Evaluation of a Dutch stemming algorithm

Wessel Kraaij and Renée Pohlmann

1 Introduction

In state of the art Information Retrieval (IR) systems the most salient problem is to improve recall rates while retaining high precision. A simple recall enhancing technique which can be useful for even the simplest boolean retrieval systems is stemming. It is obvious that an information-seeker who is looking for texts about, for example, *dogs* is probably interested in a text which contains the word *dog*. An algorithm which maps different morphological variants to their base form (stem) is called a stemming algorithm. The underlying assumption for fruitful usage of such a stemmer is that morphological variants of words are semantically related. This is obviously not always true. In information retrieval, the use of stemming is controversial (cf. Harman 1991). However, several authors (Frakes and Baeza-Yates 1992; Krovetz 1993; Popovič and Willett 1992) report favourable results[1].

The UPLIFT project[2] investigates whether linguistic tools can improve the performance of an IR system for Dutch texts. As a first step we will adapt and test two common stemming techniques for Dutch text. The first technique, quite popular in several experimental and commercial IR systems is *suffix stripping*. Suffix stripping is a pragmatic approach. The algorithms are small and efficient and are not hampered by linguistic claims. Efficiency is an important property of every subpart of an IR system, especially for modern interactive systems. However, the simple architecture of such algorithms has drawbacks, such as the easy introduction of errors. The second technique, *Stemming based on morphological analysis*, requires more complex resources. This approach tries to exploit linguistic knowledge about the internal structure of word forms. A necessary component for such a morphological analysis is a dictionary. In general, each word which has to be stemmed will involve dictionary lookup and therefore this technique will be considerably slower than suffix stripping. On the other hand, such careful

[1] See section 2 for a more elaborate discussion.

[2] UPLIFT (Utrecht Project: Linguistic Information for Free Text retrieval) is sponsored by the NBBI, Philips Research, the Foundation for Language Technology, the Ministry of Education and Science and the Ministry of Economic Affairs.

morphological analysis can eliminate most errors and can be useful for higher levels of interpretation, as in a Noun Phrase indexing module.

This paper describes the development and evaluation of a suffix stripper for Dutch. We have chosen to modify the stemming algorithm developed by Porter (1980) because it is well known and is frequently used in experimental IR systems.

2 Suffix stripping

The core of every suffix stripper is a set of rules which first test whether a word ends with a certain character sequence and subsequently delete this sequence. However, some strippers are a bit more sophisticated than others. Instead of deleting a suffix, they might replace it by another (shorter) suffix or modify the stem itself.

Harman (1991) compared three well-known stemming algorithms for English:

- S–stemmer: a simple stemmer removing the plural **s**
- Lovins (1968): a longest match stemmer consisting of 260 suffixes with a list of exceptions
- Porter (1980): a multi-step stemmer without exception list

In Harman's experiments, stemming (i.e. suffix stripping) did not yield any significant improvement in the performance of IR systems. Recall did improve, but precision deteriorated with stemming. Harman suggested that the latter effect could possibly be prevented by a more elaborate, dictionary based, stemming algorithm which checks whether the resulting stem is semantically related to the original term. The latter approach has been investigated by Krovetz (1993), who did find a significant improvement in performance compared to Porter stemming.

It could also be the case that Harman's negative results were caused by the relative simplicity of English morphology. Experiments with a Porter-like stemmer for the Slovene Language by Popovič and Willett (1992), containing 5276 suffixes, show a significant improvement in precision (at fixed retrieval of the 10 most highly ranked documents). Popovič's study included an interesting control experiment. The Slovene test corpus was translated to English and the experiment was repeated. This control experiment confirmed Harman's conclusion that Porter-like stemming does not improve retrieval for English documents. The study thus supports the

hypothesis that the effectiveness of stemming in an IR system also depends on the morphological complexity of a language. Since the morphological complexity of the Dutch language can be rated somewhere between English and Slovenian, we decided to further investigate the hypothesis through an experiment using a Dutch version of the Porter algorithm.

3 A Dutch version of the Porter algorithm

3.1 Introduction

Porter's algorithm is based on a series of steps that each remove a certain type of suffix by substitution rules. These rules only apply when certain conditions hold, e.g. the resulting stem must have a certain minimal length.

Most rules have a condition based on the so-called *measure*. The measure is the number of vowel-consonant sequences (where consecutive vowels or consonants are counted as one) which are present in the resulting stem. This condition must prevent removal of letters which look like a suffix but are actually part of the stem. Other simple conditions on the stem are:

- *Does the stem contain a vowel?*
- *Does the stem end with a consonant?*

Out of several implementations of Porter's algorithm for English we chose the version that was published by Frakes and Baeza-Yates (1992). This version has the advantage of a clear separation between substitution rules and procedures which test the attached conditions.

3.2 Extensions to Frakes' implementation

In Dutch, the prefix (or infix in the case of separable verbs) *ge-* is used in the formation of the past participle[3]. Because this affix can be easily recognised, the algorithm has been extended to handle pre- and infixes. The original Porter only treats suffixes.

[3]Exceptions to this rule are verbs starting with un-accented *be-*, *er-*, *ge-*, *her-*, *ont-* and *ver-* and inseparable complex verbs with an un-accented first syllable (e.g. voorkómen).

Another special case is the use of the compounding hyphen. In Dutch it is easy to create new words by compounding, sometimes using hyphens as glue. The hyphen is employed in a sometimes rather ad-hoc manner although strict rules apply for the official Dutch spelling. A stemmer without a dictionary is unable to do compound analysis, so three approaches are possible: treat the hyphen as a normal character, remove every hyphen between words, or replace the hyphen by a blank, having the effect of separating words. For our test corpus the second option yielded the best results.

