IJDAR (2011) 14:173–187
DOI 10.1007/s10032-010-0133-5

ORIGINAL PAPER

Character confusion versus focus word-based correction of spelling and OCR variants in corpora

Martin W. C. Reynaert

Received: 16 December 2009 / Revised: 6 August 2010 / Accepted: 12 October 2010 / Published online: 3 November 2010

Abstract We present a new approach based on anagram hashing to handle globally the lexical variation in large and noisy text collections. Lexical variation addressed by spelling correction systems is primarily typographical variation. This is typically handled in a local fashion: given one particular text string some system of retrieving near-neighbors is applied, where near-neighbors are other text strings that differ from the particular string by a given number of characters. The difference in characters between the original string and one of its retrieved near-neighbors constitutes a particular character confusion. We present a global way of performing this action: for all possible particular character confusions given a particular edit distance, we sequentially identify all the pairs of text strings in the text collection that display a particular confusion. We work on large digitized corpora, which contain lexical variation due to both the OCR process and typographical or typesetting error and show that all these types of variation can be handled equally well in the framework we present. The character confusion-based prototype of Text-Induced Corpus Clean-up (TICCL) is compared to its focus word-based counterpart and evaluated on 6 years' worth of digitized Dutch Parliamentary documents. The character confusion approach is shown to gain an order of magnitude in speed on its word-based counterpart on large corpora. Insights gained about the useful contribution of global corpus variation statistics are shown to also benefit the more traditional word-based approach to spelling correction. Final tests on a held-out set comprising the 1918 edition of the Dutch daily newspaper 'Het Volk' show that the system is not sensitive to domain variation.

M. W. C. Reynaert (✉)
Tilburg Centre for Cognition and Communication,
Tilburg University, Kamer D 342, P.O. Box 90153,
5000 LE, Tilburg, Netherlands
e-mail: reynaert@uvt.nl

1 Introduction

We present an approach to spelling variation identification and correction on the scale of very large corpora.

This work is situated in the context of the large digitization programmes underway at the Koninklijke Bibliotheek (KB), the Dutch National Library. The results of our work should be applicable to the etexts produced by a broad range of large digitization efforts comprising at least those involving languages written in alphabetic scripts.

The main objective of this work can be illustrated by reference to the difference one would find in the perceived correct version, i.e. the gold standard, produced for e.g. a printed book for the purposes of evaluating the electronic version obtained by means of Optical Character Recognition (OCR) versus the gold standard produced for evaluating a post-OCR correction system. The OCR-gold standard should faithfully represent the text as printed, i.e. would have end-of-the-line hyphenated and split words at the end of one and at the beginning of the next line. It would also mirror any typographical or typesetting errors that happen to occur. The OCR post-correction gold standard would have neither: split words would have been restored and unnecessary hyphens removed, typographical or typesetting errors would have been duly corrected. Seen from this point of view, however perfect the OCR-process may eventually become, post-correction will have real world use.

In this paper, we investigate a new approach to the identification and correction of lexical variation implemented in the Text-Induced Corpus Clean-up tool TICCL. In principle, TICCL looks for any kind of lexical variation, whatever was its cause, be it historical spelling changes, typographical or typesetting errors, OCR misrecognition, transmission noise or morphological variation.

Approximate matches are strings that are similar but not identical to the string one looks for. An in-depth overview of the state of the art in approximate string matching can be found in [1]. While many algorithms for finding approximate matches between word strings have been developed, so far no algorithm seems to have been put forward that, per character confusion, identifies all those word pairs that differ in exactly the same particular set or bag of characters, regardless of the actual character sequences. This paper proposes this approach. The approach takes a different tack from the usual local focus on one particular text string. We present a simple, efficient technique that allows for identification of all the pairs of strings in a corpus that differ in the same particular set of characters. We call this a global, character confusion-based operation, in contrast to the local, word-based variant retrieval procedures employed by most spelling correction systems.

In Sect. 2, we first discuss related work, with the focus on the two mainstream approaches to spelling correction today. We conclude the section by discussing what we see as the main drawbacks of both approaches in light of work on large-scale digitized corpora. In Sect. 3, we outline anagram hashing and both the focus word-based and character confusion-based approaches to spelling checking. In Sect. 4, we then proceed to explain which information sources and filtering techniques the implementation of our correction systems was equipped with in order to be able to achieve high and well-balanced recall and precision. In Sect. 5, we sketch the context in which we develop our systems, describe the corpora we work on and provide some statistics about the errors they contain. We evaluate extensively on a gold standard for a corpus of 327,798 pages of OCR-ed Dutch parliamentary text and on a held-out set comprising one year's edition of a Dutch newspaper. We conclude in Sect. 6.

2 Related work

A very comprehensive, but now dated, survey of the field of spelling error detection and correction prior to 1992 is provided by [2]. Since then, tremendous amounts of work have been done, not only for English, but for many other languages. The range and scope of the work has broadened enormously. An admirably concise yet very informative overview of the overall evolution in the field of lexical error correction is provided in [3]. Whereas spelling correction initially relied exclusively on a dictionary which delineated what is a real word and what is not, i.e. is a non-word, due to changes in the circumstances in which spelling correction has found application, reliance on the dictionary has lessened. Their last and most comprehensive reformulation of the spelling correction problem is the first to no longer make explicit use of a validated lexicon of the language. In the work we report on here, this is also the case, as we shall show later.

Work on the post-correction of OCR-ed documents is by no means new. We note the study by [4] in which output from various OCR engines is set to vote on the most likely character sequence.

Our work is situated mainly in non-word spelling correction, to which we see two mainstream approaches, although both have also been applied to context sensitive spelling correction, i.e. the identification and correction of confused real words. On the one hand, there is the noisy channel-based approach going back to [5], in which the work of Cucerzan is rooted. On the other hand, there are large bodies of work based on Finite State technology, which gained considerable influence at least in part due to the work of [6] on agglutinative languages.

A very recent concise overview of the work in the source channel paradigm up to date is provided in [7]. The main application of this approach in recent years is in the context of the spelling checking of search queries. This is a difficult task, as is evident from the rather low scores reported, i.e. 0.573 recall, 0.781 precision. We note that apart from accuracy, recall and precision are now being provided which implies that the importance of precision is gradually being recognized. The task of query spelling correction is very different from corpus clean-up because the online medium provides the training material necessary for applying the source channel paradigm. Fruitful use can be made of the recorded actions of the users who correct their own spelling errors in their own queries. There is unfortunately no equivalent for this in corpus clean-up. If the source channel approach is to be applied to corpus clean-up, costly hand-crafted training material will have to be created or error corpora will have to be fabricated. One possible solution might lie in digitization programmes where manual correction is performed. This would require the alertness of the programmes' managers to the need of preserving the uncorrected OCR-ed text and of providing these with the corrected versions for training and testing purposes to the document analysis, recognition and post-correction community. Benchmark sets for almost all languages are sorely lacking and this perhaps helps explain why the source channel approach is not today commonly deployed in corpus clean-up. Early work on OCR post-correction in the noisy channel paradigm is by [8]. This escaped the notice of [9] who mistakingly claimed a first and further have had to report that the system developed by [10] outperforms theirs on their own test data. Both papers only report accuracy on error lists. The lack of re-usable benchmarks, consensus on the metrics and best practices for evaluation and the fact that many researchers de facto work on different languages make fruitful comparison of achievements and progress a goal to be reached in the future.

Likely in the absence of sufficiently large corpora and lack of training materials, statistical approaches developed on the morphologically less rich English language were found wanting and researchers such as Oflazer working on the highly agglutinative Turkish language turned to Finite State techniques. These techniques also allow for modeling subtle exceptions in a language which are less likely to be discerned by statistical approaches. This has found wide acceptance and has been successfully applied to corpus clean-up. Working in this mainstream Finite State Automata (FSA) paradigm, mainly on German corpora, the group around professor Schultz at the University of Munich work on large-scale corpus clean-up. In [11], the focus is on post-correction of OCR-ed corpora, while in [12], on the cleaning of web-derived corpora. The paper describes in detail how the typical error types that are observed in collections of typographical, spelling and OCR-errors are modeled. Further how and why the ranges of errors are limited in order to build error dictionaries in which these patterns are applied to validated dictionary words and the fabricated erroneous word forms incorporated in an FSA.

We shall see that the approach presented here does not require prior error modeling, so no prior assumptions about which error types may or may not be encountered are made. Neither need we fabricate and represent the possible erroneous word forms, as is necessary in the FSA paradigm. Within the limits of a particular Levenshtein distance (LD) [13] we set it to search in, our approach exhaustively surveys the full corpus and retrieves all the variation present in it. Perhaps most importantly, the vocabulary of the corpus under consideration is dynamically incorporated and use is made of the observed token frequencies to help guide the correction process. This alleviates the well-known domain and genre effects that seem to be currently primarily attracting this group's attention, cf. [14]. We are not sure whether incorporating the corpus vocabulary at correction time is possible in the FSA paradigm.

The two mainstream approaches to spelling correction, the noisy channel and FSA approaches, both have drawbacks. Noisy channel approaches, although there have been attempts to reduce the effects of this dependency, need costly training material. FSA approaches do not necessarily need this, but underlying the FSAs that are built need necessarily be particular assumptions concerning the data to be handled, mainly to reduce the cost of the FSA itself, i.e. for reasons of scalability. These assumptions necessarily limit the capabilities of any FSA which is built, e.g. if it is assumed that the first character of a word is hardly ever substituted for another character, the system will not be able to handle this phenomenon if it nevertheless does occur.

The system we describe in this paper does not require training material, it derives the statistics about e.g. character confusions that occur from the corpus to be cleaned itself. This entails that no prior assumptions regarding the prevalence of particular phenomena within the material to be cleaned need to be made. In fact, within the limits of the LD the system is set to work, it will exhaustively gather all the variation present and employ the statistics gathered about them to the full.

One major obstacle in this work is the huge diversity in corpora. Corpora may have been born digital or be the result of digitization efforts. They can be in any language. There is a huge diversity in text types and domains. Increasingly, diachronic spelling is revived by the digitization of older text collections. There are tremendous amounts of work to be done. We would argue that a system which does not require prior modeling or specific training data is worth investigating. In the next Section, we describe such a system.

3 Anagram key spelling correction

In this section, we first describe the basic algorithm underlying both the focus word-based and character confusion-based approaches to large-scale spelling correction.

3.1 Anagram key search

3.1.1 Introduction to anagram hashing

Anagram hashing is the core spelling variation identification algorithm we first described in [15] and in more depth in [16]. Anagram hashing uses a simple hashing function to assign a large natural number to all word strings in the corpus at hand. The natural number assigned is the same for all word strings that consist of the same set of characters. For each word type in a TICCL lexicon or in a corpus to be processed by it, anagram hashing obtains a numerical value, which will serve as an index or hash key to the actual word strings. The formula represents the mathematical function devised to do this, where f is a particular numerical value assigned to each character in the alphabet and c_1 to $c_{|w|}$ the actual characters in the input string w.

$$Key(w) = \sum_{i=1}^{|w|} f(c_i)^n$$

Informally: the numerical value for a word string is obtained by summing the code value, e.g. ISO Latin-1, of each character in the string raised to a power n, where n was empirically set at: 5.

By application of this formula to the list of word strings obtained from a corpus, in effect, all anagrams, loosely defined as words consisting of a particular set of characters and present in the list, will be identified through their common numerical value. In the limit, associated with a particular key would be the $n!$ permutations given n distinct characters, if these were realized within the corpus. Given the set of characters **a, b, c**, there could be 3 x 2 x 1 = 6 permutations, but only two, i.e. 'abc' and 'cab', are likely to be encountered in e.g. an English dictionary. As the collisions produced by this function identify anagrams, we refer to this as an **anagram hash** and to the numerical values obtained as the **anagram values** (AVs) and **anagram keys**, when we discuss these in relation to the hash. Based on a word form's anagram key, it thus becomes possible to systematically and sequentially query the list for any variants of a particular word string present, be they morphological, historical, typographical, orthographical or due to OCR-misrecognition or other transmission noise. This querying is done on the basis of the AVs which represent the alphabet which is used.

TICCL in fact performs a bounded exhaustive search over the possible permutations given a particular set of characters that happen to have been realized in the particular corpus it is set to work on. This is less expensive than it may seem in that in practice within a language only a limited number of all the possible word forms given a particular set of characters are realized as valid words.

3.1.2 Alphabet

Instead of matching on actual characters, we perform simple mathematical operations with the AVs derivable from the alphabet *A* used. The actual amount of AVs is defined by the LD or edit distance the system is allowed to cover and the size *a* of *A*. The LD then defines the *k* character differences or possibly 'errors' that the system will search for. In this work, $k = 2$. Given the characters in *A* in combination with $k = 2$, the AV-alphabet contains the AVs for single characters and for all possible two-character combinations derivable from this alphabet.

3.1.3 Lexicon

The lexicon contains the vocabulary of the validated dictionary (if any) as well as the vocabulary from the corpus to be cleaned. The lexicon is a regular hash built up at run-time having the AVs as keys and chained anagrams as values.

3.2 The sequential focus word-based approach

In the focus word-based approach, each word string is examined in a sequential fashion, making the word string under consideration temporarily the 'focus word' (FW). Most approximate matching and therefore spelling correction systems work in this fashion.

The FW is not likely to contain all the characters in *A*. The subset of values from the AV-alphabet derivable from the characters actually present in FW forms the FW-alphabet. If $k = 2$, the FW-alphabet has all the AVs for the character unigram and all possible bigrams derivable from the FW. Given the AV for a particular FW and by systematically querying the lexicon hash on the basis of all the values in the AV-alphabet and in the FW-alphabet, all possible variants that fall within *k* are retrieved. The actual number of hash look-ups required is defined by the number of unique values in the AV-alphabet and by the number of unique values for all the character combinations in the FW up to *k*.

