A Cross-Language Approach to Historic Document Retrieval

Marijn Koolen[1,2], Frans Adriaans[1,3], Jaap Kamps[1,2], and Maarten de Rijke[1]

[1] ISLA, University of Amsterdam, The Netherlands
[2] Archives and Information Studies, University of Amsterdam, The Netherlands
[3] Utrecht Institute of Linguistics OTS, Utrecht University, The Netherlands

Abstract. Our cultural heritage, as preserved in libraries, archives and museums, is made up of documents written many centuries ago. Large-scale digitization initiatives make these documents available to non-expert users through digital libraries and vertical search engines. For a user, querying a historic document collection may be a disappointing experience: queries involving modern words may not be very effective for retrieving documents that contain many historic terms. We propose a cross-language approach to historic document retrieval, and investigate (1) the automatic construction of translation resources for historic languages, and (2) the retrieval of historic documents using cross-language information retrieval techniques. Our experimental evidence is based on a collection of 17th century Dutch documents and a set of 25 known-item topics in modern Dutch. Our main findings are as follows: First, we are able to automatically construct rules for modernizing historic language based on comparing (a) phonetic sequence similarity, (b) the relative frequency of consonant and vowel sequences, and (c) the relative frequency of character n-gram sequences, of historic and modern corpora. Second, modern queries are not very effective for retrieving historic documents, but the historic language tools lead to a substantial improvement in retrieval effectiveness. The improvements are above and beyond the improvement due to using a modern stemming algorithm (whose effectiveness actually goes up when the historic language is modernized).

1 Introduction

Natural languages evolve over time. In Europe, almost all languages are part of the Indo-European language family [25]; they have evolved gradually, changing in pronunciation and spelling. To a large extent, our cultural heritage, as preserved in libraries, archives, and museums, consists of documents written many centuries ago. Many cultural heritage institutions are currently exploring ways of digitizing their document collections [13], which has resulted in a number of collaborative projects on digital cultural heritage, including DigiCULT [7].

Having digital versions of old, fragile documents is a good way of preserving them, and it makes them easily accessible to a multitude of users over the web. Browsing through such documents, one can probably recognize the language they are written in as a historical variant of a modern European language, although

M. Lalmas et al. (Eds.): ECIR 2006, LNCS 3936, pp. 407–419, 2006.

one may run into significant differences with today's spelling rules and vocabulary. What if a vertical search engine is created that gives access to the historic documents? One that "knows" about changes in spelling and vocabulary? Consider a user that searches for 300 year old documents about, for instance, Central European politics. Her query will consist of modern words, and hence will not be very effective for retrieving documents containing historic terms. In order to make historic documents accessible to modern users, our vertical search engine should be able to bridge the gap between the historic language of the document and the modern language of a user's query. This is the focus of the present paper.

We define Historic Document Retrieval (HDR) as the retrieval of relevant historic documents given a modern query. Earlier research [18, 17, 3] dealt with spelling differences between a modern language and a historical variant in a collection of old documents. We continue research on this spelling problem and propose a cross-language approach to HDR. Using historic Dutch as a concrete example, we argue that the gap between modern and 17th century Dutch is substantial, and that effective retrieval therefore requires a mapping between the two languages.

A cross-language approach to HDR raises a number of questions. Manually constructing historic language tools is an unattractive option, because of the large number of spelling variants. These variants are caused by the absence of strict spelling rules and by various regional differences. Is it possible to automatically construct translation resources for historic languages? In Cross-Language Information Retrieval (CLIR, [2]), stemming algorithms have proved to be effective in modern monolingual retrieval. Are these also effective in HDR? Our research identifies HDR as a new cross-language IR problem and consists of two main parts. The first is the construction of resources for our CLIR approach to HDR. We have developed tools to automatically construct translation resources. In the second part we test the effectiveness of these translation resources on historic documents in a CLIR experiment. Since these methods are data-driven, they can be straightforwardly applied to new HDR problems.

The article is outlined as follows: Section 2 discusses Historic Document Retrieval, and details the historic documents used. Section 3 describes the automatic construction of translation tools for historic languages. Then, Section 4 focuses on HDR proper, and evaluates the effectiveness of the constructed translation tools. Finally, in Section 5 we discuss the results and our main findings.

2 Historic Document Retrieval

Robertson & Willett [17] tested spelling correction methods to find historic variants of modern word-forms in historic English documents. They also tested the effectiveness of a list of manually constructed phonetic substitutions to preprocess historic words before applying the spelling correction methods to see if preprocessing decreases the gap between 17th century and modern English. For instance, the phonetic substitution $YGHT \rightarrow IT$ (**ME**), replaces all occurrences of *yght* in the middle or at the end of a word to *it*. They find that preprocessing has very little effect. However, the spelling correction methods themselves

are very effective in finding historic word-forms. The use of the same correction techniques on old French confirmed these results [16].

Braun [3] tested the effectiveness of preprocessing 17th century Dutch documents by applying rewrite rules in a document retrieval experiment. Rewrite rules take a sequence of characters out of a word and replaces it with a new sequence. After rewriting, historic words, especially their pre- and suffixes, are closer in spelling to modern Dutch words, making the Dutch variant of the Porter Stemmer more effective. Rewrite rules are thus an effective way of decreasing the spelling gap between 17th century and modern Dutch.

The main problem with the rewrite rules in [17, 3] is that manual construction takes a lot of time and requires intimate knowledge of the specific historic language. Moreover, the rewrite rules for 17th century Dutch work for 17th century Dutch, but probably not for 17th century English, nor for 14th century Dutch. In the next section we propose data-driven methods for constructing rewrite rules using only a historic and modern document collection. Because of their data-driven nature, these methods can be applied to historic and modern document collections in other languages as well. Their output—sets of rewrite rules—can be used to construct translation dictionaries for a specific historic document collection, thus providing the resources required for our CLIR approach to HDR.

As mentioned before, we take historic Dutch as a case study. Dutch cultural heritage institutions possess large collections of old books, newspapers and other documents, and many of these are written in historic variants of modern Dutch. Uniformity in the Dutch language is a relatively new phenomenon. *Middelnederlands* is a predecessor of the modern Dutch language that was spoken during the Middle Ages and can best be thought of as a collection of different dialects. Moreover, whereas modern Dutch spelling is based on strict spelling rules, spelling in *Middelnederlands* was based on pronunciation [11]. Since pronunciations can have different orthographic representations, spelling was highly inconsistent. Each region had its own pronunciation, and hence its own spelling conventions. Although the Dutch language became more uniform in the 17th century, mainly through the nation-wide use of the first official Dutch Bible translation, the lack of spelling rules still resulted in the occurrence of many spelling variations throughout documents.

In our tool development and retrieval experiments we use the same historic corpus as [3]. It contains two 17th century collections of legal texts: the *Antwerpse Compilatae* (1609) and the *Gelders Land- en Stadsrecht* (1620). Although they are written in different dialects (southern and eastern, respectively), they are written in the same legal idiom. Hence, they contain many technical law terms and long sentences. This contrasts with other idioms from that period, such as the language of sailors. The *Antwerpse Compilatae* are part of a collection of legal texts called the *Costumen van Antwerpen* [1]. The collection consists of four parts: the Anitiquissimae (1547), the Antiquae (1571), the Impressae (1582), and the Compilatae (1609). OCR errors were manually corrected. Each section was treated as a separate document. This resulted in 222

documents. The *Gelders Land- en Stadsrecht* [9] is very similar to the *Antwerpse Compilatae*. It contains parts on the same subjects as the Compilatae, with the only difference that substantive and formal criminal law are covered in one part. This collection was digitized by manually entering the text into the computer. The *Gelders Land- en Stadsrecht* collection contains 171 documents.

