# Precise and Efficient Text Correction using Levenshtein Automata, Dynamic Web Dictionaries and Optimized Correction Models

Stoyan Mihov[*], Svetla Koeva
Bulgarian Academy of Sciences
stoyan@lml.bas.bg

Christoph Ringlstetter[†], Klaus U. Schulz[†], Christian Strohmaier
CIS – Univ. of Munich
{kristof,schulz,strohmai}@cis.uni-muenchen.de

## Abstract

*Despite of the high quality of commercial tools for optical character recognition (OCR) the number of OCR-errors in scanned documents remains intolerable for many applications. We describe an approach to lexical postcorrection of OCR-results developed in our groups at the universities of Munich and Sofia in the framework of two research projects. Some characteristic features are the following:*
*(1) On the dictionary side, very large dictionaries for languages such as German, Bulgarian, English, Russian etc. are enriched with special dictionaries for proper names, geographic names and acronyms. For postcorrection of texts in a specific thematic area we also compute "dynamic" dictionaries via analysis of web pages that fit the given thematic area.*
*(2) Given a joint background dictionary for postcorrection, we have developed very fast methods for selecting a suitable set of correction candidates for a garbled word of the OCR output text.*
*(3) In a second step, correction candidates are ranked. Our ranking mechanism is based on a number of parameters that determine the influence of features of correction suggestions such as word frequency, edit-distance and others. A complex tool has been developed for optimizing these parameters on the basis of ground truth data.*
*Our evaluation results cover a variety of corpora and show that postcorrection improves the quality even for scanned texts with a very small number of OCR-errors.*

## 1 Introduction

The quality offered by the leading commercial software systems for optical character recognition (OCR) is quite impressive. For printed documents with standard fonts, typically only a very small number of errors per page is observed. Even in this situation, for many potential applications of OCR-technology, the number of errors is too high, and the actual use of OCR-technology for document reproduction, say, in publishing companies and libraries is still very limited. Hence, good methods for postcorrection of OCR-results may help to improve the acceptance of OCR-technology and to increase the number of industrial application fields.

In this paper we describe an approach to lexical postcorrection of OCR-results that was developed in our research groups at the University of Munich and the Bulgarian Academy of Sciences during the last years (ongoing research). We concentrate on methods for *automated* postcorrection, mainly for the following reasons:

1. The aforementioned high accuracy of commercial OCR-systems heavily depends on established methods for automated linguistic postcorrection that are used "inside" these systems, using appropriate language

[*]Funded by VolkswagenStiftung
[†]Funded by Deutsche Forschungsgemeinschaft

resources. This becomes obvious when applying commercial OCR-software to non-linguistic (statistically generated) text. In such a situation, a dramatic increase of the number of errors is observed. Hence, any improvement of automated methods for postcorrection may be used to improve the quality of commercial OCR-systems, using the new methods inside the system.

2. In a straightforward sense, any technique that helps to improve automated postcorrection can also be used to simplify and/or improve methods for interactive postcorrection. As an example, consider improved methods for error detection, or methods for ranking of potential correction candidates.

Though we only comment on postcorrection of OCR-results it should be clear that many techniques and methods described below can also be used for other correction tasks. In fact some of the following ideas have been successfully exported to other applications domains (e.g., spell checking, repairing queries to internet search engines).

The paper is structured as follows. In the next section we collect some characteristics for the specific errors that are found in OCR-results. Section 3 discusses the lexical coverage of conventional (static) dictionaries for real-world texts found in scanning tasks. Section 4 describes the use of "dynamic" lexica for the treatment of texts from specific thematic areas. In our experiments, such dictionaries have been computed automatically by analyzing web pages of the given field. These lexica lead to drastic improvements in terms of lexical coverage and correction accuracy. In Section 5 we give a brief summary of our method for fast computation of correction candidates for garbled input words in large dictionaries. In Section 6 we sketch our model for postcorrection and our tool for parameter optimization of postprocessing systems. Section 7 illustrates our approach, presenting evaluation results for several corpora in English, German and Bulgarian, covering correction with standard and dynamic dictionaries. Section 8 introduces an a posteriori classification of the remaining errors after postprocessing, from which we get valuable information on the trade-off between different correction strategies. Finally in Section 9 we give concluding remarks and an outlook on further research.

## 2 Some characteristics of OCR-errors

In many cases, the recognition accuracy (number of correctly recognized words) of commercial OCR-systems achieved for printed documents with standard fonts is very high. In a NIST report from 2002, average word accuracies of 97.44% for original documents and 95.05% for copied documents have been reported [2]. In our own experiments for ICDAR'03 [10] we even found accuracies higher than 98.50%. Obviously, given these numbers, any kind of automated postcorrection becomes a challenge. As an indispensable prerequisite, background dictionaries are needed that cover almost every word of the input text. For most errors, the edit distance[1] between original word and garbled OCR-result is only 1 or 2.[2] