Retrieval in Text Collections with Historic Spelling using Linguistic and Spelling Variants

Andrea Ernst-Gerlach  
Department of Computational and Cognitive Sciences  
University of Duisburg-Essen  
ernst@is.informatik.uni-due.de

Norbert Fuhr  
Department of Computational and Cognitive Sciences  
University of Duisburg-Essen  
norbert.fuhr@uni-due.de

ABSTRACT
We present a new approach for the retrieval of texts with non-standard spelling, which is important for historic texts e.g. in English or German. In this paper, we describe the overall architecture of our system, followed by its evaluation. Given a search term as lemma, we use a dictionary of contemporary German for finding all inflected and derived forms of the lemma. Then we apply transformation rules (derived from training data) for generating historic spelling variants. For the evaluation, we regard the resulting retrieval quality. The experimental results show that we can improve the retrieval quality for historic collections substantially.

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]: Search process, query formulation

General Terms
Historic documents, rule-based search, spelling variants

1. INTRODUCTION
Recently, we have seen a number of initiatives addressing the problem of digitising books and making them available on the Internet. Since these initiatives are focusing on books in English only — and mainly as a reaction to the Google digitisation initiative — the European Union plans to create a European digital library in order to preserve the culture of the European countries. With a European digital library project, the collections that are available in the Internet could be growing exponentially.

In contrast to countries with institutions defining spelling standards (e.g. Spain, France), English¹ and German (see [19]) spelling was not stable over several centuries. English spelling was more or less fixed around 1800. In contrast, German spelling was not standardised until 1901/1902. Before that date, there was the rule ‘write as you speak’ (phonological principle of spelling see [9]). Because of the various dialects and the variations over time, German spelling before 1900 was highly time- and region-dependent. But even for languages like French where the orthography has been standardised early, spelling variants are occurring (see [4]). Furthermore, the predominant part of the 6,000 contemporary spoken languages never became official languages and thus, they have never been standardised at all (see [19]).

The non-standard spelling produces problems when searching in the historic parts of digital libraries. Most users will enter a lemma (see Table 1) of the form that could be found in a dictionary as search term. The lemma for a noun is the nominative singular and for verbs the infinitive. This form differs from the historic inflected and derived forms of a lemma used in the documents. In order to solve this problem, our project deals with the research and development of a search engine where the user can formulate queries in contemporary language for searching in documents with an old spelling that is possibly unknown to the user. For this purpose, we are developing transformation rules for generating historic spellings from a given word.

¹http://en.wikipedia.org/wiki/English_spelling, access April 04 2007 11:05

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 2007 ACM 978-1-59593-644-8/07/0006 ...$5.00.
and will create a platform that supports the interactive, iterative development of new rules.

The paper has the following structure. In section 2, we give a brief survey over related work. Section 3 illustrates our approach for the search in text collections with non-standard spelling, outlines our work, and specifies the generation of rules for transforming words into their ancient spellings. The core of our approach is presented in section 4, where we describe the architecture of our historic search engine. The generation of word forms used in our search engine in combination with the rules is evaluated in section 5, and the last section concludes the paper and gives an outlook on future work.

## 2. RELATED WORK

Lately, several research groups started working on the problem of spelling variation. The approaches in this area can be divided into dictionary-based, distance measure and rule based-approaches.

Rayson et al. [18] describe a project for dealing with historic spellings of English. They developed a variant detector (VARD) for English texts from the 16th - 19th century. One major difference to our work consists in the fact that German is a highly inflected language, in contrast to English. Thus approaches developed for English can hardly be applied for German. Another important distinction is the motivation between those two approaches. VARD is aiming in the first instance for a high precision because it is used for normalising variants. In contrast our project is focusing on finding and highlighting all historical spellings in order to achieve a high recall (see [1]).

Koolen et al. [11] are describing the problem of spelling variants for Dutch. They developed the following three approaches for generating rewrite rules:

1. The **Phonetic Sequence Similarity** is using transcriptions of one-to-one mappings of historic and modern words. This approach classifies the words into sequences of vowels and consonants. New rewrite rules are built if the sequences of the modern and the historic word have a different spelling and according to the transcription the same pronunciation.

2. Like the previous approach the **Relative Sequence Frequency** approach divides words from the contemporary and the historic corpora into sequences of vowels and consonants. Based on words with sequences that occur exceptionally frequent in the historic corpora, rules are developed by replacing the sequence with a wild card and searching for the modern variant.