The AV for the focus word and the FW-alphabet and AV-alphabet are used to query the lexicon hash for variants of the focus word. These variants can all be seen as variations and combinations of the usual error type taxonomy due to [17]. In the implementation, all four edit operations are handled as substitutions. For **substitutions**, a value from the FW-alphabet is subtracted and a value from the AV-alphabet added. A single query on the AV for 'yesterday' minus the AV for an 's', plus the AV for an 'a' may thus retrieve the typo: *yeaterday. **Insertions** are substitutions where a value from the FW-alphabet is subtracted and zero added. **Deletions** are substitutions where zero is subtracted and a value from the AV-alphabet added. To find **transposition** errors, nothing needs to be added or subtracted, but the chained anagrams for the particular focus word AV need to be examined. If only a single word string is associated with the FWAV, no transpositions can be present. If anagrams are associated with the FWAV, then a pair of these may be found to contain transpositions. The transposed characters need not be adjacent. To give an example: given that a corpus contains the words 'steenmolen' (E: stone mill) and 'molensteen' (E: mill stone), but also the non-word: 'melonsteen'. The three anagrams will be chained to the same AV. The first two have an LD of 5 and are ruled out as containing a transposition error. The last two have an LD of two and can thereby be identified as containing a transposition error.

3.2.1 Pseudo-code for the sequential word-based approach

```
Set LDlimit to $k$
Set WordLengthLimit to $l$
Foreach FocuswordAnagramValue
  Get WordLength of FocusWord
  If WordLength > WordLengthLimit
    Foreach AlphabetValue in Alphabet
      Foreach FocuswordAlphabetValue
      in FocuswordAlphabet
        NewValue = FocuswordAnagramValue
        − FocuswordAlphabetValue + AlphabetValue

        If NewValue defined in LexiconHash
        VARIANTS = list of anagrams (string variants)
        associated with NewValue
          For each variant in VARIANTS
          Calculate LD between focus and variant
            If LD <= LDlimit
```

```
                        Return variant
                        Endif
                      Endfor
                    Endif
              Endif
            Endfor
```

3.3 The sequential character confusion-based approach

Given a particular k and a particular alphabet A, we first form all the possible combinations of all the characters in A and calculate their AVs. We then perform all the additions, deletions and substitutions on the basis of all their AVs. Given that $k = 2$ and alphabet size a is 31, this gives 123,752 unique values. These values represent all the possible minimal confusions given A and k. Minimal confusions are the result of the fact that equal characters on both sides of the equation cancel each other out. Calculating the AV for the character combinations 'ab' minus 'a' or for 'bc' minus 'c', we obtain the same minimal confusion: the AV for 'b'.

Given this list of character confusion AVs, one can now query the list of lexicon/corpus anagram keys for all the word/anagram pairs which display these particular numerical differences. This we do sequentially for all the character confusion AVs by efficient iteration over both numerical lists.

3.3.1 Pseudo-code for the sequential character confusion-based approach

```
Foreach CharacterConfusionValue in CharacterConfusion
  List
  Foreach AnagramValueKey in CorpusAnagramHash
  NewValue = AnagramValueKey + CharacterConfusion
  Value
    If NewValue defined in CorpusAnagramHash
            Return and store AnagramValueKey
            as a chained hash value for key
            CharacterConfusionValue
    Endif
  Endfor
Endfor
```

3.4 Necessity of output filtering

The word pairs identified per character confusion and retrieved from the anagram hash constitute the **members** of the confusion set. Be advised that a particular confusion in fact describes a minimal confusion. The actual surface forms of a member pair may be very divergent: all we know a priori about them is that they differ by the set of characters implied by the confusion's anagram value and that one of the pair will show the extra (or in anagram value terms: numerically greater) character(s). The actual sequence of the characters in the pair may be very different. This is where the LD comes in: only for the pairs retrieved need we measure their LD. If this measured LD exceeds the LD implied by the confusion's anagram value, the pair has been spuriously linked and should be discarded.

4 Text-Induced corpus clean-up or TICCL

The TICCL prototype presented in [18] was turned into a production version for the KB according to their specifications. The production version at first performed only FW variant lookup. The move to the CC variant look-up in the current form is new. Also, new is that in the FW mode retrieved variant pairs are no longer evaluated in isolation, but as members of the CC sets they belong to, i.e. on the basis of the sizes of these sets: the statistics obtained over the whole corpus about the variation within k actually present.

Faced with the tremendous rates at which very large collections of digitized text grow and are becoming available online, we have been searching for alternative, simpler solutions to the lexical variation problem. In [19], we have presented our first attempt at approaching the problem not from a word type-centered perspective, but from the character confusion perspective. We there described a solution for finding all the word pairs in a corpus that exhibit the same character confusion in a single parallel operation. The solution proposed was a Boolean AND operation on very large and sparse bit vectors built from the anagram hash for a particular corpus. This was later found to be unnecessarily costly, the vectors being very sparse, and the idea was shelved. We here propose a faster and in fact simpler solution.

TICCL is implemented in Perl. For the character confusion iterations, we have had a fast C++ module built which efficiently performs all the necessary iterations over the anagram keys for the corpus-derived anagram hash on the basis of numerically ordered lists.

4.1 TICCL: Step 1: normalization for search space reduction

TICCL first effects a thorough normalization of the corpus it works on. Primary aim of this normalization is to drastically reduce the search space TICCL has to work in by a reduction in the actual number of different characters present in the corpus. In this paper, we reduced this number to the size a of A, where $a = 31$. In our experience, the OCR process may very well produce all the characters in the full code page, even non-printing characters. This search space needs to be reduced to manageable proportions.

The alphabet A we work with here is the range of the lowercased characters 'a' to 'z'. TICCL basically ignores numbers, internally these are all converted into '3'. This frees up the other digits. We translate all punctuation marks except for the apostrophe and the hyphen into the digit ‘2’. The word-internal punctuation marks apostrophe and hyphen retain their own identity. All characters bearing diacritics are translated into the digit ‘4’ when lowercase and ‘5’ when uppercase.

In Step 1 of processing, TICCL traverses the range of directories it was directed to and normalizes the word forms it encounters in the files to be processed, building frequency lists and, optionally, bigram word list on the way. Full word form normalization is postponed until all the word forms have been seen. Because we do not work on preprocessed input, i.e. no prior sentence splitting or tokenization is required or assumed, we iterate over the frequency list containing the corpus vocabulary and conflate all the word forms that contain normalized punctuation with the matching higher frequency word forms that do not contain normalized punctuation. At this point, the punctuated form’s frequency is tallied to the unpunctuated word form’s frequency and reset to zero. The observed surface forms, i.e. the non-normalized conflated word forms’ character strings as observed in the corpus, are stored in a separate hash. These forms are thus withheld from further cluttering the search space during variation look-up. At the end of total processing, they are linked to the canonical form reported for the normalized, unpunctuated word form. In this same manner, words having diacritics or names requiring capitalization are output in their canonical form.

