Accurate Stemming of Dutch for Text Classification

Tanja Gaustad and Gosse Bouma

Alfa-Informatica
Rijksuniversiteit Groningen

Abstract

This paper investigates the use of stemming for classification of Dutch (email) texts. We introduce a new stemmer, which combines dictionary lookup (implemented efficiently as a finite state automaton) with a rule-based backup strategy and show that it outperforms the Dutch Porter stemmer in terms of accuracy, while not being substantially slower.

For text classification, the most important property of a stemmer is the number of words it (correctly) reduces to the same stem. Here the dictionary-based system also outperfoms Porter. However, evaluation of a Bayesian text classification system with either no stemming or the Porter or dictionary-based stemmer on an email classification and a newspaper topic classification task does not lead to significant differences in accuracy. We conclude with an analysis of why this is the case.

1 Introduction

Responding adequately to email messages can become a time-consuming and expensive task for large organizations, which often receive massive numbers of emails. The research reported on below was carried out for a call centre, which receives a few hundred emails a day for one of their clients, an internet provider. The challenge is to answer email quickly (within 24 hours) and correctly. One method to support efficient processing of email is the use of a text classification system, which labels incoming email with one or more likely answer categories. The task of the agent responding to the email is to check whether the correct category is among the selected categories. If so, the corresponding answer text can be inserted automatically. An accurate classification system can help improve the efficiency of agents by almost a factor of two (Busemann, Schmeier and Arens 2000).

Statistical text classification systems compute the most likely class for a text by computing how likely the words and n-grams in the text are for any given class. Estimating these probabilities is difficult, as texts contain lots of different, often infrequent, words. One way to deal with this problem is to take into consideration only words which occurred at least n times in a given training corpus, as estimation is more reliable for frequently occurring words. Stemming is another method which can be used to reduce the number of word forms that need to be taken into consideration. Stemming reduces all inflected forms of a word to the same stem. The number of different stems in a text or training corpus will therefore in general be expected to be much smaller than the number of different word forms, and the frequency of stems will therefore be higher than that of the corresponding individual inflected forms, which in turn suggests that probabilities can be estimated more reliably.

Stemming is a well-known technique in Information Retrieval (IR) systems where the main goal is to retrieve the documents that correspond to a given query. Just as in classification tasks, stemming was conceived as a way of reducing morphological variants to a single (indexing) term. Experiments to determine the effectiveness of stemming have produced mixed results. One important factor is the language of the documents involved. Harman (1991) compared the performance of data stemmed with three suffix-stripping algorithms for English against unstemmed data in IR queries and came to the conclusion that stemming does not consistently improve performance. Krovetz (1993), on the other hand, concludes that accurate stemming of English does improve performance of an IR system. Popovič and Willett (1992) investigated whether stemming would have more effect for a morphologically complex language like Slovene. They found that precision of the retrieved documents was increased when suffix-stripping was used. Dutch is a language which is morphologically more complex than English, but not as complex as the Slavic languages. Kraaij and Pohlman (1996) found that both a stemmer using their adaptation of the Porter algorithm for Dutch (a well-known suffix-stripping algortithm) and a dictionary-based stemmer led to a decrease in IR-performance compared to using no stemming.

Recently, stemming has also been applied to text classification. Just as in IR, experiments lead to mixed results. On the basis of her experiments for English text classification, Riloff (1995) concludes that "stemming algorithms may be appropriate for some terms but not for others" and that classification systems would benefit from using all available information, including morphological variants. Busemann et al. (2000), on the other hand, have shown that morphological analysis increases performance for a series of classification algorithms applied to German email classification. Spitters (2000) compares, among others, the performance of two machine learning algorithms for topic classification of Dutch newspaper articles, using both unstemmed text and text stemmed with the Dutch Porter stemmer. He concludes that stemming does not improve the performance of either algorithm.

In this paper, we further investigate whether stemming improves accuracy of Dutch text classification by taking into consideration both the Dutch Porter stemmer (Kraaij and Pohlman 1994) as well as a more accurate dictionary-based stemmer, and by providing results for two radically different text genres, namely email messages and newspaper text. We show that dictionary-based stemming leads to a reduction in stemming error-rate of almost 90% and also leads to a significantly higher reduction of the number of features that need to be considered in classification. Nevertheless, our conclusions are similar to those given in Kraaij and Pohlman (1994) for IR and in Spitters (2000) for text categorization: neither simple suffix-stripping nor dictionary-based stemming leads to improved classification accuracy for Dutch text. We provide some potential explanations for why this is the case.

In section 2 we introduce the two stemmers used and evaluate them in terms of accuracy and speed on a corpus of manually annotated text. In section 3, Bayesian text classification is introduced and the email and newspaper topic classification tasks for Dutch are described. Classification results on unstemmed and stemmed text for a range of experiments are given. A qualitative evaluation of the results concludes the paper.

2 Stemmers

A stemmer tries to reduce various forms of a word to a single stem. For Dutch, for instance, a stemmer might reduce various forms of the verb *schrijven* (*to write*) such as *schrijf, schrijft, schrijven, schreef, schreven*, and *geschreven* to the stem *schrijv*. Stemming in general requires that inflected word forms are reduced to a stem. A simple and robust method works by removing only certain inflectional suffixes and undoing the effect of certain orthographical rules (i.e. the letter *-f* in the coda of a Dutch word sometimes corresponds to a *-v* in the stem). More accurate methods, able to deal with irregular morphology (like the fact that the past tense singular form of the stem *schrijv* is *schreef*), require a dictionary.

Below, we first briefly describe the Dutch Porter stemmer of Kraaij and Pohlman (1994). Next, we present a new stemmer for Dutch, with dictionary lookup and a rule-based backup strategy. Finally, we present some experimental results which confirm that the new stemmer is far more accurate than the Porter stemmer, while not being substantially slower.

2.1 Dutch Porter Stemmer (DPS)

The Porter stemmer (Porter 1980) is a rule-based suffix stripper which is widely used in IR systems. Porter's algorithm implements a series of steps which each remove a certain type of suffix by way of substitution rules. These rules only apply when certain conditions hold, e.g. the resulting stem must have a certain minimal length. Kraaij and Pohlman (1994) developed a Porter stemmer for Dutch[1] which uses the implementation presented in (Frakes and Baeza-Yates 1992). It removes plural *-en* and *-s* suffixes, the verbal *-t* suffix, dimunitive inflection (realized by various suffixes ending in *-je*), a number of common derivational morphemes, and undoes the effect of the spelling rule which requires consonant doubling in certain contexts.

The advantages of this simple suffix stripper are that it is very robust and the implementation is fairly easy. It is also clear that it will often produce the wrong stem for a word. I.e. the derivational suffix *-ing* can be used to form nouns from verbal stems (i.e. *regering* (*government*) from *regeer* (*govern*). However, simply stripping the suffix *-ing* from all words matching words also produces for the verb *afdwing* (*force*) the nonsense form *afdw*. Such mistakes need not be fatal: as long as words are reduced to a unique stem form, no information is lost. Potentially

[1]Available at `http://www-uilots.let.uu.nl/~uplift/`.

harmful mistakes (known as *overstemming*) occur when a word is reduced to a semantically unrelated stem. For instance, the noun *gulden* ends in *-en*, which is a plural suffix, and therefore is reduced to *gul*, which is an adjective meaning *generous*.

Another weak spot of the algorithm is that is has no way to handle irregular forms. Dutch has a large number of so-called *strong* verbs, whose past and participle forms have root vowels which differ from that in the present tense root (i.e. present tense *nemen* (*to take*) has a past tense *namen* and a participle *genomen*). Such forms will not be reduced to the same stem, a mistake known as *understemming*. The most frequent verb forms tend to be strong, and thus they are an important source of understemming. A riguorous evaluation of the Dutch Porter stemmer, reporting overstemming, understemming, and IR performance, can be found in (Kraaij and Pohlman 1995).

2.2 Stemmer with Dictionary Lookup (SteDL)

It is obvious that including dictionary information will have a positive effect on the accuracy of a stemmer. The linguistically correct form will be provided more often, which might be useful for some applications, and also the percentage of overstemming and understemming errors are likely to go down. The latter should have a positive effect in applications like IR or text classification.