3. The **Relative N-gram Frequency** is a variant of the latter approach that uses n-grams instead of sequences of vowels and consonants.

In Kempken et al. [10] different distance measures have been evaluated for the application for text collections with spelling variants. They differentiate three categories of distance measures:

1. **Phonetic distance measures**, based on the evaluation Kempken et al. developed a trainable distance measure. Starting from contemporary words and their spelling variants it estimates the costs for the transformation of one letter into another. According to the Levenshtein Distance, the most similar word for a search term should be the one with the lowest transformation costs.

2. **Approaches for the correction of typing errors** and

3. **String similarity measures**.

Based on the evaluation Kempken et al. developed a trainable distance measure. Starting from contemporary words and their spelling variants it estimates the costs for the transformation of one letter into another. According to the Levenshtein Distance, the most similar word for a search term should be the one with the lowest transformation costs.

The topic addressed in this paper is related to the problem of approximate name matching (see [16]), where names with an incorrect spelling have to be found in a list of names. However, the major difference between the two problems consists in the fact that names usually differ only in their spelling, but not in their pronunciation. In contrast, words from historic texts may also differ in their pronunciation, mainly due to regional dialects (see [19]). These differences can also have effects on the spelling (see section 1).

The problem studied here is also somewhat related to cross-language information retrieval (see [15]), since in both cases mappings between words are considered. However, our problem can be solved by means of mappings at the grapheme level, while only dictionary-based approaches are suitable for cross-language information retrieval.

The approaches mentioned above have been developed for the languages English, Dutch and German which are all part of the West Germanic language family (see [2]) and thus part of the Indo-European languages. Additionally there are also research fields which are focusing on similar problems like cross-language information retrieval and approximate name matching.

We work on this problem with a rule-based approach, in order to be able to cover the complete vocabulary (and thus increase recall). On the other hand, the rules to be developed should be sufficiently precise, for distinguishing between spelling variants of the search term and other words.

## 3. SEARCHING IN TEXT COLLECTIONS WITH NON-STANDARD SPELLING

Since stemming needs a fixed set of rules to be known at indexing time and the studied language German requires rather complex rule sets, it would be very difficult to find the inflection rules which map the ancient spelling onto the associated contemporary radical. Thus we decided to generate search term variants at retrieval time. For this query expansion rules for inflections and derivations of words as

<table>
<thead>
<tr>
<th>Word</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lemma</td>
<td>basic form in a dictionary</td>
<td>geben (English: give)</td>
</tr>
<tr>
<td>Word form</td>
<td>(e.g. nominative singular for nouns and infinitive for verbs)</td>
<td>giebt</td>
</tr>
<tr>
<td>Variant</td>
<td>inflected or derived form of a lemma</td>
<td>giebt</td>
</tr>
<tr>
<td>Variant</td>
<td>historical word form for a given lemma</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Definitions
well as for handling spelling variants are required, thus we need a mapping

\[
\text{search term} \rightarrow \text{contemporary inflections (or derivations)} \rightarrow \text{spelling variants}
\]

This approach is more flexible than stemming, as new rule sets can easily be adopted.

Indeed, the stemming approach would be a lot faster than our approach, because the time-sensitive processing of the transformation happens once when a new collection gets indexed. However, we assume that rule sets for spelling variations will not be fixed for quite a long time and so only the second approach gives us the necessary flexibility.

In our approach, we first have to deal with morphological variations, before we can start constructing the rules for spelling variants. For this purpose, we are using a dictionary of contemporary German\footnote{http://wortschatz.uni-leipzig.de/} containing the word forms (see Table 2). Thus, when the user enters a search term (lemma), the dictionary yields all word forms. Afterwards we can generate the spelling variants of these forms.

By comparing the word forms of the dictionary with the word list of our corpus (or using a spell checker), we are getting a list of candidate words in non-standard spelling (some words also may not be contained in the dictionary, though). On the other hand, this method will not be able to detect homographs (in our case especially ancient spelling that matches a different contemporary word); this issue will be addressed at a later stage of our project. This way, we get a list of candidate words from our corpus. Then, we have to check manually if the words are really in a non-standard spelling, and have to assign the equivalent words in the contemporary standard spelling. Afterwards, we can focus on the second step — the building of new rules.