4.2 TICCL: Step 2: Character confusion word pair identification

This second step is the only one in which the all-Perl FW mode and the combined Perl/C++ CC mode implementations differ.

The FW-based version in essence processes part of the corpus vocabulary word list by descending order of the word forms’ frequencies. A number of parameters may be set to restrict this process, i.e. frequency thresholds may be imposed or word length restrictions invoked. The settings for the tests we report on later were: process everything between the highest frequency observed and above frequency 1 having word length longer than 5 and shorter than 100 characters. The motivation for this limit on word length is this: for some words, there simply are no other words resembling them to the extent that these would fall within the LD of 1, 2 and even more edits. As such, words with a higher neighborhood density [20], especially short words and words derived from a stem and highly common pre- and/or affixes, are far more likely to incur more False Positives. Short words not even being indexed by search engines we therefore disregard.

The sequential CC-based version works on the lexicon hash obtained from the normalized corpus. For efficiency, this may already have been reduced in size, i.e. when building this hash one may want to opt to not include word forms shorter than a particular threshold. This we did not do in the tests described further. The C++ routine works strictly numerically on the anagram keys of the lexicon hash and the character confusion AVs. The module’s output is lists of corpus anagram key values each linked to a particular character confusion AV.

Working in CC mode has introduced the need to rethink handling the variation retrieved by the system. The FW mode returns a set of actual variants displaying a range of character confusions per particular focus word. This set has already undergone some filtering on the basis of the actual word pairs retrieved from the lexicon hash. Words shorter than the length limit set have not been retained, pairs displaying a larger LD than *k* have also not been returned. Word pairs where both members do not actually occur in the corpus, i.e. that have an undefined corpus frequency, are also discarded. The CC mode returns a possibly very large set of references to word type pairs, all displaying the same character confusion, somewhere in the bags of characters that these word types represent. For a first filtering, we now need to retrieve the character confusion word pairs from the lexicon hash and apply the LD and word length filtering. We also perform a single pass over the lexicon hash and retrieve the word pairs displaying character transpositions in the way described in Subsect. 3.2.

When this filtering has been performed, we now have sets of likely word pairs. We can now also determine the size of these sets, thereby gathering useful statistics about the actual variation present in the corpus.

4.3 TICCL: Step 3: filtering the output

In Step 3 in processing the corpus, all the character confusions are handled sequentially in descending order of the size of their character confusion set membership. This is equivalent to assigning a higher probability to one particular confusion than to another.

We next perform additional filtering of the word pairs. For this, additional sources of information are used.

4.3.1 Additional language information sources

- Validated lexicon: The validated lexicon consists of the concatenation of the official word lists for Dutch published in 1995 and 2005[1] and their 1914 predecessor[2] for all the tests performed in this paper. These are further complemented with the freely available ‘Open Taal’[3] and

[1] Freely available for research from the Dutch-Flemish HLT Agency. http://www.inl.nl/nl/corpora/.

[2] http://www.gutenberg.org/files/22722/.

[3] http://www.opentaal.org/.

Ispell[4] dictionaries. Also incorporated were the list of all the names of members of both Dutch Houses of Parliament[5] since their beginning in the early nineteenth century as well as a comprehensive list of Dutch place names.[6]

– Validated lexicon confusables: Within *k*, we build a full mapping of all the confusions between words in the validated lexicon for the language. This then constitutes a full mapping of all the confusables—here defined as any words in the validated lexicon within a particular edit distance from any other words in the validated lexicon – in the language. We build this matrix from the anagram hash key list derived from the validated lexicon. This process is equivalent to Step 2: the C++ module is run over the anagram key values using the same list of CC AVs as in the corpus variation retrieval step.
– Ispell morphological rules: To identify morphologically related words we re-use the morphological information for Dutch available in the open source spelling correction system Ispell. We derive the rules for the language provided in the Ispell affix file in conjunction with the Ispell word list. We extrapolate from this necessarily limited Ispell word list to the full vocabulary as derived from the corpus. We convert the Ispell rules into anagram values and apply them to the corpus vocabulary. For each character confusion AV handled when retrieving the confusion pairs from the corpus anagram hash, when the character confusion anagram value corresponds to (often a set of) Ispell affix rules, we check for each pair whether perhaps one of the rules applies. If this is the case, we conclude that by analogy to what we learned from Ispell, the two word forms are morphologically related and filter out the word pair. There is some danger of overgeneralization here because we do not know the words' grammatical classes.

4.3.2 Contribution of additional language information sources

In applying the information sources described above to the corpus vocabulary, we are able to:

– correctly link morphologically related word forms
– to avoid linking the semantically unrelated but typographically close word forms, i.e. the confusables
– and finally to retain primarily those word pairs that are the result of some kind of error and to link those to their (more) canonical, contemporary counterparts.

[4] Available, with affix files, for at least fifty languages from: http://www.lasr.cs.ucla.edu/geoff/ispell-dictionaries.html.

[5] http://www.parlement.com/.

[6] http://nl.wikipedia.org/wiki/Lijst_van_Nederlandse_plaatsen.

4.3.3 Filtering loops implemented in TICCL

We next examine all the remaining word pairs per character confusion AV with the latter in descending order of membership sizes.

The character AVs either correspond to the set of AVs derived from the minimal confusions of the Ispell rules or not. If corresponding, for each word pair retrieved, we check if any of the expanded Ispell rule patterns apply. If a pattern applies, the pair is identified as being morphologically related word forms. If no pattern applies, the pair is linked as being error variants. This linking is done on the basis of the observed corpus frequency of both word forms, the lesser frequent form being linked to the higher one. The assumption in this is that the more canonical word form is likely to be more frequent than an erroneous variant, given Zipf's law [21].

For those character confusion AVs that do not correspond to an AV derived from the minimal confusions of the Ispell rules, we first check for each word pair retrieved whether it occurs in the validated lexicon's matrix of confusables. If it does not, the words are linked according to their frequencies in the same manner as above.

We perform no further filtering in the current TICCL, although further refinements are due to be added. We here report the evaluation results obtained with this simple filtering only.

4.3.4 Ranking

The word pairs finally retained are output after they have undergone sorting, i.e. a final ranking of the correction candidates. We sort them according to the observed order of having been filtered according to the sizes of their CC membership set, then according to the LD between canonical form and lexical variant, finally according to the frequency of the canonical form. We have tried different sorting sequences, but obtained the best results with this one. All results reported in the next Section were obtained by sorting according to this sequence, except those where no ranking was performed.

At time of output, we can invoke a stopping criterium: with best-first ranking only a single pair is output, with 2-best or rank 2, the two first ordered pairs, etc.

