using the Naive Bayes classifier

Features	Accuracy (%)
All−Sentences	53.89
All−Letters	53.82
All−NPS	53.74
All−Words	53.74
All−NQ	53.67
All−PPS	53.67
All−LIX	53.66

Finally, Table 12 shows the best ten combinations of features using the Naive Bayes classifier. We can see that among the top features are, again, PPS, NQ and LIX in various combinations.

Table 12: Accuracy for the best twenty combinations of features using the Naive Bayes classifier

Features	Accuracy (%)
PPS+NQ	54.50
PPS+NQ+LIX	54.46
NPS+NQ+LIX	54.37
Words+PPS+NQ+LIX	54.37
Words+NQ+LIX	54.36
Letters+NQ+LIX	54.32
PPS+NPS+NQ	54.27
PPS+NPS+NQ+LIX	54.26
PPS	54.22
Words+PPS+LIX	54.21
PPS+LIX	54.13
PPS+NPS+LIX	54.13
Sentences+PPS+NQ+LIX	54.10
Words+PPS+NPS+NQ+LIX	54.09
Words+PPS+NQ	54.08
Letters+LIX	54.06
Words+Letters+PPS+NQ+LIX	54.05
Words+NPS+NQ+LIX	54.03
Letters+NPS+NQ+LIX	54.03
NPS+LIX	54.00

Table 13: Accuracy for all combinations of three features using the Naive Bayes classifier

Features	Accuracy (%)
Sentences+Words+Letters	53.47
Sentences+Words+PPS	53.67
Sentences+Words+NPS	53.52
Sentences+Words+NQ	53.62
Senteces+Words+LIX	53.63
Sentences+Letters+PPS	53.38
Sentences+Letters+NPS	53.51
Sentences+Letters+NQ	53.48
Sentences+Letters+LIX	53.39
Sentences+PPS+NPS	53.63
Sentences+PPS+NQ	53.93
Sentences+PPS+LIX	53.64
Sentences+NPS+NQ	53.60
Sentences+NPS+LIX	53.68
Sentences+NQ+LIX	53.76
Words+Letters+PPS	53.79
Words+Letters+NPS	53.55
Words+Letters+NQ	53.64
Words+Letters+LIX	53.74
Words+PPS+NPS	53.78
Words+PPS+NQ	54.08
Words+PPS+LIX	54.21
Words+NPS+NQ	53.55
Words+NPS+LIX	53.80
Words+NQ+LIX	54.36
Letters+PPS+NPS	53.85
Letters+PPS+NQ	53.98
Letters+PPS+LIX	53.98
Letters+NPS+NQ	53.53
Letters+NPS+LIX	53.66
Letters+NQ+LIX	54.32
PPS+NPS+NQ	54.27
PPS+NPS+LIX	54.13
PPS+NQ+LIX	54.46
NPS+NQ+LIX	54.37

Table 14: Accuracy for all combinations of four features using the Naive Bayes classifier

Features	Accuracy (%)
Sentences+Words+Letters+PPS	53.47
Sentences+Words+Letters+NPS	53.54
Sentences+Words+Letters+NQ	53.42
Sentences+Words+Letters+LIX	53.59
Sentences+Words+PPS+NPS	53.59
Sentences+Words+PPS+NQ	53.83
Sentences+Words+PPS+LIX	53.93
Sentences+Words+NPS+NQ	53.44
Sentences+Words+NPS+LIX	53.68
Sentences+Words+NQ+LIX	53.51
Sentences+Letters+PPS+NPS	53.46
Sentences+Letters+PPS+NQ	53.61
Sentences+Letters+PPS+LIX	53.80
Sentences+Letters+NPS+NQ	53.54
Sentences+Letters+NPS+LIX	53.59
Sentences+Letters+NQ+LIX	53.49
Sentences+PPS+NPS+NQ	53.88
Sentences+PPS+NPS+LIX	53.98
Sentences+PPS+NQ+LIX	54.10
Sentences+NPS+NQ+LIX	53.62
Words+Letters+PPS+NPS	53.66
Words+Letters+PPS+NQ	53.81
Words+Letters+PPS+LIX	53.90
Words+Letters+NPS+NQ	53.62
Words+Letters+NPS+LIX	53.80
Words+Letters+NQ+LIX	53.78
Words+PPS+NPS+NQ	54.00
Words+PPS+NPS+LIX	53.98
Words+PPS+NQ+LIX	54.37
Words+NPS+NQ+LIX	54.03
Letters+PPS+NPS+NQ	53.95
Letters+PPS+NPS+LIX	53.90
Letters+NPS+NQ+LIX	54.03
PPS+NPS+NQ+LIX	54.26