Even though the studied language is German our methods are applicable to other languages as well; e.g. from Rayson et al. (see [18]) we can build similar examples for English: always — alwaies (y → ie), sudden — suddain (e → ai), and publicly — publicily (c → ke).

In the following we list some example rules developed manually for the 19th century German (see Table 2).

<table>
<thead>
<tr>
<th>Contemporary spelling</th>
<th>19th century</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>wiedergaben</td>
<td>wiedergaben</td>
<td>wieder → wider ic → i</td>
</tr>
<tr>
<td>akzeptieren</td>
<td>acceptieren</td>
<td>kz → ic</td>
</tr>
<tr>
<td>überall</td>
<td>ueberall</td>
<td>ü → ue</td>
</tr>
<tr>
<td>seht</td>
<td>secht</td>
<td>t → et</td>
</tr>
</tbody>
</table>

Table 2: Example rules for German

The first example shows two rules at different levels of specialisation. The first rule transforms a prefix whereas the second one only transforms an allograph (see [4]). The next example also offers two possibilities. In this case the transformation can consist of one rule or the concatenation of two rules. However it becomes apparent that the precision of the first rule would be much higher than that of the second rule. The third case shows a very common rule for umlauts. The last example contains a very general rule, but it could reach a higher precision if the rule would be connected with context information (in this case the end of the word). So not only the transformation itself is important, but also the associated position.

Even though our approach requires a substantial manual effort at the beginning, we expect that only little additional work is required later when the collection is growing continually — due to the fact that we are working at the grapheme level.

The automatic rule generation method (described in detail in [6]) starts with a training sample of historic texts, on which we run a spell checker for contemporary German. For all words marked as incorrect spelling, the contemporary word form has to be assigned manually; furthermore, we determine the number of occurrences of each historic word form. Thus, we have sets of triplets containing the contemporary word forms, their historic spelling variant and the collection frequency of the spelling variant.

First, we compare the two words and determine so-called 'rule cores', the necessary transformations and also identify the corresponding contexts. For example, for the contemporary word form unmutz (= useless) and the historic word form unnuts, we would get the following 2-element set of rule cores: (unn(ü→u)ut), (t(z→s)).

As a second step, we generate rule candidates for each rule core that also takes account of the context information (e.g. consonant (C) or word-ending ($) of the contemporary word. If we use the example shown above, we find that among others the following candidate rules are generated: ü→ u, nni→ nu, iit→ ut, niit→ nut, Cü→ Cu, z$→s$.

Finally, in the third step, we select the useful rules by pruning the candidate set (where we are taking the collection frequency into account) with a proprietary extension of the PRISM algorithm (see [5]).

4. LAYERED ARCHITECTURE OF THE HISTORIC SEARCH ENGINE

Figure 1 illustrates the layered architecture of our system. Layer 4 is the user interface. In this layer the user inserts lemmas as search terms. The following layer is responsible for generating the word forms of the lemmas. Layer 2, the core of our historic search engine, transforms the contemporary word forms of the lemmas into the corresponding historic forms. Finally, layer 1 uses a standard search engine for finding the historic word forms in the bottom layer the digital library. Due to its flexible structure, our approach can also be combined with other search engines by plugging in layers 2 and 3 of our system. In the following, we describe layers 1–4 in more detail.

4.1 User Interface

In the user interface (see Figure 2) the user can type in the search terms as lemmas. On these lemmas the search for spelling variants is applied in the following layers.

Additionally the user can enter a time span and a location. This information can be used for a search with the corresponding time- and location-specific rule sets to be selected by the system, in case the metadata of the scored documents are not including the time or location informa-
Figure 1: Architecture of the historic search engine

tion. Otherwise if these attributes are available, the system will be able to select the suitable rule sets for the different document sets in the digital library independently.

We plan to generate time and location specific rule sets based on the data of the GermanC project from the University of Manchester. Within this project a representative historic corpus of written German for the years 1650 - 1800 is build. The corpus will be build from newspaper articles from five different regions and three different time periods. There will be a markup of grammatical features for the texts following the TEI (Text Encoding Initiative). Based on this features it could also be possible to build more specialised rule sets e.g. for the part of speech. Furthermore, the user can specify the term precision for the search terms. Thus, the user can influence recall and precision of search results: If he wants a high recall he can choose a lower threshold for the term precision, which allows the system to generate more spelling variants per word. Otherwise, if he is more interested in a high precision he can choose a higher precision value, thus a restricted number of spelling variants are generated. Particular with regard to this input field we will conduct user studies in order to find out if this field should be available within the normal search, for a reformulation of the query or an advanced search.