Finally the stemmer was extended to handle characters with diacritics such as diaeresis and accents. The Dutch Porter algorithm can handle the ISO-latin1 character set; the orthographic rules for the placement of these diacritics for various inflectional forms are respected by the affix-rules which will be described in the next paragraph, e.g. *creëren - creëer* but *variëren - varieer*.

3.3 Affix-rules for Dutch

The affix-rules for Dutch were written using information in the *Morfologisch Handboek van het Nederlands* (de Haas and Trommelen 1993), the *Algemene Nederlandse Spraakkunst* (Geerts et al. 1984) and *Woordfrequenties in Geschreven en Gesproken Nederlands* (Uit den Boogaart 1975).

Several criteria were taken into consideration in the definition of the coverage of the rule clusters, the following being the most important:

- Inflectional morphology should be covered as fully as possible.

  Inflectional affixes (e.g. plural endings, verbal inflection etc.) do not affect the basic meaning of the underlying stem and can therefore be removed without risk of losing information.

- Only those derivational affixes which do not substantially affect the information conveyed by the stem should be removed.

  Affixes such as *-heid* ('-ness') can be removed without losing too much information. On the other hand, removal of an affix like *on-* ('un-'), would result in the loss of valuable information.

- The most frequent affixes should be covered.

  Since the number of rules influences the efficiency of the stemming algorithm we restricted ourselves to removing only the most frequent affixes.

Taking these considerations into account, six rule clusters were created for the Dutch Porter stemmer. Each cluster represents a particular class of affixes and the rules within a class are ordered and mutually exclusive, i.e. the first rule that matches is applied, and no other rules in the same cluster are tried. The affix-clusters are defined by the level at which the affixes occur in the word formation process. For instance, inflectional suffixes which occur on the outside of words are ordered before derivational suffixes e.g. *werk* + *ing* (derivation-Al) + *en* (inflectional)[4]. Complex affixes are thus removed in consecutive steps.

In addition to the affix-rules, a number of special conditions had to be designed to cover some specific phenomena. Examples of these conditions are, for instance, EndsWithV/C, i.e. the remaining stem should end in a vowel or consonant. DupV is a special case. Long vowels in Dutch are spelled single in open syllables and double in closed ones (e.g. *schaap* - *schapen*). After removal of some affixes, e.g. adjective *-e* (*rode* - *rood*), infinitival *-en* for verbs (*lopen* - *loop*), etc., the stem vowel needs to be doubled to render an orthographic-ally correct stem. The rules which remove these affixes are marked for the DupV procedure. DupV identifies closed syllables and subsequently duplicates the vowel, otherwise the vowel is left unchanged. This works reasonably well for *a*, *o* and *u* (*i* is never doubled) but *e* poses a special problem since an *e* in spelling can also stand for an un-accented schwa which is never doubled. DupV tries to "guess" the status of the *e* based on information about the spelling of the word in which it is contained, e.g. if *e* is the only vowel in the word it is not a schwa, but without information about word stress it is impossible to consistently predict the status of *e* correctly, e.g. *kantélen* ('battlements') → *kanteel* v.s. *kántelen* ('to turn over') → *kantel*.

The affix-rules have the following general form:

suffix → *substitution* measure-condition <additional conditions> < DupV>

The first cluster of rules covers the inflectional morphology of nouns, adjectives and verbs, e.g.:

"en"	→	∅	measure > 0	EndsWithC	DupV	(*-en* plural)
"e"	→	∅	measure > 0	EndsWithC	DupV	(adjective *-e*)

The second cluster covers the diminutive suffix of nouns, e.g.:

"etj"	→	∅	no measure-condition	EndsWithC	(*-etje*[5])
"tj"	→	∅	no measure-condition	None	(*-tje*[5])

[4]In some cases this basic ordering could not be adhered to because of rule interaction; for instance, the rule removing *-d* (verbal inflectional suffix) had to be ordered after the rule removing *-end* (adjective-forming derivational suffix/verbal inflectional suffix).

[5]final *-e* has already been removed.

The third cluster contains noun-forming derivational suffixes, e.g.:

"heid"	→	∅	measure > 0	None		(*-heid*)
"ing"	→	∅	measure > 0	None	DupV	(*-ing*)

The fourth cluster contains adjective-forming derivational suffixes, e.g.:

"baar"	→	∅	measure > 0	None		(*-baar*)
"ig"	→	∅	measure > 0	None	DupV	(*-ig*)

The fifth cluster covers a special case: the affix *ge* which occurs as a prefix (regular) or infix (separable verbs) in Dutch participles.

"ge-"	→	∅	no measure-condition	None	(*ge-*)
"-ge-"	→	∅	measure > 0	None	(*-ge-*)

The final cluster contains rules that tidy up the result of previous rule applications, e.g.:

"v"	→	"f"	no measure-condition	None	(*-v → -f*)
"pp"	→	"p"	no measure-condition	None	(*-pp → -p*)

Porter reports a reduction of about a third in vocabulary size after application of his stemmer to a vocabulary of 10.000 different word forms (cf. Porter 1980: 137). We repeated this test using a larger vocabulary for both the Dutch and the English Porter algorithm (as implemented by Frakes):

	NL Porter	EN Porter
Original number of word forms	148.601	104.216
After stemming	64.035	49.323
reduction	57 %	53 %

The results of this test show that the behaviour of both stemmers is comparable with respect to the measure of vocabulary reduction. Although this measure can be used as a global indication of the effectiveness of the stemming algorithm, other evaluation measures are necessary to reveal specific error patterns. This information can subsequently be used to improve the algorithm, where possible. Some error types, however, are inherent to the suffix-stripping method and without the additional information provided by, for instance, a dictionary, these errors cannot be avoided.

The following are examples of these types of errors:

- Linguistically incorrect stems

  Some stems which are generated by the Porter algorithm are not linguistically correct. This may not be a problem if the resulting “stem” is unique and

consistent for a semantically related group of words, but if the resulting stem is identical to a stem that is not semantically related, retrieval errors occur.

- Homographs

  Homographs are words which are spelled identically but nevertheless have a different meaning, e.g. *kust* (3rd person singular of the verb *kussen* ('to kiss'), or a noun meaning 'coast'). Because the Porter algorithm does not have access to information about, for instance, word categories, the different senses of these types of words are not distinguished.

- Irregular verbs

  Some verbs exhibit irregularities in the formation of past tense, past participle or both, e.g. *drinken dronk gedronken, zien zag gezien* (base vowel alternation). Others, like *zijn* ('to be'), are almost completely irregular. For obvious reasons, a simple suffix-stripper will never be able to map the different forms of these types of verbs onto a single stem.

Generally speaking, the different types of errors introduced by suffix-stripping algorithms like the Porter algorithm can be divided into two classes:

1. Overstemming errors

   "Overstemming errors" are those errors which result in the conflation of semantically unrelated words.

2. Understemming errors

   The term "understemming errors" is used for those errors where a failure to conflate semantically related words is concerned.

In section 4 we will describe a method developed by Paice to calculate an overstemming and understemming index for stemming algorithms and introduce the results we obtained by applying his method to our Dutch version of the Porter algorithm.

4 Performance evaluation

4.1 Paice's stemmer evaluation method

In this section we will present an evaluation method proposed by Paice (1994). Paice has compared different English stemming algorithms in isolation from the

context of an IR system. He did not use the traditional precision/recall parameters, but instead introduced two new parameters: the over- and understemming index (*UI* and *OI*) and their ratio, the *stemming weight (SW)*, in order to make a qualitative comparison between different stemmers. A prerequisite of this method is a list of groups of semantically related words. An ideal stemmer should stem all words in a group to the same stem. If a stemmed group contains more than one unique stem, the stemmer has made understemming errors. In an IR system this corresponds to a negative effect on recall. If a stem of a certain group also occurs in other stemmed groups, the stemmer has made overstemming errors, which deteriorate precision. A good stemmer should therefore produce as few under- and overstemming errors as possible. Note however that there is a tradeoff here. If suffix stripping rules are added or modified in order to reduce the understemming errors, these modifications will probably introduce additional overstemming errors. The development of rules is based on a thorough analysis of stemming errors. The method to be described in 4.4 can be of help in finding an optimal balance.

It is not trivial to create a large file of grouped words. Paice started from the CISI corpus (consisting of titles and abstracts), filtered out 9757 unique word forms and produced a group file in a semi-automatic manner. A grouping program made the "obvious" decisions and referred to the user in the difficult cases. One rule of thumb that Paice used was that words must share at least two letters. He did not want to penalize the stemmer's ignorance concerning irregular verbs like 'to be' or 'to go'. Paice also experimented with *tight* and *loose* grouping processes. This meant that some tight groups were unified into loose groups, reflecting a more remote semantical relationship.

We have taken a different approach in producing a group file for Dutch. The group file was produced by a number of computer programs which exploit the morphological information of the CELEX database (Baayen et al. 1993). For our purpose we used the word forms database covering Dutch inflection and a database of lemmata which gives all possible segmentations of derivational forms and compounds. The word form database lists a lemma for each word form. Finding a root form for derivational or compound forms is a bit more complicated. We have decided not to use the compound segmentation information since compound analysis is beyond the scope of a suffix stripper. But we did use the segmentation information about derivational lemmata, e.g. *verrader+lijk*. All words with the same root form (lemma) were joined in a group. A prerequisite of Paice's evaluation method is that the collection of words which is taken as input

for the grouping process does not contain duplicates. In order to accomplish this, ambiguous word forms (i.e. homographs) were removed. Many Dutch verbs are separable[6], e.g. 'ophalen'. All inflected forms of these verbs which have separate lexemes (e.g. 'haalt op') were removed as well, because adequate stemming would require syntactic analysis.

Here are some examples of the resulting groups:

1. malloot ('idiot') mallotig malloterigheid malloterig mallotigheid malloten malloterigheden mallotigheden malloterige malloteriger malloterigere malloterigst malloterigste mallotige mallotiger mallotigere mallotigst mallotigste
2. manoeuvreren ('to manoeuvre') gemanoeuvreerd gemanoeuvreerde manoeuvreer manoeuvreerde manoeuvreerden manoeuvreert manoeuvrerend manoeuvrerende manoeuvreerbaar manoeuvreerbaarder manoeuvreerbaardere manoeuvreerbaarst manoeuvreerbaarste manoeuvreerbare
3. verraden ('to betray') verraad verraadde verraadden verraadt verradend verradende verried verrieden verraderlijkheid verraderij verrader verraadster verraderlijk verraadsters verraders verraderijen verraderlijke verraderlijker verraderlijkere verraderlijkst verraderlijkste
4. geboorte ('birth') geboorten geboren
5. boren ('to drill') boorde boorden boort borend borende geboord geboorde boorden

Processing the example by the Dutch Porter yields:

1. malloot malloot malloot malloot malloot malloot malloot malloot malloot malloot malloot malloot malloot malloot malloot malloot malloot
2. manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer manoeuvreer
3. verraad verraad verraad verraad verraad verraad verraad verried verried verrader verraad verraad verraad verrader verraad verraad verraad verrader verrader verrader verrader verrader
4. boor boor boor
5. boor boor boor boor boor boor boor boor boor

This example displays errors of two different kinds:

1. Group 3 is not conflated to a single root form. This is an example of *Unachieved merges* (understemming).

[6]These complex verbs consist of a particle and a verbal part which can be separated.