In order to test whether a linguistically more accurate stemmer would perform better than a suffix stripper, we used various existing resources to develop a new stemmer with dictionary lookup.

Dictionary information is obtained from Celex (Baayen, Piepenbrock and van Rijn 1993). Celex contains 381,292 wordforms and 124,136 stems for Dutch. Also, it contains information about the frequency of word forms. This information is useful for disambiguation: in those cases where a word form is listed with two different stems, the most frequent stem can be chosen. In an initialisation step, information about wordforms, their respective stem as well as frequency is extracted from the database.

Dictionary lookup can be time consuming, especially for large dictionaries such as Celex. The extracted lexical information is therefore stored as a finite state automaton, using Daciuk's (2000) finite state automata (FSA) morphology tools.[2] Given a word form, the compiled automaton provides the corresponding stems in time linear to the length of the input word. As a backup strategy for words which are not found in the dictionary, we use the Dutch Porter stemmer (DPS) described above.

The actual stemming procedure is shown in figure 1. The FSA encoding of the information in Celex assigns every wordform all its possible stems. For ambiguous forms, the most frequent stem is chosen. All words that were not found in Celex are processed with DPS.

[2]Available at `http://www.pg.gda.pl/~jandac/fsa.html`.

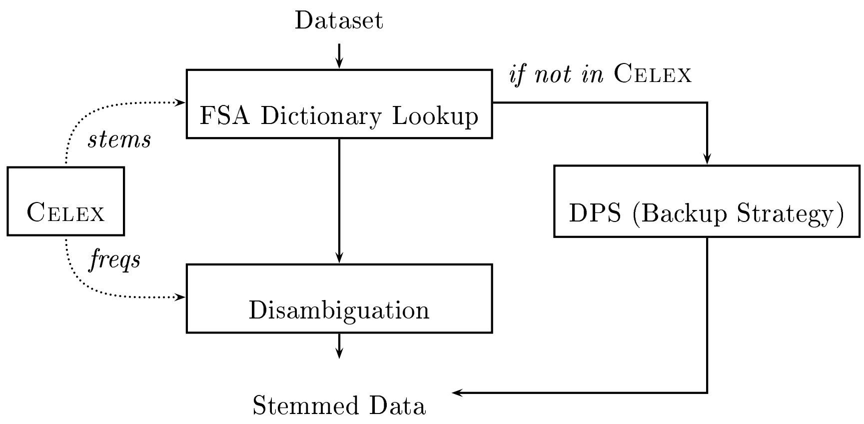

Figure 1: Diagram of the new stemmer with dictionary lookup (SteDL)

2.3 Evaluation

In order to be able to assess the contribution of stemming in text classification, it is crucial to compare the performance of the DPS and the new SteDL independently of a specific application first. For this purpose, we manually prepared a corpus with a stemmed gold standard. The corpus consisted of texts from Dutch children's books and contained ca. 45,000 words.[3]

Stemming accuracy on the test corpus was 98.23% for SteDL and 79.23% for DPS. One has to bear in mind, however, that DPS also strips derivational suffixes whereas in the gold standard these were retained. We estimate that approximately 4-5% of the difference in accuracy is due to the removal of derivational suffixes. Even when taking these differences into account, the dictionary-based stemmer still clearly outperforms the rule-based stemmer in terms of accuracy.

The contributions of various components of SteDL can also be evaluated. DPS was used as a backup strategy in only 2.98% (1,339 out of 44,905) of the cases, which means that the lexical coverage of Celex for the evaluation corpus is fairly good. We also compared a SteDL with a version of the system where a random stem instead of the most frequent form was chosen. This led to a stemming accuracy of 96.27%. Thus, including frequency information reduces the error rate with approximately 50%.

Finally, SteDL is not substantially slower than DPS. After having built the dictionary-FSA (which only needs to be done once during initialisation), the stemming process on the 45,000 word evaluation corpus takes 14 seconds with SteDL, whereas it takes 5 seconds with DPS. The dictionary lookup FSA itself is very fast (0.5 seconds), but the scripts making up the complete system have not been opti-

[3]This corpus is half of the training data from the Dutch Senseval 2 word sense disambiguation task, available at `http://www.sle.sharp.co.uk/senseval2/`.

mised for speed, which explains the difference in time. An obvious improvement would be to integrate dictionary lookup and disambiguation into one FSA. Table 1 summarises the comparison regarding accuracy and speed.