5 Evaluating TICCL

5.1 Preliminaries to the evaluation of TICCL

This work was undertaken with a specific user in mind, i.e. the KB. At least to some extent, it was undertaken in accordance with the user's requirements and according to his specifications. The user undertakes huge digitization programmes in order to make available online large parts of his immense text collections. These programmes are not undertaken in uniform ways, although all adhere to specific international standards. As such, the user requires flexibility and independence from our tool and our solutions. We were not asked to deliver a tool that would correct the KB's digitized texts. Rather, we were asked to deliver a tool that would allow them to enrich the digitized texts with canonical forms for the word variants present in these texts. The canonical forms added to the texts should allow for better recall on users' queries through a search engine built on top of the particular text collection.

The above considerations help to define what the target of our tool is and thereby how we should evaluate it. The target should be those word forms that enhance the recall of a user's query. The KB would like its users to be able to query historical documents in the contemporary spelling, given that the user may well not be familiar with historical spelling and/or the fact that historical spelling varied a great deal throughout time and space. This we do not deal with nor measure in the current paper. What we do measure here is how well our two approaches manage to deal with lexical variation due to typographical/typesetting or OCR misrecognition error.

5.2 The corpora

On invitation by the KB we have worked on contemporary and historical text collections. We developed on the contemporary collection, which comprises the published Acts of Parliament (1989–1995) of The Netherlands, referred to as 'Staten-Generaal Digitaal' (henceforth: SGD).[7] From the historical newspaper collection, the 'Database Digital Daily Newspapers'[8] (DDD) we have chosen the 1918 edition of 'Het Volk' (E: The People). We list statistics on the corpora in Table 1.

We list statistics obtained from the OCR-ed corpora we here work with: error statistics on 5,047 mainly OCR-errors from the SGD in Table 2 and 3,799 from the DDD in Table 3. For the SGD, we focused on the word 'belasting' (E: tax), a common topic in parliamentary debate, and strove to identify all variants in all morphological and compound forms of the word. In all, we identified 1,577 variants for the various guises of the noun 'belasting'. For the DDD, we opted to identify all the variants for the lemma 'regeering', i.e. 'government'. This lemma yielded 1,468 variants in a single newspaper in the DDD, the 1918 edition of 'Het Volk' alone. A multiple error cannot be described by reference to just one of the 4 categories of errors, i.e. either to insertion, deletion, transposition or substitution alone. A multiple contiguous error (multi-C) would be the OCR-error 'regeermg' for 'regeering', i.e. the multiple error consisting of deletion of an 'i' and substitution of the 'n' by 'm' is situated in one location within the word. A multiple noncontiguous error (multi-NC) would be the OCR-error 'rcgecring' for 'regeering'.

5.2.1 Gold standards

We measure the extent to which TICCL manages to achieve acceptable recall and precision on our gold standards. These gold standards, while quite large, are necessarily limited and may well not be adequate to show the systems' full, real performance. The system might actually fail regarding phenomena that occur in the corpus but for which there happen to be no instances in the gold standard.

We evaluate on a subset of the paired lists of variants and focus word for which we presented error distribution statistics. The subsets involved all the variants for the 20 SGD focus words in Table 4, 890 in all, and all the variants for the 17 focus words for 'Het Volk', 3,102 in all. Listed next to the focus words are the numbers of variants found. Sampling the typographical variation present within the corpora involves exhaustively gathering all the typographical variants for the focus words. There is some overlap in the common words between the SGD and 'Het Volk'. Names, especially names of historical figures, being more tied to their era, provide less opportunity for such overlap.

We built a gold standard for the years 1989–1995 (SGD8995). We here re-use this for evaluation on the development set and for running ablation tests with the aim of demonstrating the contribution of the various language information sources to full system performance. We eventually also test on the held-out gold standard derived from 'Het Volk' 1918-articles (HETVOLK1918). For the purposes of the present work, we ignore the fact that Dutch spelling was officially changed, repeatedly, since then and test the system on its ability to link word variants to their canonical form at the time. So we do not require that e.g. the 1,468 variants for the current spelling 'regering' (E: government) are linked to this current canonical spelling, but measure whether they are linked to the then canonical spelling 'regeering'.

5.2.2 Test settings

The tests were limited to unigram correction only, we here did not perform variant retrieval for word bigrams. The gold standard did not contain any instances involving spaces. k is 2. We use real world dictionaries, which do not necessarily contain all the correct word forms for all the variants present. We evaluate on the n-first or 'best' ranked correction candidates returned by the systems, and we measure how well the system is able to detect errors and to suggest the appropriate correction candidate within the n candidates it maximally reports without making use of any local context. This is measured in terms of the impact on word types, but for the held-out test set, also in terms of the impact on the

[7] http://www.statengeneraaldigitaal.nl/.

[8] http://kranten.kb.nl/.

Table 1 Corpora Statistics: Corpus, language (CD: Contemporary Dutch, HD: Historical Dutch), number of OCR-ed text pages, number of word tokens, number of word types

Corpus	Lang.	Pages	Tokens	Types
SGD	CD	327,798	125,209,007	1,156,998
DDD	HD	8,664	7,950,950	1,535,529

Table 2 SGD 1989–1995: overview and statistics per LD of error types in a sample of 5,047 non-word variants

Category	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	LD 7	Total	(%)
Deletion	221	10	3	1				235	4.66
Insertion	1,980	27	6	11				2,024	40.10
Substitution	1,065	49	37	3		1		1,155	22.89
Transposition		26						26	0.52
Multi-C		722	30	10	1	1		779	15.46
Multi-NC		303	271	101	22	5	2	710	14.09
Total	3,380	1,138	347	126	23	7	2	5,047	
(%)	66.98	22.55	6.88	2.50	0.46	0.14	0.04		100.00

Table 3 DDD ‘Het Volk’ 1918: overview and statistics per LD of error types in a sample of 3,799 non-word variants

Category	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	Total	(%)
Deletion	31	27	1	12			71	1.87
Insertion	133	25	3	4			165	4.34
Substitution	575	276	109	2			962	25.32
Transposition		3					3	0.08
Multi-C		203	193	9	2	1	412	10.85
Multi-NC		810	1,277	77	15	3	2,182	57.44
Total	743	1,344	1,583	104	17	4	3,799	
(%)	19.56	35.38	41.67	2.74	0.45	0.11		100.0

word tokens, i.e. in terms of how often a word type appears in the corpus and may thus influence the overall quality of the text.