To clarify the differences between the historic language of the corpus and modern Dutch, we took a random sample of 500 words from the historic collection. Each word was assigned to one of three categories: modern, spelling variant, or historic. The overlap between historic and modern Dutch is significant (177 words, 35%). These words are spelled in accordance with modern Dutch spelling rules. An example is the word *ik* (English: I) which is often found in historic texts, but it has not changed over time. It turns out that most of the words (239 words, about 48%) are historic spelling variants of modern words. These words can still be recognized as a modern word, but are spelled in a non-modern way. An example is the word *heyligh* which is easily recognized as a historic spelling of the modern word *heilig* (English: holy) The remaining words (84 words, 17%) have a non-modern morphology, or cannot be recognized as a modern word at all. An example is the word *beestlijck*. Even adjusting its historic spelling, producing *beestelijk*, it is not a correct modern Dutch word. Taking a look at the context makes it possible to identify this word as a historic translation of the modern word *beestachtig* (English: bestial or beastly).

All documents in our collection were transformed into the standard TREC document format, resulting in 393 documents, with an average length of 912 words. In total, the corpus contains 17,794 distinct word tokens. We created a topic set consisting of 25 modern Dutch known-item topics. The topic creators are non-experts in the field of historical law texts, and are therefore unfamiliar with specific historical law terms. This is an important criterion for a HDR topic set, since this leaves the linguistic difficulties to the system. Here is an example topic (description field):

(Q25) *Welke methoden zijn geoorloofd ter ondervraging van gevangenen?*
(English: *Which methods are allowed in the interrogation of prisoners?*)

Topics were given the familiar TREC topic format: title, description, and narrative.

3 Tools for Historic Document Retrieval

Our goal is to design algorithms for mapping historic spelling variations of a word into a single modern form. Below, we describe three tools for creating rewrite rules for 17th century Dutch. One of these tools exploits the phonological overlap (i.e., how words are pronounced) to find sequences that are spelled differently but sound the same. The other two algorithms exploit the orthographical overlap in a historic and a modern word to find the most probable modern version of a historic sequence. These two algorithms use *only* a corpus of 17th century Dutch documents and a corpus of modern Dutch documents. Since the corpus described

in the previous section is rather small, for the construction of the rewrite rules, the corpus was expanded with a number of 17th century literary works taken from the DBNL [6]. To make sure that they were written in roughly the same period as our main corpus, we used texts written between 1600 and 1620. The literary texts from DBNL are not suitable for a document retrieval experiment because it is hard to determine the topic of such a text, but they do contain the same spelling variations, so they can be used for devising rewrite rules. As a modern corpus, we used the Dutch newspaper *Algemeen Dagblad*, part the Dutch corpus of CLEF [4].

The three techniques are related to spelling correction techniques such as isolated-word error correction [12, 20]. Since the context of a historic word is also historic text, context dependent (semantically or syntactically informed) error correction techniques are no option in the case of cross-language HDR.

PSS. The Phonetic Sequence Similarity (PSS) algorithm finds historic spelling variants of modern words by comparing the phonetic transcriptions of historic and modern words.[1] If the phonetic transcription of a historic word W^{hist} is equal to the phonetic transcription of a modern word W^{mod}, but their spelling is different, then W^{hist} is a spelling variant of W^{mod}. All words in the historic corpus and the modern corpus are converted to phonetic transcriptions by the grapheme-to-phoneme converter tool in NeXTeNS, a text-to-speech generation system for Dutch [15]. W^{hist} and W^{mod} are then split into sequences of vowels and consonants, and these sequences are aligned. If sequence S_i^{hist} is orthographically different from sequence S_i^{mod}, S_i^{hist} is a historic spelling variant of S_i^{mod}. The resulting rewrite rule is: $S_i^{hist} \rightarrow S_i^{mod}$.

Take the following example. The 17th century Dutch verb *veeghen* (English: to sweep) is pronounced the same as its modern counterpart *vegen*; their phonetic transcriptions are both *v e g @ n* according to Nextens. Splitting both words into sequences and aligning these, results in:

historic:	v	ee	gh	e	n
modern:	v	e	g	e	n

At positions 2 and 3 we find differences between the historic sequences and the modern counterparts. This results in the rewrite rules: $ee \rightarrow e$ and $gh \rightarrow g$. Each rewrite rule is assigned a value N, where N is the number of times the rule was generated. If N is high, there is a high probability that the rule is correct.