Table 15: Accuracy for all combinations of five features using the Naive Bayes classifier

Features	Accuracy (%)
Sentences+Words+Letters+PPS+NPS	53.52
Sentences+Words+Letters+PPS+NQ	53.75
Sentences+Words+Letters+PPS+LIX	53.68
Sentences+Words+Letters+NPS+NQ	53.56
Sentences+Words+Letters+NPS+LIX	53.63
Sentences+Words+Letters+NQ+LIX	53.43
Sentences+Words+PPS+NPS+NQ	53.68
Sentences+Words+PPS+NPS+LIX	53.87
Sentences+Words+PPS+NQ+LIX	53.79
Sentences+Words+NPS+NQ+LIX	53.58
Sentences+Letters+PPS+NPS+NQ	53.66
Sentences+Letters+PPS+NPS+LIX	53.81
Sentences+Letters+PPS+NQ+LIX	53.71
Sentences+Letters+NPS+NQ+LIX	53.54
Sentences+PPS+NPS+NQ+LIX	53.93
Words+Letters+PPS+NPS+NQ	53.80
Words+Letters+PPS+NPS+LIX	53.88
Words+Letters+PPS+NQ+LIX	54.05
Words+Letters+NPS+NQ+LIX	53.96
Words+PPS+NPS+NQ+LIX	54.09
Letters+PPS+NPS+NQ+LIX	53.84

5 Conclusions and Future Work

I can stand brute force, but brute reason is quite unbearable.
Oscar Wilde, *The Picture of Dorian Gray*

The results in section 4 show that I have verified my hypothesis, that it is possible to automatically ascertain linguistic differences between granted and refused grant applications and that these differences are statistically significant.

I have shown that the feature PPS (section 3.1) is the best single predictor for this classification problem (Table 4) and that combinations of the PPS, LIX and NQ features give the best overall results (Table 12). I have also shown that a Naive Bayes classifier (section 3.2) is a suitable classifier for this type of task, and that, using such a classifier, an accuracy of up to 54,5 % can be achieved (Table 12).

For even better results one could try other classifying techniques shown to work well for text categorization, e.g. Support vector machines (Sahlgren and Cöster, 2004) or techniques such as Latent Semantic Analysis (Deerwester et al., 1990; Foltz et al., 1998) and Random Indexing (Sahlgren, 2005). See Sahlgren (2006) for a discussion of the word-space model on which these types of classifier build. These techniques, however, lie outside the scope of this thesis, where the main point was to see if an automatic classification was at all possible and not necessarily to try for the best possible results.

I have also only classified the documents into two categories: granted and non-granted applications. In reality this type of either/or-classification would be the end result of a long process, where applications are weighed and probably graded on a finer scale to begin with. Not having access to material from inside an application process I have no way of classifying into anything else than granted/non-granted applications. If such material were available, i.e. if I had access to applications graded on a scale between yes and no, a project such as mine could probably tell us more about the assessment of grant applications and probably get better results.

A future project of this kind could also be enlarged to use actual grant applications in their entirety to see whether my results still hold true. A reasonable assumption is that the statistical differences between the entire application texts will be much greater than the differences between the abstracts used in this study.

Possible future developments could be devising tools based on these results to help applicants maximize their applications' chances of being granted - or at least minimize the risk for being refused on linguistic (be they explicit or implicit) grounds.