5.2.3 Metrics used

We evaluate in terms of recall and precision, resulting in the combined F-score [22]. These metrics are derived from the numbers of True Positives (TPs), False Positives (FPs) and False Negatives (FNs) returned by the system. **True Positives** are defined by what constitutes the **target** of our exercise. The target is the primarily non-word variants present in the corpus-derived list to be processed. **False Positives** are non-word variants or real words, that are erroneously reported to be variants for a particular focus word. **False Negatives** are those items in the list of known, annotated variants for the particular focus word that are absent from the list of variants returned for this focus word, i.e. that the system was not able to retrieve or ‘correct’. The formulae used are as follows:

$$\text{Recall} = \mathrm{R} = \frac{TP}{TP + FN} \qquad \text{Precision} = \mathrm{P} = \frac{TP}{TP + FP}$$

Since we deem recall and precision to be equally important, the harmonic mean of R and P, the simplified F measure, F, is given by:

$$\text{F-score} = \mathrm{F} = \frac{2 \times R \times P}{R + P}$$

5.3 Evaluation of TICCL

All performance scores reported here are accumulated LD 2 scores. We did not set k higher, so we do not measure higher LD scores. Scores are on n-best first ranking, in contrast to previously reported results on these corpora. Scores reported in [18] were ‘overall’: however many correction candidates were retrieved, given that the correct one was among them, the system was there given credit for it. Good first or second best and well-balanced scores are a prerequisite for a

Table 4 Overview of the SGD8995 and HETVOLK1918 focus words and their observed numbers of variants which constitute the evaluation sets

Focus SGD8995	#	Focus 'Het Volk 1918'	#
Achttienribbe-Buijs	23	Amsterdam	307
Amsterdam	43	Annexionisten	20
Bolkestein	18	België (Belgium)	104
Jorritsma-Lebbink	33	Bismarck	10
Nieuwenhoven	22	Compiègne	3
Rotterdam	47	Hindenburg	32
Wolffensperger	25	Nederlandsche (Dutch)	572
belasting (tax)	36	Posthuma	264
belastingen (taxes)	56	Richthofen	7
belastingplichtige (taxable person)	41	Trotzky	45
belastingplichtigen (taxable persons)	37	Wilhelmina	42
doelstelling (aim)	82	Zeeuwsch-Vlaanderen	19
doelstellingen (aims)	58	belasting (tax)	102
evaluatie (evaluation)	44	belastingen (taxes)	34
faciliteiten (facilities)	27	distribueeren (to distribute)	52
goedkeuring (approval)	36	eenheidsworst (unity sausage)	21
inkomstenbelasting (income tax)	81	regeering (government)	1468
motorrijtuigenbelasting (motor vehicle tax)	70		
studiefinanciering (study financing)	93		
vennootschapsbelasting (corporate tax)	52		

Capitalized words are proper names

Table 5 SGD 1989–1995: Performance results on word types for the full system run in both FW and CC modes

Mode rank	Focus word				Character confusion			
	R	P	F	Minutes	R	P	F	Minutes
1	0.923	0.955	0.939	1347.8	0.923	0.956	0.939	122.0 + 72
2	0.963	0.948	0.955	1344.0	0.963	0.949	0.956	119.6 + 72
3	0.963	0.946	0.954	1354.7	0.963	0.947	0.955	125.0 + 72
5	0.963	0.946	0.954	1304.8	0.963	0.947	0.955	120.9 + 72
10	0.963	0.946	0.954	1293.1	0.963	0.947	0.955	119.1 + 72
Averages	Perl processing time			1328.9	Perl and C++ processing time			121.3 + 72

N-best ranking scores are on ranks 1, 2, 3, 5 and 10-best. Note that the FW approach requires an order of magnitude more time to achieve similar results as the CC approach

spelling and OCR correction system to be set to work fully automatically on a corpus.

5.3.1 Comparison of FW *and* CC *modes on* SGD8995

We have now evaluated and timed both approaches. Table 5 shows performance of TICCL in scores and time in minutes required when run in FW versus CC modes. The only difference in the implementations and running of the two approaches lies in the identification of the variants present. For steps 1, 3 and 4 of the whole process, the Perl code is shared. The C++ CC-based code replaces that for step 3 of the FW approach.

The differences in performance scores are negligible. We have hereby shown that a sequential word-based system may also collect the global information about the various character confusions seen within a corpus and apply this knowledge in the same way as the CC approach does, i.e. by handling all the sets of character confusions in the order of decreasing size of their membership sets.

Processing times differ by an order of magnitude: average run times over the five runs here are over 22 h for the FW mode and just over 3 h for the CC mode. While ostensibly feasible, building all possible character bigrams for every word type to be examined by the FW approach is costly. It is this cost which is translated into the far longer processing times required. This provides conclusive evidence that the

CC approach allows faster identification of spelling variation with less work, given a large corpus. Given a relatively small corpus e.g. a single OCR-ed book would however not call for an exhaustive look-up over all possible CCs, indicating the use of the FW approach.

5.3.2 Ablation tests on CC *mode on* SGD8995

Further evaluations are performed in CC mode only. We run ablation tests in which information sources are not available to the system and test the effect of ranking the correction candidates.

Table 6 shows performance of TICCL when information sources are left out in comparison with the full system.

Measurements beyond rank 3 on ranks 4, 5 and 10 show no further fluctuations in the scores when ranking is done. Not using a validated lexicon has the greatest effect on precision. The validated lexicon and the lexicon confusables information actually have an adverse effect on recall, but greatly contribute to the level of precision attained by the full system. While their impact is slight, the Ispell dictionary and affix information do contribute about 1% to overall performance of the full system. This should be due to the effect of the morphological information being

Table 6 SGD 1989–1995: Performance results on types for the full system, compared with test runs with information sources withheld

Rank	R	P	F
Full system			
1	0.923	0.956	0.939
2	0.963	0.949	0.956
3/4/5/10	0.963	0.947	0.955
Without lexicon			
1	0.926	0.894	0.910
2	0.984	0.893	0.936
3/4/5/10	0.984	0.890	0.935
Without lexicon confusables			
1	0.926	0.901	0.913
2	0.984	0.898	0.939
3/4/5/10	0.984	0.895	0.938
Without Ispell lexicon and affix files			
1	0.920	0.940	0.930
2	0.960	0.935	0.947
3/4/5/10	0.960	0.933	0.946
Without any external information sources			
1	0.924	0.893	0.909
2	0.990	0.892	0.939
3/4/5/10	0.990	0.889	0.937
Without any ranking			
1	0.494	0.955	0.651
2	0.661	0.956	0.782
3	0.768	0.958	0.853
4	0.818	0.958	0.882
5	0.862	0.955	0.906
10	0.950	0.947	0.948
Without any ranking or external information sources			
1	0.509	0.878	0.644
2	0.705	0.893	0.788
3	0.804	0.896	0.847
4	0.849	0.895	0.872
5	0.888	0.894	0.891
10	0.978	0.889	0.932

applied by analogy to unknown words. Dutch is highly compounding, and compounds are not listed in dictionaries when their meaning is inferable from the compounding parts.