RSF. The Relative Sequence Frequency (RSF) algorithm exploits orthographic overlap between historic and modern words, to construct rewrite rules for the parts that do not overlap. First, all words in the historic corpus are split into sequences of consonants and sequences of vowels. An index is made, containing the corpus frequency $F(S_i^{hist})$ of each unique sequence S_i^{hist}. The same is done for all words in the modern corpus.

[1] This requires a tool to transform the orthographic form into a phonetic transcription. For a number of European languages, such a tool exists, making the PSS algorithm useful for several languages.

The relative frequencies $RF(S_i^{hist})$ and $RF(S_i^{mod})$ of a sequence S_i are given by:

$$RF(S_i^{hist}) = \frac{F(S_i^{hist})}{N^{hist}} \quad \text{and} \quad RF(S_i^{mod}) = \frac{F(S_i^{mod})}{N^{mod}},$$

where N^{hist} is the total number of sequences in the historic corpus, and N^{mod} is the total number of sequences in the modern corpus. The relative sequence frequency $RSF(S_i)$ of sequence S_i is then defined as

$$RSF(S_i) = \frac{RF(S_i^{hist})}{RF(S_i^{mod})}.$$

In words, $RSF(S_i)$ is the frequency of S_i in the historic corpus compared to its frequency in the modern corpus. If S_i is relatively more frequent in the historic corpus than in the modern corpus, its RSF value will be greater than 1. Sequences with a high RSF-value are sequences with *typical historic spelling*. The RSF-algorithm tries to find modern spelling variants for these typical historic sequences.

RSF proceeds as follows. The sequence S^{hist} in a historic word W^{hist} is replaced by a wildcard. Vowel sequences are replaced by vowel wildcards, consonant sequences by consonant wildcards. These wildcard words are matched with words from the modern corpus, and the modern sequence S^{mod} matching the wildcard sequence is considered a possible modern spelling variant of S^{hist}. Consider the 17th century Dutch word *volck* (English: people). This is split into the following consonant/vowel sequences: `v o lck`. The sequences *v* and *o* are fairly frequent in modern Dutch, but *lck* is *much* more frequent in 17th century Dutch than in modern Dutch: it is a typical historic sequence. It is replaced by a consonant wildcard *C*, so the wildcard word becomes: `v o C`. The *C* wildcard can be matched with any sequence of consonants. In the modern Dutch corpus, *voC* is matched with *vol* (English: full), *volk* (English: people), and *vork* (English: fork), among others. Thus, the rewrite rules $lck \rightarrow l$, $lck \rightarrow lk$ and $lck \rightarrow rk$ are created and receive score 1. (If one of these rules has already been created by another wildcard word, its score is increased by one.) After all wildcard words containing *lck* have been processed, the rule with the highest score is the most probable.

RNF. A variant of the RSF algorithm is the Relative N-Gram Frequency (RNF) algorithm. Instead of splitting words into sequences of consonants and sequences of vowels, the RNF algorithm splits words into n-grams of a certain length, and tries to find typically historic n-gram sequences. With an n-gram length of 3, the word *volck* is split into the following n-grams: `#vo vol olc lck ck#`, where # denotes a word boundary.

Since the restriction on consonants and vowels is dropped, another restriction on the wildcard is necessary to prevent overly productive matches. If the *lck* sequence is considered typically historic, the wildcard word *voW* (with *W* being the wildcard) can be matched with any modern Dutch word starting with *vo*, including *voorrijkosten* (English: initial driving charge). Clearly the length of the modern sequence replacing *lck* should be similar, we allow a maximal difference in length of 2. The rest of the algorithm is the same as RSF.