2. Group 4 and 5 are conflated to one single root form creating a potential source of "noise" in an IR system. This is an example of *Unwanted merges* (overstemming).

The under- and overstemming index can be computed from four parameters:

1. GDMT: The Global Desired Merge Total
2. GDNT: The Global Desired Non-Merge Total
3. GUMT: The Global Unachieved Merge Total
4. GWMT: The Global Wrongly Merged Total

For each group g the *desired merge total* is equivalent to the total number of different possible word form pairs in the particular group. On the other hand, the *desired non-merge total* for a group can be computed by counting the possible word form pairs that are composed by a member and a non-member word form from the group. This can be expressed in the following definitions, where n_g denotes the number of words in a group and W is the total number of words.

$$DMT_g = \frac{1}{2} n_g (n_g - 1) \tag{1}$$

$$DNT_g = \frac{1}{2} n_g (W - n_g) \tag{2}$$

For each *stemmed* group the Unachieved Merge Total (UMT) can be calculated by counting the number of merges between individual words that were not achieved. Suppose a group of size n_g contains s distinct stems after stemming and that the number of instances of these stems are u_1, u_2, ... u_i respectively. The number of understemming errors for that group is then given by:

$$UMT_g = \frac{1}{2} \sum_{i=1}^{s} u_i (n_g - u_i) \tag{3}$$

An extra operation is needed to compute the Wrongly Merged Total (WMT). An inverted group file is created by grouping all identical stems labeled with their original concept group index. Now we can compute the Wrongly-Merged Total for each *inverted* group by comparing the indices. If an inverted group contains only identical indices, then there are no overstemming errors and the WMT value for that group is zero. However, if an inverted groups contains

different indices, this means that semantically unrelated words are conflated to the same stem. In this case the WMT can be computed by counting the number of possible combinations of two words with different indices within an inverted group. Suppose a stem group of size n_s contains t different indices. The stems in this group thus originate from t different concept groups. The number of each original concept group (labeled with the same index) within this stem group is represented by v_1, v_2,...v_t. The number of overstemming errors for this group (WMT) is then defined by:

$$WMT_s = \frac{1}{2}\sum_{i=1}^{t} v_i(n_s - v_i) \tag{4}$$

The "Global" values of DMT, DNT, UMT and WMT are simply summations over all groups.

Here are the inverted groups for the examples above:

- malloot: malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) malloot(1) *WMT = 0*
- manoeuvreer: manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) manoeuvreer(2) *WMT = 0*
- verraad: verraad(3) verraad(3) verraad(3) verraad(3) verraad(3) verraad(3) verraad(3) verraad(3) verraad(3) verraad(3) verrader verraad(3) verraad(3) verraad(3) *WMT = 0*
- verried: verried(3) verried(3) *WMT = 0*
- verrader: verrader(3) verrader(3) verrader(3) verrader(3) verrader(3) verrader(3) *WMT = 0*
- boor: boor(4) boor(4) boor(4) boor(5) boor(5) boor(5) boor(5) boor(5) boor(5) boor(5) boor(5) boor(5) *WMT = 27*

The GDMT and GDNT values are used to normalize the number of over- and understemming errors to a fraction of 1: $UI = GUMT/GDMT$. The overstemming index is $OI = GWMT/GDNT$. The stemming weight is defined as: $SW = OI/UI$. SW gives some indication whether a stemmer is weak (low value) or strong (high value).

4.2 A comparison with Paice's results

Paice has compared three English stemmers: Lovins, Paice/Husk and Porter with the truncate "stemmer". The trunc(n) stemmer reduces a word to its first *n* characters.

We replicate the results of Paice's analysis in table 1 so that we can compare our results. We selected the results for tight grouping in which strict semantic rules were applied for the grouping process, as it seems to correspond well with our CELEX-based automatic grouping method.

stemming algorithm	UI	$OI \times 10^{-5}$	$SW \times 10^{-4}$
trunc(4)	0.062	81.4	131.00
trunc(5)	0.18	26.2	14.80
trunc(6)	0.34	7.3	2.18
trunc(7)	0.53	2.8	0.54
trunc(8)	0.70	1.2	0.17
Lovins	0.33	6.3	1.93
Paice/Husk	0.12	11.8	9.80
Porter	0.37	2.8	0.74

Table 1: Comparison of English stemmers (Paice)

Since we did not have another stemmer for Dutch at our disposal, we decided to run tests with the truncate "stemmer" as well, just as Paice did for English. We also ran tests to investigate the effect of varying the number of rule clusters that were applied: The Porter algorithm with only the first cluster of rules (inflection), with clusters 1 and 2, etc. The results of these tests are presented in table 2.