One could also imagine devising tools to help assessors "see beyond" the language of an application, as a grant application with a high "language

score" is not necessarily a good application in any other way than the language used. The same, in reverse, is true for applications written in poor language, which could be flagged for extra scrutiny somewhat along these lines: "Watch out! This application uses subpar language - do not let that cloud your assessment of its scientific merit!"

Language alone is of course not a basis for granting or discarding grant applications, although I have shown that it does play a role. Future research based on the results from my thesis could be of service to both applicants and granting bodies, to increase the awareness of the linguistic factors present in a grant application process - that language cannot be taken for granted.

References

Aycock, J. (1998). Compiling little languages in python. In *Proceedings of the 7th International Python Conference*.

Bayes, T. (1764). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*.

Bird, S., Klein, E., and Loper, E. (2007). *Introduction to Natural Language Processing*. TBA.

Brants, T. (2000). Tnt – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference*, Seattle, Washington.

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Collins-Thompson, K. and Callan, J. (2005). Predicting reading difficulty with statistical language models. *Journal of the American Society for Information Science and Technology*, 56(13):1448–1462.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Foltz, P. W., Kintsch, W., and Landauer, T. K. (1998). The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25(2&3):285–307.

Freund, Y. and Mason, L. (1999). The alternating decision tree learning algorithm. In *Proceedings of the 16th International Conference on Machine Learning*, pages 124–133, San Francisco, CA.

Kohavi, R. (1995). The power of decision tables. In Lavrac, N. and Wrobel, S., editors, *Proceedings of the Eighth European Conference on Machine Learning*, pages 174–189, Berlin, Germany.

Landauer, T. K., Laham, D., and Foltz, P. (2003). Automatic essay assessment. *Assessment in Education*, 10(3).

Larkey, L. S. (1998). Automatic essay grading using text categorization techniques. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development on Machine Learning*, pages 90–95.

Larsson, P. (2006). Classification into readability levels: implementation and evaluation. Master's thesis, Department of Linguistics and Philology, Uppsala Universitet.

Liu, X., Croft, B., Oh, P., and Hart, D. (2004). Automatic recognition of reading levels from user queries. In *SIGIR'04*.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts.

Megyesi, B. (2002). *Data-Driven Syntactic Analysis - Methods and Applications for Swedish*. PhD thesis, Department of Speech, Music and Hearing, KTH, Stockholm, Sweden.

Melin, L. and Lange, S. (2000). *Att analysera text: stilanalys med exempel*. Studentlitteratur, Lund, 3 edition.

Nordman, M. (1992). *Svenskt fackspråk*. Studentlitteratur.

Platzack, C. (1974). *Språket och läsbarheten*. PhD thesis, Lund University, Sweden.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA.

Sahlgren, M. (2005). An introduction to random indexing. In *Proceedings of the Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005*, Copenhagen, Denmark.

Sahlgren, M. (2006). *The Word-Space Model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. PhD thesis, Department of Linguistics, Stockholm University, Sweden.

Sahlgren, M. and Cöster, R. (2004). Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *Proceedings of COLING 2004*, Geneva.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Sebastiani, F. (2005). Text categorization. In Zanasi, A., editor, *Text Mining and its Applications to Intelligence, CRM and Knowledge Management*, pages 109–129. WIT Press, Southampton, UK.

Sjöbergh, J. (2006). *Language Technology for the Lazy - Avoiding Work by Using Statistics and Machine Learning*. PhD thesis, School of Computer Science and Communication, KTH, Stockholm, Sweden.

SUC (1997). Suc. department of linguistics umeå university, and department of linguistics, stockholm university. 1997. suc 1.0 stockholm umeå corpus, version 1.0. isbn: 91-7191-348-3.

Svensson, J. (1993). *Språk och offentlighet*. Number A:47 in Lundastudier i nordisk språkvetenskap. Lund University Press.

Witten, I. H. and Frank, E. (2005). *Data Mining: practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, CA, 2 edition.

Zhang, H. (2004). The optimality of naive bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, Miami Beach, Florida, USA.