3.1 Evaluation

The algorithms PSS, RSF, and RNF construct a large amount of rules; not all of them make sense. Pruning of rewrite rules can be done in several ways. Simply selecting the highest scoring rule for each typical historic sequence is one way. Another way is to test the rules on a small test set containing historic Dutch words from the collection and their modern Dutch counterparts. First, the edit distance $D(W^{hist}, W^{mod})$ between the historic word W^{hist} and its modern form W^{mod} is calculated, similar to [24]. Next, the rewrite rule R_i is applied to W^{hist}, resulting in W^{rewr}. $D(W^{rewr}, W^{mod})$ is the distance between W^{rewr} and W^{mod}. The test score S for rule R_i is:

$$S(R_i) = \sum_{j=0}^{N} D(W_j^{hist}, Wj^{mod}) - D(W_j^{rewr}, W_j^{mod}),$$

where j ranges over all N word pairs in the test set. If $S(R_i)$ is positive, applying the rewrite rule on W^{hist} has decreased the edit distance between the historic words and their modern forms. In other words, the historic spelling is more similar to the modern spelling after rewriting. The test set contains 1600 manually constructed word pairs. For each typical historic sequence S^{hist}, the rule with the highest test score is selected. By setting a threshold, rules that have a negative score can be filtered since they do not bring the historic word and its modern variant closer to each other.

To compare the three rule construction algorithms PSS, RSF and RNF, another test set with 400 new word pairs (historic Dutch words and their modern spelling) was used; the historic words were fed to the various algorithms, and their outputs were compared against the corresponding modern word.

The first column in Table 1 shows the method used (for RNF, the suffixes indicate the n-gram length); the second column shows the number of selected rewrite rules; the third gives the total number of words from the test set that were affected, while the fourth column gives the number of historic words that are rewritten to their correct modern form (edit distance is 0). This is used as an extra measure to compare the rule sets. The last column gives the average edit distance between the rewritten historic words and the modern words, plus the difference with the baseline in parentheses.

Table 1. Results of evaluating the different sets of rewrite rules

Method	number of rules	total rewrites	perfect rewrites	new distance
none	–	–	–	2.38 –
PSS	104	253	101	1.66 (−0.72)
RSF	62	252	140	1.33 (−1.05)
RNF-2	12	271	152	1.29 (−1.09)
RNF-3	127	274	162	1.19 (−1.19)
RNF-4	276	269	166	1.20 (−1.18)
RNF-5	276	153	97	1.79 (−0.59)
RNF-all	691	315	207	0.97 (−1.41)
RNF-all + RSF + PSS	753	337	224	0.86 (−1.52)

The RNF algorithm clearly outperforms the other 2 algorithms with almost all n-gram lengths, except for n-gram length 5. The rule set for $N = 3$ gives the best results. Compared to the baseline (no rewriting at all), this rule set reduces the average edit distance between the historic words and the modern words in the test set by 50%. However, by combining the rule sets of all n-gram lengths, even better results are obtained. This shows that the rule sets have a complementary effect on the test set.

Finally, a combination of all 3 algorithms was used. First creating and applying these rules using one algorithm, then constructing and applying rules using the second algorithm, then the third algorithm was used. These 3 sets of rules were then combined and tested again on the 400 word pair test set (in Table 1, only the best order of application is given). We see that the combination of methods scores best on all of the measures.

Which of our rewrite methods is most effective in bridging the spelling gap between 17th century and modern Dutch? A bigger reduction in edit distance does not always lead to a better rule. The modern Dutch spelling for the historic sequence *cx* should be *ks*. The rule $cx \rightarrow k$ leads to a bigger reduction than the rule $cx \rightarrow cs$, but also leads to a change in pronunciation and often a change in word meaning as well. The number of perfect rewrites provides additional information. A larger reduction in edit distance leads to a larger number of perfect rewrites, leading to more direct matches between historic and modern Dutch. Together, these measures give a fair indication of the effectiveness of the rule sets. For now, this suffices: our aim is to enable the retrieval of historic documents. Does the rewrite method with the biggest reduction in edit distance and/or the largest number of perfect rewrites give rise to the best retrieval performance?—This is the topic of the next section.