The system will be able to give a preview of the generated variants. In this preview the user can select the relevant search terms. Otherwise if the preview does not show satisfying results he can reformulate his query based on the preview.

4.2 Building word forms

For generating the word forms we use the German Vocabulary database (see [14]). It has been built automatically from newspapers and scientific journals which were available in a machine readable format. It includes over 3.7 Million word types which are stored together with collocations, grammatical information like inflections and stop words. References to the word forms are also included. Even though algorithms for error corrections are used, the list

3http://www.llc.manchester.ac.uk/Research/Projects/GerManCproject/
4http://www.tei-c.org/

with the word forms has an error rate of 1-2 % (see [14]). These errors are mainly based on spelling errors in the source documents.

The building of inflections is based on two methods:

- Inflections included in the database: The first method assumes that all inflections are included in the database. Therefore the possible word forms are built for a lemma e.g. with typical suffixes. If the result is included in the dictionary it is assumed that a new inflection has been built.

- Inflections of similar words (relating to their structure) are built in a similar way: The second method assumes that similar words form their inflections in a similar way. This method is mainly used for compounds. The class of inflections of a compound is based on its last part. Where it is possible to detect the correct last part of the compound the inflections can easily be adopted from the database.

The whole process of inflection building is also accompanied by error correction algorithms.

An alternative to the vocabulary database would be the usage of a search service like e.g. Canoo which also provides word forms. Short tests with the online services looked promising, but unfortunately it was not possible to get a research license.

4.3 Rule application

We generated a set of probabilistic rules as described before, which can be applied in our search engine. For a contemporary word \( a \), we want to generate all historic spellings. Thus, wherever a rule \( r_1 \) matches, it is applied to \( a \), thus yielding the word \( a_1 \). This way we are generating a set of historic spellings for a single word \( a \) by application of single rules. Obviously, these word forms are not all equally precise. Therefore, these words should be assigned weights

5http://www.canoo.net/
which reflect the precision of the rules they resulted from (see [20]).

### 4.4 Search engine

Variants can be considered for ranking the retrieval results.

For the search component we use the probabilistic IR engine PIRE (see [12]). PIRE is based on probabilistic Datalog which is a probabilistic extension of predicate Horn logic and uses probabilistic methods for indexing and retrieval. The usage of logic combined with probabilistic theory offers the possibility to define and utilise new data types (e.g. names, numbers) and various weighting schemes. Additionally the retrieval function is not fixed. These features make PIRE rather flexible for an embedding into our system.

For our application studied here, it is important that PIRE allows for probabilistic weighting of search terms: this way, the impression following from the generation of search term variants can be considered for ranking the retrieval results.

### 4.5 Search example

In Figure 3 shows an example for the search. The user enters the search term ruhen into the user interface. The next layer generates for this search term among other the word forms geruht, ruhte, ruhn, ruhen, ruhten. For each of these word forms historic word forms are generated e.g. for the word form geruht the variant geruhtet is generated by application of the rule \( t \rightarrow et \). All generated historic word forms are expand the query within the search engine and the resulting query retrieves the documents.

### 5. EVALUATION

For evaluating our approach, we are regarding the set of historic word forms generated for a query term (given as a lemma). As text collection, we used documents from the Nietzsche collection and other smaller collections. The texts are from the 15th to the 19th century. Our collection contains 534827 word tokens (total number of words), with 51721 types (total number of different words). Based on a small training set of 478 evidences we generated 65 rules. In Table 3 some frequently used are shown. The statistics for the collection can also be found in Table 4.

<table>
<thead>
<tr>
<th>Table 4: Collection statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>texts</td>
</tr>
<tr>
<td>word tokens</td>
</tr>
<tr>
<td>word types</td>
</tr>
<tr>
<td>training set</td>
</tr>
<tr>
<td>rules</td>
</tr>
</tbody>
</table>

For our evaluation of generating word forms we built two test sets. For the first set, we randomly selected word types from our collection. With this test set, we can simulate the complete historic search engine. For the second test set, we only chose historic spellings from our collection, in order to examine the quality with regard to the spelling variants only.

