A Fast and Simple Stemming Algorithm for German Words[1]

Jörg Caumanns
Free University of Berlin, CeDiS
caule@gmx.de

Abstract

In this report a stemming algorithm for morphological complex languages like German or Dutch is presented. The main idea is not to use stems as common forms in order to make the algorithm simple and fast. The algorithm consists of two steps: First certain characters and/or character sequences are substituted. This step takes linguistic rules and statistical heuristics into account. In a second step a very simple, context free suffix-stripping algorithm is applied. Three variations of the algorithm are described: The simplest one can easily be implemented with 50 lines of C++ code while the most complex one requires about 100 lines of code and a small wordlist. Speed and quality of the algorithm can be scaled by applying further linguistic rules and statistical heuristics.

1 Introduction

Even in our times of Multi- and Hypermedia written text still is the preferred encoding of information. More than 80% of all documents found in the *World Wide Web* are textual [Bray96]. The largest search engine on the Web - *Altavista* - claims to have indexed over 100 million documents. These two numbers imply that anyone with access to the internet as well has free access to millions of textual documents.

As no one is able to keep track of this mass of information there is a strong demand for automatically indexing, searching, extracting, integrating, and restructuring information.

1.1 Stemming

One of the problems for any kind of automated text processing is the detection of different morphological variations of the same word. Especially for information retrieval, data mining, and automated text comparison these variations must be detected and mapped to a common form. Most algorithms for this conflation of terms use word stems as common forms. Stem-based conflation is usually just called *stemming:*

> *A stemmer should conflate together all and only those pairs of words which are semantically equivalent and share the same stem.* [Paice96]

[1] The algorithm described in this report was developed as part of a seminar at the department of computer science at the Free University of Berlin (Hinze, A., Sabisch, A., Schweppe, H., „Grundlagen Digitaler Bibliotheken". Wintersemester 1998/1999).

Stemming can be either weak or strong:

1. In the easiest case only different declensions of a word are detected and conflated. E.g. "house" and "houses" are stemmed to "house" and "mouse" and "mice" to "mouse". This is called *weak stemming*.
2. With *strong* stemming all suffixes (and sometimes even all prefixes) are removed. E.g. even a noun like "illness" is conflated to is adjective stem "ill".

For most applications weak stemming is both sufficient and desired, as in many cases prefixes and suffixes carry lexical and semantic meaning.

The main problems to be solved with weak stemming are:

- detecting the boundary between stem and ending,
- detecting regular changes of the stem (e.g. "Haus" and "Häuser" in German),
- detecting glue characters (e.g. "swim" and "swim**m**ing" in English or "essen" and "ge**g**essen" in German),
- detecting irregular changes of the stem (e.g. "go", "went", "gone" in English or "nehmen", "nahm", "genommen" in German).

For languages with just few morphological variants (e.g. English) the most common stemming method is *suffix stripping*. The idea of suffix stripping is to iteratively cut off all suffixes from a word. Suffix stripping usually is based on a dictionary of suffixes, a longest match algorithm, and some simple morphological rules. An examples for this strategy is [Porter80], which is used by most web-based search engines.

1.2 Stemming German Texts

Stemming for morphological more complex languages like Dutch and German is not that easy. The reasons why the combination of longest suffix matches and simple linguistics doesn't work properly are:

- Changes of the stem usually don't occur at the end of the stem but in the middle (e.g. "Haus" and "Häuser").
- All (even incomplete) rules for declensions of nouns are based on gender. Without lexical analysis or dictionary search it is impossible to say whether "er" is a suffix (as in "Bilder") or part of the stem (as in "Leber").
- There are many exceptions for when to exchange a vowel for an *Umlaut* to build a plural form (e.g. "Hund - Hunde" but "Mund - Münder").
- The German language makes heavy use of compound nouns. E.g. the German term for "nuclear power plant" is "Atomkraftwerk" and the room the director of nuclear power plant works in would be "Atomkraftwerksdirektorenzimmer".

As far as we know there are currently no fast **and** powerful algorithms available for stemming German texts. Linguistic approaches have difficulties with compound words which must be decomposed before stemming. Dictionary based approaches are slow because of their large word lists (about 50,000 entries seem to be needed [Sheridan96, Piotrowski98]). This is the reason why German search engines on the web - e.g. *fireball* - do no stemming at all.

1.3 Overview

In this report a stemming algorithm based on context free suffix stripping is introduced. The core algorithm is very fast and very simple (less than 100 lines of code). It can handle an infinite number of words and gets even stronger the more compounding occurs. The only requirement for the algorithm is that all stop words - determiners and other function words – must have been removed from the text to be stemmed.

The algorithm was designed for stemming German words but it can more or less easily be adapted for other languages. It has been implemented as part of an experimental search engine and is currently used for keyword search and text comparison within an application in the area of dynamic documents [Caumanns98].

The next chapter provides a description of the algorithm and the ideas it is based on. The algorithm has been tested with a wordlist containing more than 75,000 terms. Some of the results of these tests are given in chapter 3.

It should be noticed that the algorithm is not better than existing approaches but it is much smaller and faster. Further on it can easily be extended by either linguistic rules, statistical heuristics, or wordlists to improve quality for the price of speed.

2 The Substitute-and-Strip Algorithm

As mentioned above the purpose of stemming is to reduce different morphological forms of a word to a common form. Within this report the common form of different declensions of the same word is called the *discriminator* of the respective word. Discriminators should be unique in a way that only equivalent words that were derived from the same stem are mapped onto the same discriminator. Word stems are very good discriminators as they are usually unique (if homonyms are ignored).

In real life the requirement of uniqueness can never be met. Morphological homonyms like "nuts" (Plural of "nut" and synonym for "crazy") cannot be detected without performing a semantic analysis of the text to be stemmed. This adds some noise to any stemming algorithm as there will always be a certain number of words not mapped onto an unique discriminator.

Another problem for any stemming algorithm are irregular declensions. Verb forms like "go - went - gone" or nouns like "vertex - vertices" are very hard to detect as common. Especially for languages like Dutch or German that make heavy use of compounding these irregular form might get buried deep inside a word. Stemming these words correctly not only requires a wordlist of all irregular forms but even very sophisticated decomposing and recomposing algorithms.

To sum up these considerations it can be said that any stemming algorithm that is not based on semantic text analysis produces some kind of errors. Even the best algorithm will never be able to stem all words correctly.

The idea behind this work is very simple: If one knows that stemming is never perfect, why should one try? For most applications it doesn't matter if the error rate caused by wrong stemming is 2.1% or 3.1%. In either case the application has to deal with some amount of uncertainty.

The algorithm introduced in this report is based on a two-step strategy. In a first step certain characters and/or groups of characters are replaced within each word to be stemmed. In a second step suffixes are cut off in a context free manner. Both steps cause stemming errors in a sense that different words are mapped onto identical discriminators. The question of interest is if this additional error rate is low enough to be acceptable.

2.1 Recursive Context Free Suffix-Stripping

One of the most common stemming methods is *suffix stripping*. With suffix stripping each word is matched against a list of suffixes. The longest suffix matching the word's end is cut off (E.g. "going" is stemmed to "go" by cutting of the inflectional suffix "ing"). If suffix stripping is just based on simple character matching it is called *context free*.

The idea of context-free suffix stripping is not new, but it is often claimed to be error prone:

> *"Unfortunately, context free removal leads to a significant error rate. For example, we may well want UAL removed from FACTUAL but not from EQUAL."* [Rijsbergen79]

The problem addressed within this quote is easy to understand and easy to accept, but it is irrelevant. Why should "ual" not be removed from "equal"? As long as there is no other word mapped onto "eq" it is no problem to stem "equal" to "eq". E.g. my dictionary lists nine words starting with "eq" that are not derived form "equal". None of these nine words would be stemmed to "eq" by a context-free suffix stripping algorithm. The oversimplification of this quote is that "eq" surely is not the stem of "equal" but nevertheless it is an unique discriminator for "equal".

In German there are seven declensional suffixes for nouns: -s, -es, -e, -en, -n, -er, and -ern, 16 for adjectives: -e, -er, -en, -em, -ere, -erer, -eren, -erem, -ste, -ster, -sten, and -stem, and 48 for verbs: -e, -est, -st, -et, -t, -en, -ete, -te, etest, -test, eten, -ten, -etet, tet, -end-, and -nd- (-end- and -nd- turn verbs into adverbs and can be followed by any of the adjective suffixes). Additionally -ge is either used as a prefix or an infix to denote participles.

These suffixes partly include each other which leads to the observation that any declensional suffix can be build up from a combination of "e", "s", "n", "t", "em", "er", and "nd". By giving up some amount of correctness for speed the stemming algorithm described in this report recursively cuts of any occurrence of "e", "s", "n", "t", "em", "er" or "nd" from the end of the term to be stemmed. The additional error rate caused by this simplification is minimal but the gain in speed is significant as now only 8 different suffixes must be checked instead of 71.

After all suffixes have been removed any occurrence of "-ge-" either as prefix or suffix is cut off. By post processing terms in this way participles are reduced to their respective stem. This replacement is risky because it is the source for a number of 'different word - same discriminator' errors.

It was previously stated that the algorithm is context free. This is not completely true because there are four restrictions on suffix stripping which take the length and the case of a term into account:

- If a term is shorter than four characters no further characters are removed.
- If a term is shorter than five characters neither "em" nor "er" is removed.
- If a term is shorter than six characters no "nd" is removed.
- "t" is not removed from terms starting with an uppercase letter. This is because -t is a verb suffix and its removal is of no help for nouns which start with an uppercase letter in German.

One of the effects of this restrictions is that 2-letter words are ignored by the algorithm. As there are just a handful of such words in German the effect of ignoring them can be ignored.

The following table lists some terms and their respective discriminators as calculated by suffix stripping:

term	**discriminator**	**term**	**discriminator**	
singt	sing	singen	sing	
beliebt	belieb	beliebtester	belieb	
stören	stö	stöhnen	stöh	
Kuß	Kuß	Küsse	Küss	*error!*
Verlierer	Verli	Verlies	Verli	*error!*
Maus	Mau	Mauer	Mau	*error!*
stören	stö	Störsender	stö	*error!*

The top three rows give examples of correct stemming, the last four rows examples of incorrect stemming. Stemming errors as given in row four and five are prevented by the first part of the stemming algorithm - substitution. Errors as in line six and seven are the price for the algorithm's speed and simplicity.

2.2 Substitution

Just doing suffix stripping without taking any contextual and linguistic information into account leads to two main stemming errors:

- In German language many plural forms are built by changing a vowel to an Umlaut (ä, ö, ü) and/or replacing ß by ss. These changes of the stem cannot be handled by suffix stripping. The same holds for irregular verb forms (e.g. "halten - hielt").
- Character sequences that belong together because they form a single sound (e.g. "sch", "ei", and "ie") are split and (partly) cut off by the stripping algorithm.

To prevent these stemming errors some substitutions are done before suffix stripping:

1. Each *Umlaut* is replaced by its corresponding vowel and "ß" is substituted by "ss".
2. The second character of each double character is replaced by "*"
3. "sch", "ch", "ei", and "ie" are replaced by special characters ("$", "§", "%", "&").

Purpose of the first substitution is to conflate all plural forms correctly. Drawback of this substitution is that different words like "Stück" and "Stuck" or "Eisbär" and "Eisbar" are mapped onto the same discriminator. As the number of these errors is very low (only 12 of these pairs were found in a list of more than 50,000 nouns), there is no effort done to handle them. The second and third replacements are mainly done to save fixed character sequences from being split. Further more replacing "sch" and "ch" by single characters some more characters are saved from splitting at words starting with "sch" or "ch".

The algorithm can be improved by substituting other characters, too, e.g. "x" for "z" to handle irregular plural forms like "Matrix" and "Matrizen" (*matrix* and *matrices*).

The only requirement for any substitution is that it should either result in another morphological form of the same word or produce a non-word. If this requirement is not met a stemming error will occur. E.g. substituting "ß" for "ss" in "Buße" leads to "Busse" which is the plural form of "Bus". For this reason "Bus" and "Buße" are both stemmed to the same discriminator "Bus".

Substitutions are a very powerful instrument as they can capture linguistic rules and statistical heuristics. If - for example - "ge" within the character sequence "ige" can never be an infix denoting a participle, "ig" should be substituted in order to save it from being split. Most of these substitutions are rather short and can be implemented efficiently using transition tables or finite state machines.

3 Evaluation

The algorithm described in this report has been tested with a list of about 75,000 terms taken from the German version of the *Linux* spell checker. Additional tests have been performed with scientific texts and short stories

The two possible kinds of stemming errors are:

- two declensions of the same word are not conflated to the same discriminator (*understemming*)
- two ore more different terms are conflated to the same discriminator (*overstemming*)

Errors of either kind within a given text or wordlist sum up to the overall error rate of the stemming algorithm.

3.1 Understemming Errors

Even though the stemming algorithm tends to strip off more letters than necessary it may happen that not the whole declinational suffix of a term is removed. This kind of error occurs with three groups of words:

- Irregular verbs (and all of their compoundings) are not reduced to the same discriminator. The Duden lists 173 of these verbs including many of the most commonly used.
- Female forms of professions are in many cases not conflated to the same discriminator. E.g. "Schauspielerin" (*actress*) is not stemmed at all while "Schauspielerinnen" is conflated to "Schauspielerinn". The same holds for female inhabitants of many countries and cities (e.g. "Engländerin" vs. "Engländerinnen").
- Many words with a Greek or Latin origin are not conflated correctly. Examples for these kinds of words are "Drama" (Plural: "Dramen") and "Minimum" (Plural: "Minima").

The second kind of error can easily be prevented by substituting each occurrence of "erinn" with "erin". With the Linux word list it would reduce the amount of stemming errors by 21 without causing any additional errors. In Franz Kafka's novell "Der Prozeß" only one female form of a profession is mentioned (only in singular), and within all scientific texts about computer science and linguistics that have been checked no word of this kind occurred. For this reason it was decided not to do the "erinn" - "erin" replacement as the additional computational time is not justified by that minimum effect. Errors of the first and third kinds can only be prevented by using a dictionary.

In general it could be said that cases where two different morphological forms of the same word are stemmed to different discriminators are very rare for nouns and adjectives. Giving a qualitative estimate for the amount of understemming errors on verbs is difficult. The vast majority of all verbs are regular and easy to stem. But the few irregular verbs that cause understemming errors are very common in spoken and written German. In a wordlist just 5% of all verbs may be irregular but in a single text their amount could be 50% and more. Depending on the purpose the substitute-and-stem algorithm is used for, irregular verbs may cause an error rate somewhere between 1% and 10% (about 20% of all German words are verbs). As this error rate may not be acceptable a solution for the 'irregular verb' problem will be given at the end of this section.

3.2 Overstemming Nouns

The crucial aspect for the quality of context-free suffix stripping is the amount of overstemming errors, that is the number of different words that are conflated to the same discriminator.

The first experimental test was done on nouns by only removing the suffixes "e", "n", "s", and "er". The result was extremely positive: The Linux spell checker's wordlist contains 53300 nouns, some of them in both singular and plural forms. These terms were conflated into 50058 different discriminators by the stemming algorithm. Of these 50058 discriminators only 160 (0.32%) were not unique. Of the 53300 nouns only 325 (0.61%) were not stemmed to a unique discriminator. In one case 4 different words were stemmed to the same discriminator ("Buch", "Büchse", "Buchse", and "Büchner"), in 13 cases 3 different words were reduced to the same discriminator.

The 160 non-unique discriminators can be grouped as follows:

- with 5 of these errors at least one of the words was not German, e.g. "Cent" and "Center".
- 15 errors were caused by names, e.g. Albrecht Dürer's name was stemmed to "Dur".
- 58 errors occurred on words with overlapping stems. Examples are "Rind" and "Rinde" or "Eis" and "Eisen". In 32 of these cases the error was caused by stems ending on "er".
- 12 errors occurred on derived forms of the same stem. Examples for this kind of error are "Tanz" and "Tänzer".
- 12 errors were caused by Umlaut substitution. Examples are "Stuck" and "Stück" or "Bar" and "Bär". 34 more errors were not caused by but enabled by Umlaut substitution (e.g. "Körper" and "Korps" were both stemmed to "Korp").
- all remaining 27 errors were caused by pairs of words where the algorithm stripped letters from both words' stems. Examples are "Wien" and "Wiese" or "Hafen" and "Hafer".

These figures allow for a comparison of context-free suffix stripping with linguistic and dictionary based stemming algorithms:

Errors of the first two kinds (Non-German words and names) could occur as well with these more sophisticated approaches. Overlapping stems can only be handled if the gender of a word is known. Detecting gender is very difficult for German nouns, so many of the 58 errors produced by suffix stripping would as well cause problems with any other algorithm[2]. Only errors of the last three kinds - which are about 50% of all non-unique discriminators - are typical for suffix stripping.

[2] For example, the Porter stemmer conflates *general*, *generous*, *generation*, and *generic* to the same root. [Hull96]

None of the words stemmed to a non-unique discriminator was longer than twelve characters. More than 80% were even shorter than or equal to 6 characters. This proofs the assumption that the longer the words the fewer are the errors caused by suffix stripping.

With most single documents stemmed with the algorithm the amount of overstemming errors for nouns was about 0.5%. More than half of the non-unique stems were caused either by names or by derived forms of the same word. Once again the great majority of error-causing terms were shorter than seven characters.

3.3 Overstemming Verbs, Adjectives, and Nouns

The second test was performed on the same set of documents. In this case all suffixes listed in chapter 2 were removed. As weak stemming was desired identical discriminators were assumed to be different if their first letter was not of the same case (e.g. "kranker" is distinguishable from "Kranker"). As German sentences very seldom start with verbs or adjectives the amount of 'same word - different discriminator' errors only increased slightly. For the most representative test with the complete Linux wordlist the following results were obtained:

The size of the wordlist is about 75,000 terms of which about 70% are nouns. The stemming algorithm calculated 66959 different discriminators (50058 of them for nouns). The number of overstemming errors raised from 160 non-unique discriminators for nouns to about 510 non-unique discriminators (0.76%) for the whole wordlist. About 1600 words were detected that were not conflated to an unique discriminator (2.8% of all words). For verbs and adjectives 1.6% of all discriminators were not unique and 6.7% of all verb and adjective forms were stemmed to non-unique discriminators.

In a third tests all words were converted into lowercase after stemming. This scenario - called *medium stemming* - is close to strong stemming as many derived forms are reduced to the same discriminator as their stem. Nevertheless it is assumed that still weak stemming is desired but that there is no problem with conflating closely related terms to the same discriminator. For most retrieval and mining applications this should be the case as many of these 'errors' could only have been detected by very time-consuming linguistic and semantic analysis.

With medium stemming about 620 non-unique discriminators (0.93%) were detected for the whole wordlist. About 2100 words (2.1%) were not conflated to an unique discriminator. As with the previous test most of the errors were caused by verbs and adjectives.

3.4 Improvements

With none of the tests the amount of overstemming errors was above 3%. For most single texts it was at about 1%. With most tests the number of overstemming errors was lower than the number of understemming errors. The reason for this unexpected behavior is the high frequency of irregular verbs in written texts.

Irregular verbs van very easily be handled by extending the substitution part of the algorithm. The idea is to reduce all morphological forms of an irregular verb to one of these forms by using a small word list.

The common form should be the most significant character sequence of all forms. E.g. the common form for "kommen - kam" is "komm" while the common form for "laufen - lief" is "lief". Substitution is done on a substring base in order to handle compound words, too. By doing this simple substitution the verb "ankam" would be changed into "ankomm" and "zugelaufen" would be changed into "zugeliefen". As replacements of irregular verb forms in most cases lead to non-words the number of additional overstemming errors is very small.

By using a small wordlist the number of understemming errors can be decreased significantly. The tradeoff is a loss in time as about 200 additional substring comparisons have to be done for each term.

If hard stemming is required more suffixes should be cut off. Especially for the most common derivation suffixes ("-ung", "-heit", "-keit", etc.) the number of additional overstemming errors can be ignored.