The results of the ablation study are highly informative concerning the contribution to the excellent performance of the full system. We discuss their impact on first-best ranking results. Both validated lexicon and confusables list cause some loss of recall in the full system, but this cost is offset by the gain in precision, respectively, 6.2% for the validated lexicon and 5.5% for the confusables list. The morphological information derived from Ispell, in combination with its lexicon, helps both recall (0.3%) and precision (1.6%). The combination of the three information sources in the full system causes a negligible loss of recall (0.1%) which may well be due to rounding effects, for a total gain (6.3%) in precision which may well be fully due to the validated lexicon. All language information resources omission tests showed no further divergence in scores beyond rank 3. What we do see is that from rank 2 onwards recall climbs to 99.0%. Divergence in scores on ranks higher than 3 we do observe when we omit the ranking. When the information sources are in place, we see that precision remains strong and stable, showing 0.7% gain on rank 2. Recall, however, plummets to just below 50% on rank 1 to almost fully recover to 95% on rank 10. Note that with no ranking at all in place, the system in effect randomly outputs retrieved word pairs due to the random nature of Perl hashes. When the information sources are then also left out, recall drops to 50.9% to recover better to 97.8%. Precision remains below 90% on all ranks.

We have now discussed our performance results, obtained on the development set SGD8995. We now continue with results on the held-out test set HETVOLK1918, for the full system only.

5.3.3 *Evaluation of* CC *mode on held-out* HETVOLK1918

We list the results obtained per rank on HETVOLK1918 in Table 7. We list the results on word types and on word tokens.

The performance scores on word types obtained on the held-out test set HETVOLK1918 are fully in line with those obtained on the qualitatively very different SGD8995. They represent very different text types and domains.

We claim to have developed an unsupervised, flexible, viable and competitive approach to large-scale spelling and OCR-error correction which requires no training data, adapts to the lexical variation actually present within a corpus and travels well across domains.

We developed on the SGD8995 gold standard and performed final evaluation runs on the held-out HETVOLK1918 gold standard. These are qualitatively very different corpora, the first representing state-of-the-art high accuracy OCR-ed text, the second very low-quality OCR-ed text obtained from low quality, non-OCR oriented microfilms made from low-quality paper and print. On the basis of the performance scores on the SGD and HETVOLK1918 gold standards, we conclude that given the extra information available to TICCL, i.e. the affix rules borrowed from Ispell, their application by analogy to the full corpus vocabulary and the availability of full knowledge concerning the possible confusables present in the validated lexicon within *k*, all combine to deliver a highly efficient system for the post-correction of large corpora, whatever the form or origin of the lexical variation within them. The scores show that TICCL handles typographical and OCR misrecognition errors equally well. Given its exhaustive look-up, there is no need to model for arguably different types of error.

5.4 Discussion of TICCL

5.4.1 *Variation in character confusion in different corpora*

We have so far worked under the assumption that just about anything can happen to the character strings when a corpus is digitized, depending on circumstances such as the OCR-software used, mode of scanning, quality of the input whether paper or microfilm, etc. Since our approach does not require prior error modeling, we do not risk having made incorrect or incomplete assumptions about the actual variation present in a particular corpus. If we look at the top 20 character confusions observed in our test runs on SGD8995 and HETVOLK1918, shown in Table 8, we see that at least with respect to these major character confusion classes our

Table 7 HETVOLK1918: Performance results on n-best first ranking on types and tokens for the CC approach

Level rank	Types			Tokens		
	R	P	F	R	P	F
1	0.937	0.929	0.933	0.877	0.960	0.917
2	0.957	0.928	0.942	0.881	0.959	0.919
3	0.961	0.924	0.942	0.882	0.959	0.919
4	0.963	0.923	0.942	0.883	0.958	0.919
5/10	0.963	0.923	0.942	0.883	0.958	0.919

Table 8 SGD8995 and HETVOLK1918: Observed divergence in the top 20 character confusions

SGD8995			HETVOLK 1918		
Char. conf. AV	Min. confus.	# Memb.	Char. conf. AV	Min. confus.	# Memb.
26615200501	ne/en	24087	1000200002	e ↦ c	25050
20113571875	s	20318	6340481050	e ↦ o	16662
10510100501	e	14543	7340681052	o ↦ c	12559
1930465143	i ↦ l	10911	5819380357	n ↦ u	12278
16105100000	n	9615	10510100501	e	10769
13481676076	ni ↦ m/in ↦ m	7625	7517759743	n ↦ a	8949
19254145824	r	5410	16105100000	n	8378
21003416576	t	5334	8240600951	i ↦ t	7621
1922760244	e ↦ a	4938	1930465143	i ↦ l	7429
10000000000	d	4814	2000400004	cc ↦ ee	7249
12762815625	i	4657	11414379856	e ↦ u	6997
20510100501	ed/de	4623	26615200501	ne/en	6994
184528125	–	4488	8744045323	e ↦ r	6946
29764246325	re/er	4265	6310135808	l ↦ t	6945
15911861449	ri ↦ n/ir ↦ n	3995	1922760244	e ↦ a	6858
14693280768	l	3978	5594999499	e ↦ n	6788
12344620132	i ↦ 5	3868	5073898806	o ↦ u	6726
13845455867	ig ↦ e/gi ↦ e	3627	9603471374	e ↦ s	6569
6310135808	l ↦ t	3616	20113571875	s	6386
4515616808	k ↦ c	2984	13337140100	u ↦ a	6240

Note that in anagram hashing actual adjacency of character combinations is not required. Single characters that do not map to other characters denote insertions or deletions. Characters mapping to other characters denote substitutions, but directionality is not implied: the earlier transposition example 'melonsteen' might be produced by minimal confusion 'e' to 'o' occurring twice within the same word

assumption is confirmed that the actual variation present in the text tokens may widely diverge between different text collections. The top two confusions in SGD8995 in fact reflect morphological variation in Dutch, these two being singular/plural character confusions. The top two confusions in HETVOLK1918 show the main OCR-induced confusions on the most frequent character in Dutch, i.e. 'e'.

5.4.2 Language independence

The algorithm we have presented is not language-dependent in se. In [16], we worked on both English and Dutch and built a trilingual spelling correction system by further adding French to a mixed English-Dutch system. TICCL retains this feature.

5.4.3 Distributability for parallelization

An attractive feature of character confusion-based spelling variant retrieval is the fact that the search for particular confusions can easily be distributed over as many processors or computers one has at hand. So the character confusion lookup enables easy parallelization of the full task. This should enable the method to scale to the largest corpus sizes. TICCL could easily be run-on x processors for this work by simply dividing the list of the anagram keys for the n confusions to be examined in x equal parts. This would fully ensure there is no overlap between the systems running independently and that no double work is done. All tests reported on here were run-on single Intel Xeon 3Ghz. processors.

5.4.4 Further steps

We have shown that anagram hashing provides a powerful framework for tackling lexical variation in corpora. We have shown that it allows for easy integration of external information sources regarding the language. Validated word lists and lists of named entities contribute heavily to the performance. The full mapping of confusables within the validated lexicon likewise. Morphological information, too, can easily be accommodated and contributes. The framework explored here provides a solid basis for further extension and refinement. No doubt even better best-first ranking can be achieved, by means of more refined ranking mechanisms as explored in related work. We have here limited ourselves to addressing the variation given $k = 2$. Moving beyond that is straightforward but requires more processing. Extending the alphabet with a space, given word bigrams in the lexicon, allows for addressing the problem of split and run-on words, as was shown in [16].