In order to evaluate the quality of our approach, we are not performing retrieval experiments in the usual way. This approach would have required a rather big effort, and it also would have been very difficult to draw conclusions with regard to our research focus. Instead, we are regarding one-term queries only, and then measure a) the proportion of retrieved word forms belonging to the query term and b) the proportion of occurrences of all word forms retrieved. In the following, we call occurrences of word forms belonging to the query terms ‘relevant tokens’. Then we can define appropriate variants of the precision and recall measure:

\[
\text{precision} = \frac{|\text{relevant tokens} \cap \text{found tokens}|}{|\text{found tokens}|} \quad (1)
\]

\[
\text{recall} = \frac{|\text{relevant tokens} \cap \text{found tokens}|}{|\text{relevant tokens}|} \quad (2)
\]

As with the standard recall measure, we also have to cope with the problem that it is hardly possible to identify all relevant tokens in the collection. In order to avoid this problem, we use a technique similar to the document-source method for estimating the recall of a retrieval system: For a random set of tokens from our collection, we construct the lemma manually, enter it as search term and then check if the original token is retrieved. The proportion of tokens retrieved this way is our recall estimate.

As a baseline for this evaluation we took the well-known engine Lucene. Lucene is an open source full text search engine library. For building the index with Lucene we employ the German variant of the Snowball stemming algorithm. We use this combination as a baseline for the German vocabulary database. For the second step — the search for spelling variants — we used the Levenshtein distance as a baseline. The Levenshtein distance is a similarity measure that has been developed for string comparisons. It calculates the minimum number of basic transformations like e.g. insertion, deletion and substitution for mapping one string on another. In order to facilitate the comparison we combined the stemming approach as well as the vocabulary database with the rule application and the Levenshtein distance. For our experiments we regarded a maximum Levenshtein distance of 1 and 2, respectively. The corresponding recall and precision values for the word forms and for the spelling variants can be found in Table 5.

### 5.1 Evaluation of words from the whole collection

First we took all word types from our collection excluding those from the stop word list from the German vocabulary database. The remaining types have an overall frequency of 300532.

---

6http://lucene.apache.org/

7http://snowball.tartarus.org/
For our evaluation, we formed a representative sample of 200 word forms. Thereby the frequency distribution of types in our sample is similar to that of the whole collection (assuming that the occurrence frequency of terms in queries would be similar to their collection frequency).

For these words the lemmas are built manually. This step can not be done by using the database, because we are working with historic collections so the word forms from the text also include historic forms. Given the lemmas, we retrieve all word forms of the lemma from the vocabulary database. For these forms, the correctness was analysed manually. So, the generated word forms are only marked as incorrect if they are real false forms, which belong to a different lemma, and not only include spelling errors that could be historic spelling variants. This step is followed by the transformation into historic spellings; for each variant generated this way, we check if it occurs in our collection, and if it does, whether or not it belongs to the same lemma. These results are compared with the stemming that is done by the stemming algorithm.

5.1.1 Conventional search engine

In order to simplify the classification of the evaluation results for the words from the whole collections we have also calculated recall and precision for a conventional search engine. This engine would only find those types which occur as a lemma in the text. Obviously we are reaching a precision of 1 since the search terms are directly found in the text without any transformation. However, the recall would only be 0.36 — that means, we would miss almost two out of three occurrences of a search term!

5.1.2 Stemming

By using the stemming algorithm in the search engine we are getting 3394 different word forms for the lemmas. This results in a recall of 0.82 and a precision of 0.94. These are already quite good results even if we miss nearly one of five words.

5.1.3 Generation of word forms

For the test set of 200 types 1760 different word forms of the lemmas were generated. So we get in average 8.8 word forms for each lemma. Only 10 of the 1760 word forms were assessed as not correct and only two of them are actually occurring in our documents. The generated word forms reach an overall frequency of 96582 and the incorrect forms have a total collection frequency of 121. Thus we reach a precision of 1.0.

For 11 types no word forms were found. For another 28 types the word forms from the test set are not found in the German Vocabulary database. Although we are only finding 161 types (81 %) of the original word forms we are getting a recall of 0.91 based on the term frequency in our collection. Thus, we are cutting in half the number of word forms not retrieved, in comparison with the stemming approach.

5.1.4 Levenshtein distance

Combined with the stemming approach, the Levenshtein distance 1 reaches a recall of 0.98 which is a very good result. But the precision is 0.49, so every second found occurrence is incorrect. In combination with the Vocabulary database with a recall of 0.40 and a precision of 0.84 the results are the other way round.