Stemmer	Accuracy
DPS	79.23%
SteDL (no frequency info)	96.27%
SteDL	98.23%

Table 1: Accuracy on 45,000 word evaluation corpus

It is not surprising that a stemmer with dictionary lookup outperforms a rule-based suffix stripper in terms of accuracy. For text categorization, however, accuracy and the ability to reduce related word forms to a single stem, is of importance. This aspect of the performance of the two stemmers is adressed in the application specific evaluation of stemming in the next section.

3 Stemming for Text Classification

In this section, we introduce a Bayesian text classifier, which uses unigram and bigram statistics for classification. Accuracy is improved by taking into account only those unigrams or bigrams which occur at least n times and/or considering only the m most informative words according to information gain. The classifier is tested on an email message classification task and on a newspaper topic classification task. Especially, the effect of stemming text by the DPS or SteDL respectively is investigated. While the number of different unigrams or bigrams is reduced by both algorithms, no consistent effect on classification accuracy could be found.

3.1 Bayesian Text Classification

In the experiments reported below, we chose for a Naive Bayes classification algorithm. Naive Bayes is a simple yet effective statistical classification method which is widely used for text and email classification (e.g. Androutsopoulos et al. 2000).

Naive Bayes estimates the probability that a given document containing features $f_1..f_n$ (where features are usually unigrams or bigrams) belongs to class $c \in C$ by means of the following formula:

$$P(c|f_1...f_n) = \frac{P(c) \cdot \prod_{i=1}^{n} P(f_i|c)}{\sum_{k \in C} P(k) \cdot \prod_{i=1}^{n} P(f_i|k)}$$

The formula is naive in that it assumes that features (words) are independent, an assumption which is almost certainly false in practice. The advantage of using this assumption, however, is that estimating $P(f_i)$ and $P(f_i|c)$, where features range over unigrams or bigrams, is relatively straightforward, given a reasonable sized training corpus consisting of classified documents. The probability $P(c)$ is also estimated on the basis of the training corpus. Classification of a document containing features $f_1..f_n$ now amounts to assigning it the most likely class c according to the formula above.

For the experiments reported below we used the RAINBOW front-end to the software-package BOW.[4] It provides a collection of routines for building statistical classification models (naive Bayes, maximum entropy, k-nearest neighbours, etc.). Furthermore, it offers a wide range of additional techniques which have proved to be useful in automatic text classification: preprocessing (various tokenization options, filtering of stopwords or html-codes, etc.), and various smoothing and feature selection techniques.

Feature selection turned out to be especially important, particularly when using bigrams. The number of different bigram features for a given document collection can be extremely large, with many of them occurring only once. Estimating the probability of features which occur so sparsely is impossible. To circumvent this problem, the features included in the model can be restricted by a frequency cut-off (i.e. include only those features which occur at least n times). Another method includes only those features which are good at discriminating between classes. A metric for finding such features is *information gain*. In the experiments below, this metric is also used to select the m most discriminating features. All results were computed on 90% training and 10% testing material using tenfold cross validation.

3.2 Email Message Classification

In the experiments described in this section, we used a helpdesk email data set. This data set contains questions asked by users of a free internet provider. 6,000 emails have been classified in no less than 193 classes with standard answers by the (human) agents responsible for replying to email.[5] The average length of the emails is 79 words.

One of the major problems with the given hierarchy is that it contains too many classes for accurate classification. A few classes cover a large portion of the data, while most classes contain very few emails. Another problem is the fact that there can be considerable overlap in subject between classes (there are almost 100 standard answers related to functionality and technical issues, and as many as 27 answers related to email alone).

To circumvent using empty or near-empty classes, we decided to select only classes which contain at least 12 emails. This way, we retain 69 categories. This covers 92.5% of all classified emails. See table 2 for an overview. We also investigated what the effect on accuracy would be if we restricted the data to those classes containing at least 25, 50, or 100 emails (see table 3).

[4] Available at `http://www.cs.cmu.edu/~mccallum/bow`.

[5] Actually, the dataset consisted of 41,000 emails, but only 15% of these were assigned a standard answer. Also, the list of standard answers contains 293 items, but only 193 of those were actually used.

Total classified emails	5,965
Total categories	293
Non-empty categories	193
Categories >12 emails	69
Total emails in 69 categories	5,519

Table 2: Characteristics of the helpdesk email dataset

Emails per class	Classes	Number of emails	
All	193	5,965	100.0%
>12	69	5,518	92.5%
>25	47	5,140	86.2%
>50	28	4,502	75.5%
>100	17	3,694	61.9%

Table 3: Overview of emails per class

Given the limited amount of training material and the large number of classes, accuracy of the most likely class assigned by the system was expected to be too low for use in practical applications. However, in a call center environment where a human agent eventually chooses the best answer to a particular email, providing a small list of potential answers can be useful. Therefore, we also include results on n-best classification. In this case, we compute how often the correct class is among the n most likely classes assigned by the system.

3.2.1 Results for Email Classification

We investigated how stemming affects accuracy on the email classification task for a range of experiments using the parameters described above. Before looking at the results, it is useful to compare the two stemmers described in section 2 in terms of feature reduction. In order for stemming to be effective for classification, the number of different words or bigrams in the training data should reduce substantially.

As can be seen in table 4, the feature set does become smaller, especially for bigrams. The stemmer with dictionary lookup (SteDL) performs better than the Porter stemmer (DPS). Email messages tend to contain a high percentage of ad hoc constructions, foreign (especially English) words (*account*), spelling mistakes, and vocabulary items that are not contained in CELEX, such as e.g. product/brand names (*freebees*) or compounds specific to the internet related domain (*inbelnummer*). This is reflected by the fact that in the SteDL stemmer 77% of the dataset (containing 560,000 words) was covered by CELEX and 23% was treated by DPS.

Table 5 presents n-best accuracy results for classification on subsets of the data containing at least 12, 25, 50, or 100 emails per class. The baseline is the accu-

	unigrams		bigrams	
Unstemmed	24,568	100.00%	169,870	100.00%
DPS	23,709	96.50%	163,104	96.02%
SteDL	23,347	95.03%	156,407	92.07%

Table 4: Number of different unigrams and bigrams for unstemmed and stemmed versions of the email corpus

racy for a method which always selects the n most likely classes. The parameters for feature selection (using information gain and frequency) were experimentally determined to give the best results.

Stemming does not consistently improve classification accuracy. Also, even though SteDL reduces the feature set more than DPS, SteDL does not consistently outperform DPS in terms of classification accuracy. The differences found are not statistically signifcant.

Emails	Number of classes	n	Unstemmed av. (stderr)	DPS av. (stderr)	SteDL av. (stderr)	Baseline
>12	69	1	**41.87** (0.43)	41.43 (0.36)	40.53 (0.47)	15.93
		3	**68.97** (0.45)	67.88 (0.46)	68.06 (0.63)	26.87
		5	78.35 (0.26)	78.24 (0.37)	**78.60** (0.34)	35.04
>25	47	1	45.43 (0.45)	45.53 (0.57)	**45.54** (0.52)	17.10
		3	**72.22** (0.44)	72.16 (0.62)	72.22 (0.57)	28.85
		5	**82.24** (0.56)	81.07 (0.42)	81.67 (0.50)	37.62
>50	28	1	50.47 (0.69)	**51.09** (0.49)	50.80 (0.66)	19.52
		3	78.60 (0.51)	**78.62** (0.79)	78.04 (0.64)	32.94
		5	87.96 (0.51)	**87.96** (0.40)	87.58 (0.60)	42.96
>100	17	1	**57.94** (0.55)	57.53 (0.61)	57.07 (0.53)	23.80
		3	**87.26** (0.43)	86.45 (0.71)	86.75 (0.52)	40.15
		5	**94.28** (0.45)	93.96 (0.22)	93.82 (0.39)	52.35

Table 5: Accuracy of n-best email classification (in %), using the 2,500 most informative unigrams and bigrams according to information gain and a frequency cut-off of 15.