5.4.5 Availability

To conclude, a final remark about TICCL's availability: TICCL and the KB gold standards are to be made available under open source licenses. The corpora we have worked on here are freely available online at the KB.

6 Conclusions

In this paper, we have presented a global approach to tackling spelling variation in corpora.

We have proposed a character confusion-based look-up algorithm for identifying all the pairs of words that happen to be confused in particular characters. We have demonstrated that we can exhaustively examine and further process the word type list of a large Dutch corpus for all the character confusions up to LD 2 in a couple of hours.

We have conducted formal evaluations on a contemporary corpus and on a historical corpus both of which have been digitized by the KB, the National Library of The Netherlands. In these tests, conducted with both a development gold standard for the contemporary corpus and a held-out gold standard for the historical corpus, we have compared the focus word-based working mode of TICCL with the character confusion-based mode. Preliminary work had given us useful new insights into how the more traditional, sequential word-based approach can also be made to gather global statistics about the variation in a corpus. This should also be applicable to any other focus word-based spelling correction approach.

Our contribution is that we have proposed an unsupervised viable alternative based on anagram hashing for spelling correction purposes of large OCR-ed corpora to the mainstream source model and Finite State approaches, both of which need costly training materials. We have shown that by focusing not on each word string individually but rather on the character confusions that sets of word pairs have in common, useful statistics are gained which enhance the ranking of correction candidates and help to provide not only great recall, but also sufficiently high precision for the process to be run automatically, in an unsupervised fashion.

We have in this paper demonstrated that the framework of anagram hashing allows for easy incorporation of useful sources of information helpful to the process of fully automatic, unsupervised clean-up of large OCR-ed corpora. We have shown that morphological information may be applied by analogy to unknown words, which helps in highly compounding languages such as Dutch.

The CC approach has been demonstrated to be fast on large corpora. The FW approach remains viable, and indicated, for smaller clean-up tasks such as a single OCR-ed book.

Faced with the huge scale of the current digitization programmes, the inherent distributability of the CC approach is a valuable asset. The numerical work to be performed on just two lists of numbers can very straightforwardly be distributed over as many processors as one has at one's disposal. Given the right hardware, i.e. sufficient numbers of processors at hand, the gigantic task of cleaning-up say one hundred years' worth of SGD OCR-ed text might well soon be performed in a single day.

Acknowledgements We are grateful to our anonymous reviewers for their rightful criticisms of our first draft. We like to thank our contacts at the KB for their support and patience: Paul Doorenbosch, Astrid Verheusen, Tineke Koster en Evelien Ket. Heartfelt thanks to scientific programmer Ko van der Sloot at ILK, whose reimplementation of our basic ideas demonstrated to us their essence. Early TICCL prototypes were developed within a Netherlands Organization for Scientific Research (NWO) Exact Sciences Hefboom project. The production version of TICCL was commissioned by the Koninklijke Bibliotheek - Den Haag. Development continues under the Stevin project SoNaR (STE07014). TICCL was turned into the online processing system TICCLops with funding from CLARIN-NL (CLARIN-NL-09-011).

References

1. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001)
2. Kukich, K.: Techniques for automatically correcting words in text. ACM Comput. Surv. **24**(4), 377–439 (1992)
3. Cucerzan, S., Brill, E.: Spelling correction as an iterative process that exploits the collective knowledge of web users. In: Lin, D., Wu, D. (eds.) Proceedings of EMNLP 2004, pp. 293–300. Association for Computational Linguistics, Barcelona (2004)
4. Lopresti, D., Zhou, J.: Using consensus sequence voting to correct OCR errors. Comput. Vis. Image Underst. **67**(1), 39–47 (1997)
5. Kernighan, M.D., Church, K.W., Gale, W.A.: A spelling correction program based on a noisy channel model. In: COLING-90, vol. II, pp. 205–211. Helsinki (1990)
6. Oflazer, K., Güzey, C.: Spelling correction in agglutinative languages. In: ANLP, pp. 194–195. (1994)
7. Sun, X., Gao, J., Micol, D., Quirk, C.: Learning phrase-based spelling error models from clickthrough data. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10) (2010)
8. Teahan, W.J., Inglis, S., Cleary, J.G., Holmes, G.: Correcting English text using PPM models. In: Storer, J.A., Reif, J.H. Proc Data Compression Conference, pp. 289–298. IEEE Computer Society Press, Society Press, Los Alamitos, CA (1998)
9. Kolak, O., Resnik, P.: OCR error correction using a noisy channel model. In: Proceedings of the second international conference on Human Language Technology Research, pp. 257–262. Morgan Kaufmann Publishers Inc., San Francisco, CA, (2002)

10. Brill, E., Moore, R.C.: An improved error model for noisy channel spelling correction. In: Proceedings of the 38th Annual Meeting of the ACL, pp. 286–293. (2000)
11. Strohmaier, C.M., Ringlstetter, C., Schulz, K.U., Mihov, S.: Lexical postcorrection of OCR-results: the web as a dynamic secondary dictionary? In: International Conference on Document Analysis and Recognition 2:1133 (2003)
12. Ringlstetter, C., Schulz, K.U., Mihov, S.: Orthographic errors in web pages: toward cleaner web corpora. Comput. Linguist. **32**(3), 295–340 (2006)
13. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. In: Cybernetics and Control Theory, vol. 10(8), pp. 707–710 (1965), original in: Doklady Nauk SSSR 163(4):845–848 (1965)
14. Gotscharek, A., Neumann, A., Reffle, U., Ringlstetter, C., Schulz, K.U.: Enabling information retrieval on historical document collections: the role of matching procedures and special lexica. In: AND '09: Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data, pp. 69–76. ACM, New York, NY (2009)
15. Reynaert, M.: Text induced spelling correction. In: Proceedings COLING 2004, Geneva (2004)
16. Reynaert, M.: Text-induced spelling correction. PhD thesis, Tilburg University (2005)
17. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Commun. ACM **7**(3), 171–176 (1964)
18. Reynaert, M.: Non-interactive OCR post-correction for giga-scale digitization projects. In: Proceedings of CICLing 2008. Lecture Notes in Computer Science vol. 4919/2008, pp. 617–630. Springer, Berlin (2008)
19. Reynaert, M.: Parallel identification of the spelling variants in corpora. In: Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data 2009 (AND-2009), pp. 77–84. Barcelona, Spain (2009)
20. Frauenfelder, U., Baayen, R., Hellwig, F., Schreuder, R.: Neighbourhood density and frequency across languages and modalities. J. Mem. Lang. **32**, 781–804 (1993)
21. Zipf, G.K.: The psycho-biology of language: an introduction to dynamic philology, 2nd edn. The M.I.T. Press, Cambridge, MA (1935)
22. van Rijsbergen, C.J.: Information Retrieval. Butterworths, London (1975)