By setting the maximum Levenshtein distance to 2 and combining it with stemming, recall improves slightly from 0.98 to 0.99. For the combination with the vocabulary database recall is increasing from 0.40 to 0.75, i.e. it is almost duplicated. Due to the high number of generated historic word forms (26112 for the stemming approach and 4687 for the vocabulary database) the precision increases dramatically to 0.06 and 0.21 respectively.

5.1.5 Rule application

For the combination of stemming and rule application we get a recall of 0.89 and a precision of 0.77. Thus the Levenshtein distance achieves a higher recall in contrast to the rule
application which reaches a higher precision in combination with the stemming.

After applying our rules on the generated word forms of the lemma from the vocabulary database, we can reduce the number of not found word forms from 39 to 17. This way, we are reaching a recall of 0.99, which is a very good result. This indicates that we generate almost all frequent word forms occurring in the collection. The only drawback is the reduced precision, which drops from 1.0 to 0.89. However, this result is mainly due to the preliminary nature of the transformation rules employed for this evaluation. This combination also reaches the best recall and precision values of the combined methods regarded here. Here we can also see the relation between rule application and the number of retrieved word forms. For both approaches we achieve roughly the same increase in recall when we apply the transformation rules.

5.2 Evaluation restricted to historic forms

For this evaluation, we took pairs of contemporary word forms and their corresponding historic spelling. 17 % of the tokens in our collection are in historic spelling. We choose the test set we already used in [6] in order to facilitate comparisons. This test set contains 239 different word types. In this case also the lemmas are built manually and afterwards the word forms of the lemma are generated and analysed. After this step the historic forms are generated by the application of our rules.

5.2.1 Stemming

With the stemming we get 202 word forms for the search for spelling variants. On average, we get 0.85 word forms per lemma. Thus several lemmas could not be found with the stemming algorithm. On the other hand stemming reaches a perfect precision, because every found word form was correct. The recall of 0.05 indicates clearly that the stemming algorithms are not able to stem historic word forms.

5.2.2 Generation of word forms

For this test set we get 7.7 word forms per lemma on average. We achieve a precision value of 1.0 because none of the 4 generated false word forms is occurring in our collection. Similar to the first test set, for only 77 % of the word types the given word forms in contemporary spelling are generated.

5.2.3 Levenshtein distance

When restricted to the historic forms the Levenshtein distance 1 comes off badly. It reaches a recall of 0.24 using stemming and a recall of 0.35 based on the vocabulary database. Thus it misses three out of four (or two out of three, respectively) occurrences of a spelling variant. Also the precision with 0.41 for the stemming and 0.35 for the vocabulary database does not score well.

For a maximum Levenshtein distance of 2 the recall is increasing from 0.25 to 0.45 for the search on the stemmed index. For the search on the vocabulary database the recall is increasing in a similar amount from 0.35 to 0.52. For the whole test set we are generating here 2389 spelling variants for the stemming approach respectively 2167 spelling variants for the vocabulary database. Once again we receive a large drop in precision for the Levenshtein distance of 2 to 0.11 and rather 0.10.

5.2.4 Rule application

Even though stemming does not retrieve various word forms, the rule application reaches a recall of 0.71. For the found variants with a occurrence frequency of 3906 this combination reaches a perfect precision of 1.0.

For the historic forms we reach a precision of 0.93 and a recall of 0.7. The recall decreases in comparison to [6] where we reached a recall of 0.88; this difference is due to the fact that our previous experiments were restricted to pairs contemporary word forms — historic word forms, whereas we are now regarding to triplets lemma — generated word forms — historic word forms; that is, we are also considering the effect of the dictionary.

Overall, we can say: we are missing 23 % of the contemporary word forms from the test set; however recall only decreases by about 18 %. So also word forms of the lemma which are not identical to the original word forms are useful for searching spelling variants.

In comparison to the combination of stemming and the rule application the precision with 0.93 is decreasing slightly with a nearly constant recall. However, the retrieved variants have an occurrence frequency of 13198. Thus we can find more than three times as many as in combination with the stemming approach.

5.3 Discussion

The two series of experiments with words from the whole collection and historic word forms showed that our approach can be applied successfully for the retrieval of historic texts. Due to the high degree of inflection of the German language,

<table>
<thead>
<tr>
<th>Approach</th>
<th>Terms from the whole collection</th>
<th>Restricted to historic forms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall</td>
<td>Precision</td>
</tr>
<tr>
<td>Conventional search engine</td>
<td>0.36</td>
<td>1.00</td>
</tr>
<tr>
<td>Stemming</td>
<td>0.82</td>
<td>0.94</td>
</tr>
<tr>
<td>Vocabulary database</td>
<td>0.91</td>
<td>1.00</td>
</tr>
<tr>
<td>Stemming + Levenshtein 1</td>
<td>0.98</td>
<td>0.49</td>
</tr>
<tr>
<td>Stemming + Levenshtein 2</td>
<td>0.99</td>
<td>0.06</td>
</tr>
<tr>
<td>Stemming + Rule application</td>
<td>0.89</td>
<td>0.77</td>
</tr>
<tr>
<td>Vocabulary database + Levenshtein 1</td>
<td>0.40</td>
<td>0.85</td>
</tr>
<tr>
<td>Vocabulary database + Levenshtein 2</td>
<td>0.75</td>
<td>0.21</td>
</tr>
<tr>
<td>Vocabulary database + Rule application</td>
<td>0.99</td>
<td>0.89</td>
</tr>
</tbody>
</table>

Table 5: Recall and precision for word forms from the whole collection and restricted to historic forms
a method for dealing with this phenomenon is essential for achieving a satisfying retrieval quality. However, in historic texts, about 10% of all word occurrences will be missed when historic spelling is not taken into account. Our transformation rules can reduce this loss drastically, but at the expense of a small decrease in precision. It would be interesting to see how much a combination of the stemming and the vocabulary database could improve the recall of the rule application.

Furthermore, the experiments show that the stemming algorithm does not work well for historic words. For the Levenshtein algorithm, the best results are achieved in combination with the stemming algorithm on the words from the whole collection. Here this method yields recall values comparable to the combination of the vocabulary database with the rule application. However, the evaluation restricted to the historic forms showed that the Levenshtein distance is not competitive to the rule application.

Overall we can say that the rule based approach shows a great stability. The Levenshtein distance is only achieving comparable recall results for the terms from the whole collection in combination with the stemming algorithm. However, in this case we have to consider that the retrieved word forms of the stemming algorithm directly increasing the recall, since these variants are included in the text. In contrast, many word forms generated from the vocabulary database do not occur in the text, and so there is still a transformation necessary that generates the historic forms.

6. CONCLUSION AND FUTURE WORK

In this paper, we have discussed the problem of retrieval in historic texts with non-standard spelling. Due to the large variations in historic spellings, the standard stemming approach is not promising. Instead, historic variants of the search terms have to be generated. By using a contemporary dictionary containing the word forms, we first map the search term onto its word forms, and then generate the corresponding historic variants before a standard search engine can be used.

The evaluation has shown that our transformation rules, in combination with generating word forms of the lemma via the German Vocabulary Database, gives very good results. As with all dictionary-based approaches, it is not possible to generate all word forms; however, nearly all generated forms are correct.

As the stemming showed very good precision results it could be used twofold as an additional technique. First it can be used in the way as it is already employed in this paper for a search on the index of documents that has been built through stemming. A second possibility would be the building of an index by stemming the word types of a contemporary corpus and an utilisation of this data as additional source for word forms. In order to further reduce of bottleneck, beside other similar tools to the vocabulary database, part of speech taggers (e.g. connexor) that also return the corresponding lemma of a given word in combination with contemporary corpora could be used. By applying the tagger onto the word types of the corpora we could extend the German vocabulary database very easily and thus increase the number of the word forms for a lemma. With a parallel search on the German Vocabulary database, an index with stemming and an additional index of a word list of a contemporary corpora, this bottleneck is supposed to be removed.

Another bottleneck of this approach are the manually collected word pairs of contemporary inflected and derived words and the corresponding historic variant which are needed as input for the training data. In a later stage of the project these word pairs should be developed automatically.

The manual analysis of the errors on the corresponding test set has shown that a large number of errors is caused by a few words. Thus, we will try to reduce the errors via statistic analyses. Alternatively these words should be considered as exceptions in the rules — so the rule generation algorithm should be modified accordingly.

The next step is a further optimisation of the automatic rule generation. So far, our method has been applied to German texts only. However, by exchanging the linguistic resources (i.e. the dictionary containing the full word forms) it can be easily applied to other languages. A small test in [1] showed promising results.

7. ACKNOWLEDGEMENTS

This work is supported by the DFG (project RSNSR).

8. REFERENCES