The performance of the Dutch Porter algorithm is consistent with the English version, i.e. it is a rather cautious or "weak"[7] stemming algorithm. The tendency for understemming is, however, not so obvious when the stemmed group files are inspected: the Dutch stemmer copes very well with verbal inflection or derivation. A possible cause for this understemming tendency in comparison with the English stemmers could be the language difference, or a different coverage of the language. One way to compare the two group collections is to compare the average group size. We started from 286461 words extracted from CELEX which were split into

[7] A *weak* stemmer produces more understemming than overstemming errors, and a *strong* stemmer the other way around.

stemming algorithm	UI	$OI \times 10^{-5}$	$SW \times 10^{-5}$
trunc(4)	0.200	80.00	410.00
trunc(5)	0.290	23.60	81.10
trunc(6)	0.390	7.23	18.60
trunc(7)	0.510	2.15	4.20
trunc(8)	0.620	0.92	1.47
trunc(9)	0.723	0.44	0.61
Porter (1 cluster)	0.827	0.05	0.06
Porter (2 clusters)	0.815	0.06	0.08
Porter (3 clusters)	0.621	0.17	0.28
Porter (4 clusters)	0.425	0.36	0.84
Porter (all clusters)	0.310	0.51	1.67

Table 2: Comparison of Porter and trunc(n)

74625 groups i.e. 3.8 words per group. Paice's corpus consists of 5101 different groups (tight grouping procedure) which makes the average group size 1.9 words per group. The Dutch groups contain every possible inflection resulting in large concept groups which expand from a verbal concept. Our stemmer is not able to stem inflectional forms of "strong verbs" to one identical stem. These errors immediately result in a high UMT value just because of the large average "verbal group" size.

Figure 1 shows the effect of removing rule clusters. Note that Porter with only 1, 2 or 3 rule clusters performs worse than truncation. The plot can be interpreted as follows: better stemmers are closer to the origin. Unfortunately, *UI* and *OI* cannot be compared in an absolute sense so it is difficult to define an ideal stemming weight or to express the total error rate in one parameter such as the length of the vector. The next paragraph describes two possible solutions for this problem.

4.3 Improved metrics: MUR, MOR & MMF

Paice (1995) recently proposed to redefine the Overstemming index by normalizing it with the *Global Actual Merge Total* instead of the *Global Desired Non-merge Total*. The AMT for each stem group s is defined as:

$$AMT_s = \frac{1}{2} n_s (n_s - 1) \tag{5}$$

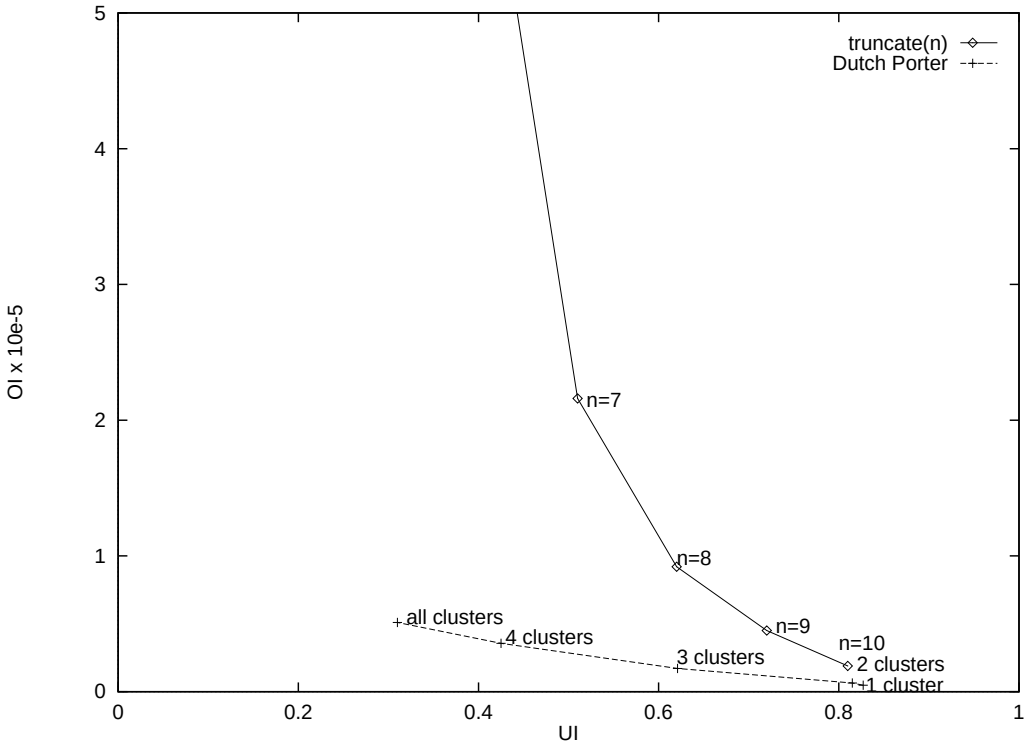

Figure 1: UI x OI plot

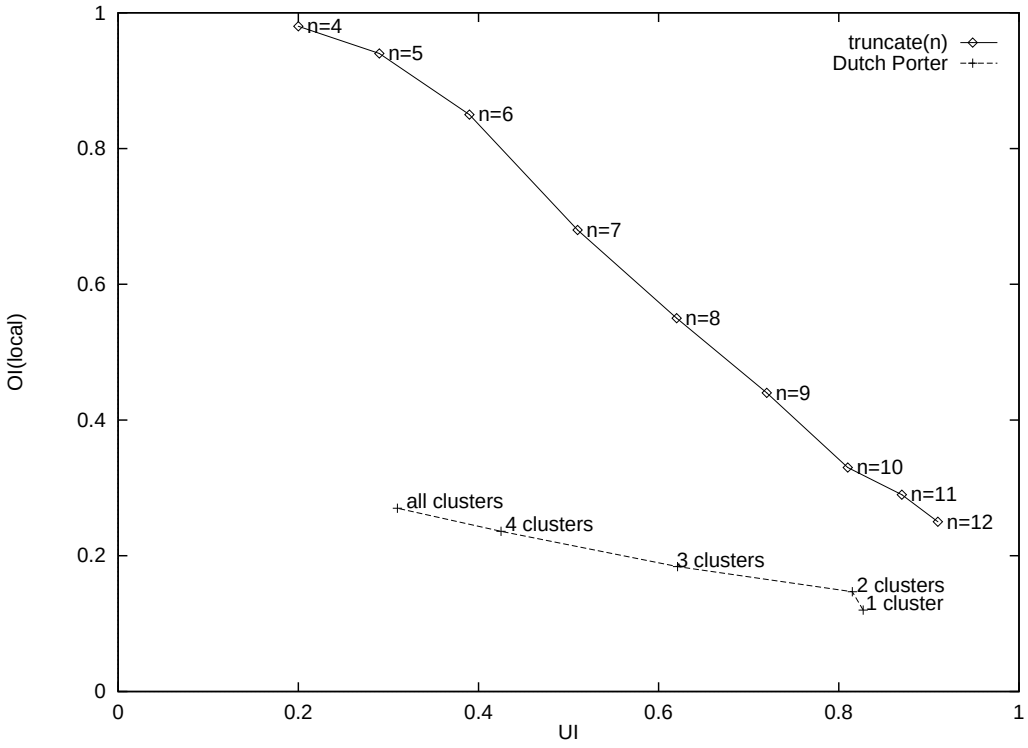

Figure 2: UI x OI(local) plot

The normalization in this "local" definition results in figures comparable to the Understemming Index. Moreover, Paice reports that the new local OI definition is less dependent on the size of the test corpus. However, Paice does not explain why the new OI definition gives more satisfactory results (i.e. why the variability of the "local" definition is smaller).

Figure 2 shows a plot of the same data with the new local OI definition.

We propose to extend the idea of *locality* even further by giving alternative definitions of under- and overstemming. Consider the model in figure 3:

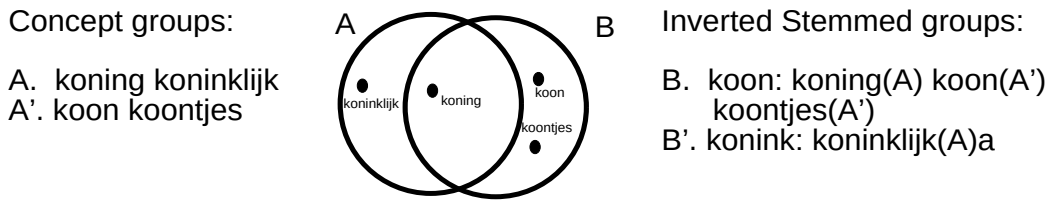

Figure 3: A local stemming model for the word 'koning'

The test corpus in this example consists of two small concept groups which result in two inverted stem groups after stemming. The example shows both under- and overstemming errors. The idea is that for each *word* in the test corpus we can look up its concept group (set A), and its expansion (i.e. the set of unstemmed words which are stemmed to the same root as the word in question). The expansion corresponds to set B. Now the *under-* and *overstemming ratio* can be straightforwardly defined. The understemming ratio is just the fraction of the original concept group A which is not conflated by the stemmer. The overstemming ratio is the fraction of the expansion set B which contains unrelated concepts.

(6) $$UR = \frac{card(A - B)}{card(A)}$$

(7) $$OR = \frac{card(B - A)}{card(B)}$$

The figure shows sets A and B for the word *koning ('king')*. The concept group also contains the word *koninklijk ('royal')* but both words are not stemmed to the same root. Instead, the expansion (set B) of *koon* (the stem of *koning*) contains the words *koon ('cheek')* and *koontjes ('small cheeks')* which are part of the concept group A'.

In the same spirit we can define the *match factor* as the level of overlap between original concept group A (desired expansion) and actual expansion set B. An ideal stemmer would result in a MF of 1 for each word.

(8) $$MF = \frac{card(A \cap B)}{card(A \cup B)}$$

We have computed the UR, OR and MF for each word in our test corpus. These figures were averaged, resulting in the Mean UR (MUR), Mean OR (MOR) and Mean MF (MMF) respectively. Figure 4 shows the results.

The difference between Paice's UI and OI (local) and our alternatives MUR and MOR is conceptually not very large. Paice computes the UI and OI by taking the quotient of two averages (the GWMT, GDNT etc. can be seen as *M* times the average value, where *M* is the number of groups). Our local definition of MUR and MOR is based on computing the mean of a quotient. This difference is known as macro- versus micro-averaging.

Our approach has the advantage that the effects of words in large concept groups (for example with all verbal forms) do not dominate the results. The weight of each word is equal.

The second advantage of our proposed alternative model is that we have a single metric for the quality of a stemmer, the Mean Match Factor. However, the MMF presupposes that under- and overstemming errors are given equal importance.

Figure 5 combines two performance metrics in one plot: The mean match factor is given for different values of the stemming weight, which in turn correspond to different versions of the Porter and truncate stemmer. The plot clearly shows that the truncate stemmer performs best at n=8. The performance of the Porter stemmer increases strongly when more rule clusters are applied, with only a modest increase in stemming weight. The stemming weight stays below 1, indicating that our stemmer favours precision over recall.

4.4 Fine-tuning the stemmer

In addition to applying Paice's evaluation method to different stemmers, we also applied his method to different versions of our Dutch stemmer. The rich lexical information in the CELEX database enabled us to create group files for a number of distinct morphological categories. This made it possible to assess the merits of the Dutch Porter algorithm for different aspects of Dutch morphology in a very precise way.

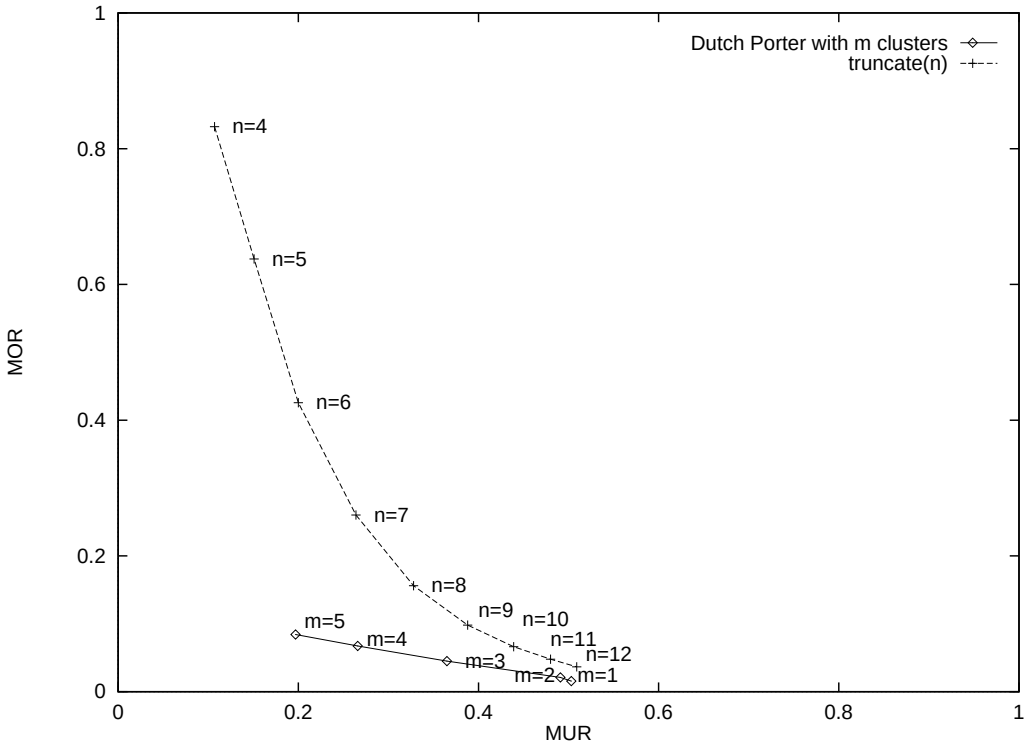

Figure 4: MUR vs. MOR

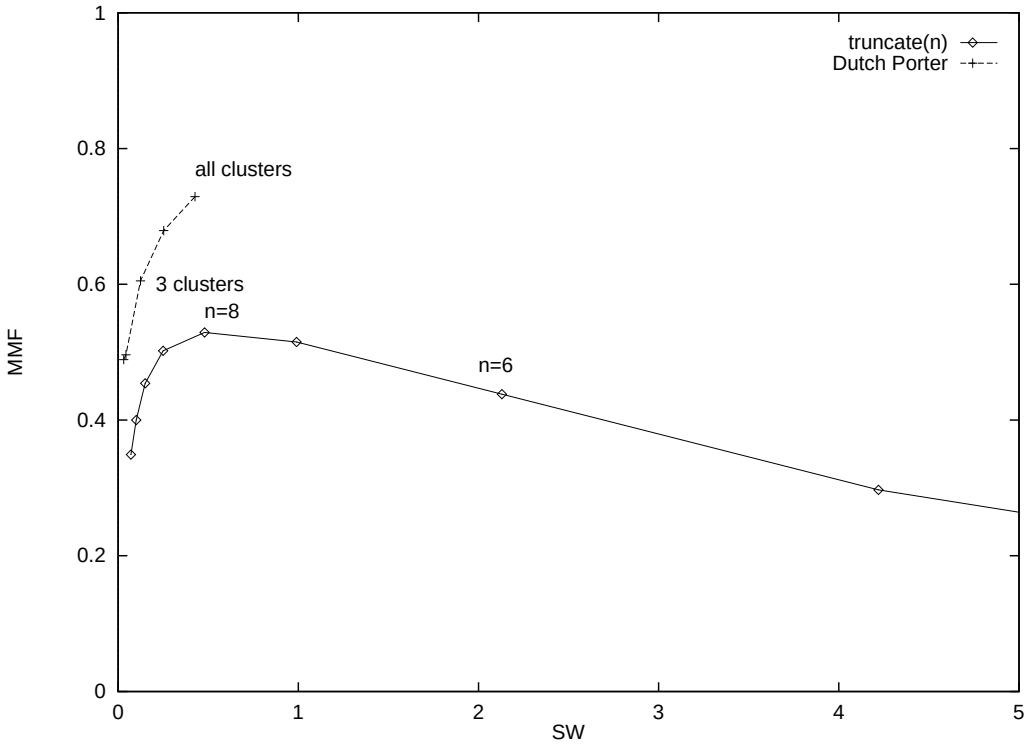

Figure 5: MMF vs. Stemming Weight

The CELEX lexical database distinguishes between inflectional and derivational morphology. We have evaluated nearly all inflectional categories. Infinitive (verbs), absolute (adjectives) and singular (nouns) were excluded from the list of inflectional categories because these forms are root forms in CELEX and have no inflection[8]. Because the Dutch Porter only tries to remove inflectional and derivational affixes, we only isolated derivational word forms and their root forms from CELEX and did not decompose compounds. So compound forms without any inflectional or derivational affixes were treated as root forms. The categories 'deriv+affixsub' and 'deriv+allomorf' (which are subsets of 'derivations') exhibit irregular morphology. Allomorphy is the phenomenon where stems within words are different from their generally accepted stem form, e.g. *aanspreek* + *-elijk* yields *aansprakelijk*. Affix Substitution is the process whereby part of the stem is replaced when stem and affix are joined together, e.g. *emigreer* + *-atie* yields *emigratie*.