4 Conclusion

In this report a stemming algorithm based on substring substitution and context-free suffix stripping is presented. Three variants of the algorithm have been tested:

1. When only nouns were considered the amount of stemming errors was very low (< 1%). It can be assumed that a maximum error rate of 1% is not worse than the error rates produced by more sophisticated algorithms based on linguistic analysis or word lists. The advantage of substitute-and-strip algorithm (beside its speed) is that it gets the better the longer the terms to be stemmed are.

2. When all kinds of words were considered the amount of overstemming errors raised up to nearly 3% depending on the stemming semantics. A significant number of additional errors were caused by understemming (mainly irregular verbs). This leads to an overall error rate between 4% and 15% depending on the texts to be stemmed.

3. By matching each term against a list of irregular verbs most understemming errors could be prevented. As the number of overstemming errors only barely increases the overall error rate can be assumed to be less than 5%.

For most applications in the area of information retrieval a stemming error rate of about 5% is assumed to be acceptable [vanRijsbergen79]. This handicap can easily be met by the substitute-and-strip algorithm.

The low error rate of the algorithm is very astonishing as it is very small and fast. From practical experiences and the tests presented in this report it looks as if this 100 lines-of-code algorithm is only hardly worse than other approaches which are either based on sophisticated linguistic methods or on wordlists with 50,000 and more entries.

Further more the algorithm is an ideal extension to wordlist based approaches. Theses systems tend to have error rates close to zero for short terms while the substitute-and-stem algorithm is nearly error-free for long terms. By combining both approaches a wordlist containing less than 10,000 entries should be sufficient to reduce the amount of stemming errors to less than 1% (plus noise).

The only drawback of the algorithm is the relatively high error rate for verbs and adjectives. Nearly 7% of all terms within these groups were not stemmed to unique discriminators. For this reason it should not be used for semantic analysis or other areas were verbs are extremely important. But for any other kind of applications (e.g. information retrieval or text comparisons) it is a good, small, and fast alternative to existing approaches. Simplified versions of the algorithm (e.g. without a wordlist) can be used for all kinds of interactive systems in order to allow users to type natural language sentences into entry fields and other dialog controls.

References

[Bray96] Bray, T., "Measuring the Web". In *Proc. 5th International World Wide Web Conference, WWW5*. Paris, May 6-10, 1996.
Available at http://www5conf.inria.fr/fich_html/papers/P9/Overview.html

[Caumanns98] Caumanns, J., "A Bottom-Up Approach to Multimedia Teachware". In *Proc. 4th International Conference on Intelligent Tutoring Systems, ITS-98*. San Antonio, August 1998.

[Dijkstra93] Dijkstra, T. & Kempen, G. *Einführung in die Psycholinguistik*. Bern, Göttingen: Verlag Hans Huber. 1993.

[Duden84] *Duden Grammatik der deutschen Gegenwartssprache*. Mannheim: Bibliographisches Institut. 1984.

[Hull96] Hull, D. A., "Stemming Algorithms: A Case Study for Detailed Evaluation". *Journal of the American Society for Information Science*. 47(1), pp. 70-84, January 1996.

[Paice96] Paice, C. D., "Method for Evaluation of Stemming Algorithms Based on Error Counting". *Journal of the American Society for Information Science*. 47(8), pp. 632-649, August 1996.

[Piotrowski98] Piotrowski, M. *NLP-Supported Full-Text Retrieval*. Master's Thesis, Universität Erlangen-Nürnberg. 1998.
Available at http://www.linguistik.uni-erlangen.de/~mxp/Magister/ma-as-report.pdf

[Porter80] Porter, M. F., "An Algorithm for Suffix Stripping". *Program* 14(3), pp. 130-137. July 1980.

[vanRijsbergen79] van Rijsbergen, C.J. *Information Retrieval*. London: Butterworth. 1979.
Available at http://www.dcs.glasgow.ac.uk/Keith/Preface.html

[Sheridan96] Sheridan P. & Ballerini, J.P., "Experiments in Multilingual Information Retrieval using the SPIDER System". In *Proc. 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 96*. Zurich, August 20-28, 1996.