3.3 Newspaper Topic Categorization

In order to test whether the poor perfomance of stemming was related to specific properties of the email corpus (which is extremely informal, contains frequent spelling errors, and a high percentage of technical or English jargon), classification experiments were repeated for a Dutch newspaper corpus (*de Volkskrant on CD-ROM, 1997*). Newspaper proze is in general grammatical and contains fewer jargon words or foreign words, and thus, one might expect stemming to have more effect in this case. The corpus consisted of 2,649 articles, selected from 5 different categories (economy, sports, arts, national news, international news), with an average length of 356 words.

The performance of the two stemmers in terms of feature reduction is given in table 6. As was the case for the Email dataset, SteDL achieves a higher reduction of the feature space than DPS. Furthermore, the reduction is substantially larger than for the Email dataset (18% versus 5% for unigrams, and 11% versus 8% for bigrams).

	unigrams		bigrams	
Unstemmed	61,721	100.00%	538,439	100.00%
DPS	52,313	84.76%	504,350	93.67%
SteDL	50,850	82.39%	478,928	88.95%

Table 6: Number of different unigrams and bigrams for stemmed and unstemmed versions of the Volkskrant corpus

3.3.1 Results for Topic Classification

Classification results on the newspaper corpus are given in table 7. The best results were obtained using the 25,000 most informative bigrams. As feature reduction was relatively higher for unigrams, we also evaluated the performance for unigrams. Here, feature pruning in general had a negative effect. Even though stemming reduces the feature space even more than for the Email dataset, there is still no clear effect on accuracy. Also, differences between the Porter and dictionary-based stemmer remain small. This result confirms the findings in Spitters (2000).

Settings	Unstemmed av. (stderr)	DPS av. (stderr)	SteDL av. (stderr)	Baseline
uni-/bigrams	88.93 (0.20)	89.35 (0.19)	**89.50** (0.20)	28.10
unigrams	85.93 (0.21)	**86.20** (0.19)	86.18 (0.20)	28.10

Table 7: Classification accuracy for the Volkskrant dataset (in %) using the 25,000 most informative unigrams and bigrams according to information gain and a frequency cut-off of 2 and using only unigrams without feature selection.

4 Qualitative Evaluation

The results presented in the previous section prove that stemming does significantly reduce the number of different word forms in a text, but that this does not have a significant or consistent effect on classification accuracy.

A potential explanation could be that although the overall number of different features is reduced, the n best features according to information gain do not change substantially. We therefore looked at the top 100 bi- and unigrams with respect to their information gain value to find evidence of how many words are stemmed and in how far stemming actually leads to the conflation of different wordforms into a single stem. We expected stems originating from conflated wordforms to raise in rank.

Only 3 out of the top 100 uni- and bigrams of the unstemmed emails and only 4 out of the top 100 uni- and bigrams of the unstemmed newspaper texts are actually conflated into one stem (see tables 8 and **??**).

Of course, some of the most informative words may have been conflated with words outside the top 100. However, we found that in general there is quite a high correlation between the rank of a word in the list of unstemmed uni- and bigrams, and the rank of the corresponding stem in the stemmed uni- and bigrams, and thus the effect of such conflations seems to be small.

It seems, therefore, that stemming does not reduce enough (informative) word forms to a single stem to have a significant effect on the ranking of features in terms of *information gain*. The effect of stemming on classification accuracy will therefore also be small.

Unstemmed		DPS		SteDL	
form	rank	form	rank	form	rank
gespaarde	8	spaar	3	spaar	3
gespaard	44				
sparen	83				
uur	11	uur	7	uur	7
uren	40				
probleem	39	probleem	32	probleem	13
problemen	41	problem	41		

Table 8: Conflated words from 100 most informative uni/bi-grams

Unstemmed		DPS		SteDL	
form	rank	form	rank	form	rank
bedrijf	15	bedrijf	8	bedrijf	8
bedrijven	63				
leider	37	leider	17	leider	17
leiders	65				
speler	45	speler	13	speler	13
spelers	46				
wedstrijd	3	wedstrijd	1	wedstrijd	1
wedstrijden	13				

Table 9: Conflated words from 100 most informative uni/bi-grams for the Newspaper corpus

5 Conclusion and Future Work

In this paper, we have first presented a new stemmer with dictionary lookup for Dutch which clearly outperforms the Dutch Porter stemmer in terms of accuracy and ability to reduce related word forms to a single stem. Next, we applied both stemmers to two widely different text classification tasks and found that stemming does not consistently improve classification accuracy.