4 CLIR Approaches to Historic Document Retrieval

Finally, we turn to retrieval, and investigate the effectiveness of the translation tools developed in the previous section for the retrieval of historic documents. Our main issue is whether the translation resources help the user in retrieving historic documents. For comparison, we take a monolingual approach as our baseline; here, no mapping between the languages takes place. For comparison with earlier research on historic English [18], we also apply the SoundEx algorithm that translates words into codes based on phonetic similarity [19]; based on preliminary experiments we use code length 7.

Also based on preliminary experiments, we found that document translation outperforms query translation. In the case of HDR, translating the historic documents into modern Dutch provides additional advantages over query translation. An advantage for the user is that the "modernized" documents are easier to read. Also, since no stemming algorithm exists for historic Dutch, document translation enables us to use tokenization techniques that have proven to be useful in modern Dutch. This is important because successful cross-language retrieval requires both effective translation and tokenization [22, 10]. In addition

to the performance of the translation tools, we investigate the effectiveness of a stemming algorithm for modern Dutch [23].

Our third set of questions concerns the use of long versus short topic statements. Since the spelling bottleneck may have an especially detrimental impact on retrieval effectiveness for short queries, we conjecture that our translation tools will be more effective for short topics than for long topics.

4.1 Experimental Setting

We used the corpus in historic Dutch, and the topic set in modern Dutch described in Section 2 above. All runs were of the following form: query in modern Dutch, with relevant document in 17th century Dutch. All runs used out-of-the-box Lucene [14] with the default vector space retrieval model and the Snowball stopword list for Dutch [23]. In addition to our monolingual baseline run, we generated runs using the outputs of the various tools described in the previous section, runs with and without the use of stemming, and runs using only the title field of the topic statement as well as runs that use the description field. The measure used for evaluation purposes is mean reciprocal rank (MRR), a natural (and standard) measure for known-item retrieval [5]. To determine whether the observed differences between two retrieval approaches are statistically significant, we used the bootstrap method, a non-parametric inference test [8, 21]. We take 100,000 resamples, and look for significant improvements (one-tailed) at significance levels of 0.95 (*) and 0.99 (**).

4.2 Results

Table 2 shows the results for runs produced without invoking a stemmer. First, restricting our attention to the title queries in the top half of the table, we see that all translation resources (except RSF) improve retrieval effectiveness. SoundEx is surprisingly effective, almost on a par with the combination of all

Table 2. Evaluating translation effectiveness, using the title of the topic statement (top half) or its description field (bottom)

Method	**MRR**	% Change
Baseline (titles)	0.1316	–
Soundex7	$0.2600^\star$	+97.6
PSS	$0.2397^\star$	+82.1
RSF	0.1299	-1.3
RNF-all	$0.2114^\star$	+60.6
RNF-all + RSF + PSS	$0.2780^{\star\star}$	+111.2
Baseline (descriptions)	0.1840	–
Soundex7	0.1890	+2.7
PSS	0.2556	+38.9
RSF	0.1861	+1.1
RNF-all	0.2025	+10.1
RNF-all + RSF + PSS	$0.2842^\star$	+54.5

Table 3. Does the stemming of modern translations further improve retrieval? Using the title of the topic statement (top half) or its description field (bottom)

Method	**MRR**	% Change
Baseline (titles)	0.1316	–
Stemming	0.1539	+16.9
RNF-all + RSF + PSS	0.2780**	+111.2
RNF-all + RSF + PSS + Stemming	0.2766**	+110.2
Baseline (descriptions)	0.1840	–
Stemming	0.1870	+1.6
RNF-all + RSF + PSS	0.2842*	+54.5
RNF-all + RSF + PSS + Stemming	0.3410**	+85.3

translation resources. The results for runs that use the description field of the topic statement, shown in the bottom half of Table 2, are somewhat different. Here, Soundex only makes a minor difference. How can this behavior be explained? Soundex transforms all words into codes of a certain length. Many short words that start with the same letter are transformed into the same code, matching the short (and often irrelevant) words in the description with many other short Dutch words. Soundex adds much more of these short words to the query than the rewrite rules. The titles contain only content words, which are often longer than non-content words, and are matched far less by other, irrelevant words. While still impressive, the relative gain in MRR produced by the combination of all translation resources (on the description field of the topic statement) is only about half the gain on the title topics.