For each morphological category, groups were produced by taking the inflected form of a word and its lemma (provided by CELEX). Aggregate group files were produced by merging inflectional categories from the same syntactical category. 'All verb forms' contains all verbal inflectional forms , 'all nouns' contains singular, plural nouns and diminutives, 'all adjectives' contains 'absolute', 'comparative' and 'superlative'. Subsequently we ran the UI/OI analysis on all these group files.

The merit of this detailed analysis is that the effects of small changes in a particular rule cluster (aimed at the removal of a certain inflectional or derivational suffix) can be evaluated in a precise way. It is easy to see whether the change affects the particular suffix category at which it is aimed and (equally importantly) whether it does not harm the performance on other categories. The scripts can also automatically generate and sort all over- and understemming errors. We think that this method is important support for a pure trial and error approach to rule development.

Table 3 shows that the understemming index is within the same order of magnitude for all categories. UI is high for the irregular derivational categories, as could be expected. The overstemming index is low (zero) for these categories, because the Dutch Porter algorithm treats these words as having regular morphology. The resulting stems have only a small chance to conflate with other stemmed irregular

[8]The Dutch Porter algorithm considers the first person singular present tense as the root form for verbal inflection instead of the infinitive. This discrepancy, however, does not affect the evaluation method.

forms. However, some of them will probably conflate with stems of regular word forms.

Category code	# groups	# words	UI	$OI \times 10^{-6}$	$SW \times 10^{-6}$
1st pers sing pres	9448	18930	0.253	2.80	11.0
2nd pers sing pres	11001	26854	0.281	3.32	11.8
3rd pers sing pres	10955	21923	0.310	2.75	8.9
present participle	11240	33516	0.190	2.19	11.4
past tense	11752	50687	0.265	6.12	23.1
past participle	11100	29342	0.295	4.72	16.0
participial adjective	11622	30370	0.299	5.57	18.6
all verb forms	12810	94459	0.305	6.65	21.8
plural nouns	60288	123191	0.189	2.47	13.1
diminutives	4034	10608	0.159	20.20	127.0
all nouns	58253	127854	0.245	1.51	6.2
genitive	91	183	0.505	0.00	0.0
dative	54	108	0.167	0.00	0.0
comparative	5501	17544	0.225	16.30	72.7
superlative	5406	17247	0.204	17.60	86.6
all adjectives	12441	55494	0.210	14.50	69.3
derivations	14755	36524	0.388	4.46	11.5
deriv+affixsub	1648	3757	0.665	0.00	0.0
deriv+allomorf	375	793	0.939	0.00	0.0

Table 3: End results of the Dutch Porter

5 Conclusion

The results of our evaluation can be summarised as follows:

- The Dutch Porter stemmer performs rather well, taking into account the limitations of the algorithm. We expect that Porter stemming will be effective for Dutch Text Retrieval.
- The qualitative evaluation method based on the over- and understemming indices introduced by Paice in combination with the lexical information in the CELEX database offers valuable support for the development and comparison of stemming algorithms.

- We propose an alternative "local" model of under- and overstemming. This model also supports a single metric for the error rate of a stemmer: the *mean match factor*.

The following further research seems advisable:

- Comparison of the Porter stemmer with a stemmer based on morphological analysis. Both techniques should be tested in an IR environment with a collection of Dutch texts.

References

Baayen, R. H., R. Piepenbrock, and H. van Rijn (Eds.) (1993). *The CELEX Lexical Database (CD-ROM)*. University of Pennsylvania, Philadelphia (PA): Linguistic Data Consortium.

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

Geerts, G., W. Haeseryn, J. de Rooij, and M. van der Toorn (Eds.) (1984). *Algemene Nederlandse Spraakkunst*. Groningen: Wolters Noordhoff.

de Haas, W. and M. Trommelen (1993). *Morfologisch Handboek van het Nederlands*, Volume 7 of *Aan het Woord*. 's-Gravenhage: SDU Uitgeverij.

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

Kraaij, W. and R. Pohlmann (1994). Porter's stemming algorithm for Dutch. In L. Noordman and W. de Vroomen (Eds.), *Informatiewetenschap 1994: Wetenschappelijke bijdragen aan de derde STINFON Conferentie*, pp. 167–180.

Krovetz, R. (1993). Viewing morphology as an inference process. In *Proceedings of ACM-SIGIR93*, pp. 191–203.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics 11*, 22–31.

Paice, C. D. (1994). An evaluation method for stemming algorithms. In *Proceedings of ACM-SIGIR94*, pp. 42–50.

Paice, C. D. (1995). A method for evaluation of stemming algorithms based on error counting. (to appear in JASIS in 1995).

Popovič, M. and P. Willett (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. F. (1980). An algorithm for suffix stripping. *Program 14*(3), 130–137.

Uit den Boogaart, P. C. (Ed.) (1975). *Woordfrequenties in Geschreven en Gesproken Nederlands*. Utrecht: Oosthoek, Scheltema en Holkema.