A potential explanation is that even though stemming helps to reduce the number of features, the features which actually pass the feature-selection criteria are not affected substantially. One might conclude from this observation that stemming by itself is not effective enough to have an effect on classification accuracy. It is possible, however, that a combination of stemming with other means of text analysis and normalization could be effective. In particular one might investigate the effect of shallow text processing (i.e. replacing dates, names, phone number, IP-addresses, etc. by a single word) and compound analysis in combination with stemming. In (Busemann et al. 2000) such an approach is effectively appliedto German and one might therefore expect that a similar, more involved preprocessing, approach might also work for Dutch.

6 Acknowledgments

This research was carried out within the framework of the PIONIER Project *Algorithms for Linguistic Processing* and the KOP (*Kennisontwikkeling in Partnerschap*) project on Email Classification, both at the University of Groningen. The PIONIER Project is funded by NWO and the University of Groningen. The Kop project is funded by BSC Customer Care, Groningen, and the University of Groningen.

References

Androutsopoulos, I., Paliouras, G., Karakletsis, V., Sakkis, G., Spyropoulos, C. and Stamatopoulos, P.(2000), Learning to filter spam e-mail: A comparison of a naive bayesian and memory-based approach, *Proceedings of the Workshop on Machine Learning and Textual Information Access, PFDD 2000*, Lyon, pp. 1–3.

Baayen, R. H., Piepenbrock, R. and van Rijn, H.(1993), The CELEX lexical database (CD-ROM). Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA.

Busemann, S., Schmeier, S. and Arens, R. G.(2000), Message classification in the call center, *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, WA.

Daciuk, J.(2000), Finite state tools for natural language processing, *Proceedings of the COLING 2000 Workshop "Using Toolsets and Architectures to Build NLP Systems"*, Centre Universitaire, Luxembourg, pp. 34–37.

Frakes, W. B. and Baeza-Yates, R. (eds)(1992), *Information Retrieval: Data Structures & Algorithms*, Prentice Hall.

Harman, D.(1991), How effective is suffixing, *Journal of the American Society for Information Science* **42**(1), 7–15.

Kraaij, W. and Pohlman, R.(1994), Porter's stemming algorithm for dutch, *in* L. Noordman and W. de Vroomen (eds), *Informatiewetenschap 1994: Wetenschapelijke bijdragen aan de derde STINFON Conferentie*, Tilburg, pp. 167–180.

Kraaij, W. and Pohlman, R.(1995), Evaluation of a dutch stemming algorithm, *The New Review of Document and Text Management*, Taylor Graham, London, chapter 1, pp. 23–45.

Kraaij, W. and Pohlman, R.(1996), Using linguistic knowledge in information retrieval, *OTS Working Paper OTS-WP-CL-96-001*, OTS, University of Utrecht.

Krovetz, R.(1993), Viewing morphology as an inference process, *in* R. Horfhage, E. M. Rasmussen and P. Willett (eds), *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, PA, pp. 191–203.

Popovič, M. and Willett, P.(1992), The effectiveness of stemming for natural language access to slovene textual data, *Journal of the American Society for Information Science* **43**(5), 384–390.

Porter, M.(1980), An algorithm for suffix stripping, *Program* **14**(3), 130–137.

Riloff, E.(1995), Little words can make a big difference for text classification, *in* E. A. Fox, P. Ingwersen and R. Fidel (eds), *Proceedings of the 18th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Seattle WA, pp. 130–136.

Spitters, M.(2000), Comparing feature sets for learning text categorization, *Proceedings of the 6th Conference on Content-Based Multimedia Information Access (RIAO 2002)*, Paris, pp. 1124–1135.