Next, to find out whether there is an added benefit of performing stemming on top of the translated documents, we turn to the results in Table 3. Note that the SoundEx algorithm generates codes rather than human readable text, defying the application of further linguistic tools. On title-only topic statements (see the top half of Table 3) stemming improves effectiveness, but it does not add anything to the combination of the translation resources. In contrast, on the description topics (see the bottom half of Table 3), the grand combination of all translation resources plus stemming leads to further improvements over stemming and over the translation resources.

The previous section showed that the resulting rule set of a combination of the RNF, RSF, and PSS algorithms produced the largest reduction in edit distance and the largest number of perfect rewrites. The question was whether these measures provide a reliable indication of the retrieval effectiveness. The success of the combination is reflected in the retrieval results: all individual algorithms are outperformed by the combined method. It should be noted, however, that the contribution of the RSF algorithm seems minimal. This is likely caused by the relatively small number of rewrite rules it produces: of the 17,794 unique words in the corpus, somewhat more than 4,000 words are rewritten by the RSF rule set, while the PSS rule set rewrites over 8,000 words. The RNF rule set and the combined rule sets rewrite over 11,000 words.

5 Discussion and Conclusions

We proposed a cross-language approach to Historic Document Retrieval, and investigated (1) the automatic construction of translation resources for historic languages, and (2) the retrieval of historic documents using cross-language information retrieval techniques. Our experimental evidence was based on a collection of 17th century Dutch documents and a set of 25 known-item topics in modern Dutch. Our main findings are as follows: First, we are able to automatically construct rules for modernizing a historic language based on comparing (a) phonetic sequence similarity, (b) the relative frequency of consonant and vowel sequences, and (c) the relative frequency of character n-gram sequences, of historic and modern corpora. Second, modern queries are not very effective for retrieving historic documents, but the historic language tools lead to a substantial improvement of retrieval effectiveness. The improvement is above and beyond the improvement due to using a modern stemming algorithm. In fact, modernizing the historic language generally has a beneficial impact on the effectiveness of the stemmer. In sum, our translation resources reduce the spelling gap between 17th century and contemporary Dutch, showing that a cross-language approach to HDR is a viable way of bridging the gap between the historic language of the document and the modern language of a user's query.

Following Braun [3], one can identify two bottlenecks for retrieving documents written in a historic language. The *spelling bottleneck* is caused by differences in spelling between the modern and historic language. The highly inconsistent spelling also resulted in the existence of multiple spelling variations of a word within a single document. A second problem is caused by vocabulary changes. As languages evolve, new words are introduced, while others disappear over time. Yet other words remain part of the language, but their meanings shift. This problem forms the *vocabulary bottleneck*. Our CLIR approach to HDR implies the use of translation resources for retrieval purposes. At present, we make no distinction between different linguistic relations that may hold between translations. The automatically produced rewrite rules exploit the fact that there are common elements in the different orthographic forms of words. Hence, they are an effective method for addressing the spelling bottleneck. The vocabulary bottleneck is a much harder problem. We are currently exploring methods that address the vocabulary bottleneck both directly and indirectly. First, we address it indirectly by using query expansion techniques that specifically expand queries with words not occuring in a modern corpus. Second, we address it directly by mining annotations to historic texts published on the web. This exploits the fact that these words require explanation for modern readers, frequently leading to annotations that explain the historic meaning of a term.

All resources used for the experiments in this paper (the corpus, the topics, and the qrels) are available from `http://ilps.science.uva.nl/Resources/`.

Acknowledgments. Thanks to Margariet Moelands (National Library of the Netherlands) for drawing our attention to historic document retrieval. Thanks

to Loes Braun for making available the *Antwerpse Compilatae* and *Gelders stad-en landrecht* corpus.

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 016.054.616, 017.001.190, 220-80-001, 264-70-050, 365-20-005, 612.000.106, 612.000.207, 612.013.001, 612.066.302, 612.-069.006, 640.001.501, and 640.002.501.

References

1. Brabants recht. Costumen van Antwerpen, 2005. `http://www.kulak.ac.be/facult/rechten/Monballyu/Rechtlagelanden/Brabantsrecht/brabantsrechtindex.htm`.
2. M. Braschler and C. Peters. Cross-language evaluation forum: Objectives, results, achievements. *Information Retrieval*, 7:7–31, 2004.
3. L. Braun. Information retrieval from Dutch historical corpora. Master's thesis, Maastricht University, 2002.
4. CLEF. Cross language evaluation forum, 2005. `http://www.clef-campaign.org/`.
5. N. Craswell and D. Hawking. Overview of the TREC 2004 web track. In *The Thirteenth Text REtrieval Conference (TREC 2004)*. National Institute for Standards and Technology. NIST Special Publication 500-251, 2005.
6. DBNL. Digitale bibliotheek voor de Nederlandse letteren, 2005. `http://www.dbnl.nl`.
7. DigiCULT. Technology challenges for digital culture, 2005. `http://www.digicult.info/`.
8. B. Efron. Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, 7:1–26, 1979.
9. Gelders recht. Gelders Land- en Stadsrecht, 2005. `http://www.kulak.ac.be/facult/rechten/Monballyu/Rechtlagelanden/Geldersrecht/geldersrechtindex.htm`.
10. V. Hollink, J. Kamps, C. Monz, and M. de Rijke. Monolingual document retrieval for European languages. *Information Retrieval*, 7:33–52, 2004.
11. M. Hüning. Geschiedenis van het Nederlands, 1996. `http://www.ned.univie.ac.at/publicaties/taalgeschiedenis/nl/`.
12. K. Kukich. Technique for automatically correcting words in text. *ACM Computing Surveys*, 24:377–439, 1992.
13. M. Lesk. *Understanding Digital Libraries*. The Morgan Kaufmann series in multimedia information and systems. Morgan Kaufmann, second edition, 2005.
14. Lucene. The Lucene search engine, 2005. `http://jakarta.apache.org/lucene/`.
15. NeXTeNS. Text-to-speech for Dutch, 2005. `http://nextens.uvt.nl/`.
16. A.J. O'Rourke, A.M. Robertson, P. Willett, P. Eley, and P. Simons. Word variant identification in old french. *Information Research*, 2, 1996. `http://informationr.net/ir/2-4/paper22.html`.
17. A.M. Robertson and P. Willett. Searching for historical word-forms in a database of 17th-century English text using spelling-correction methods. In *Proceedings ACM SIGIR '92*, pages 256–265, New York, NY, USA, 1992. ACM Press.
18. H.J. Rogers and P. Willett. Searching for historical word forms in text databases using spelling-correction methods. *Journal of Documentation*, 7:333–353, 1991.
19. R.C. Russell. *Specification of Letters*, volume 1,261,167 of *Patent Number*. United States Patent Office, 1918.

20. D. Sankoff and J. Kruskal. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley Publishing Co., Reading, Massachusetts, USA, 1983.
21. J. Savoy. Statistical inference in retrieval effectiveness evaluation. *Information Processing and Management*, 33:495–512, 1997.
22. J. Savoy. Combining multiple strategies for effective monolingual and cross-language retrieval. *Information Retrieval*, 7:121–148, 2004.
23. Snowball. A language for stemming algorithms, 2005. `http://snowball.tartarus.org/`.
24. R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21:168–173, 1974.
25. Wikipedia. Indo-european languages, 2005. `http://en.wikipedia.org/wiki/Indo-European_languages`.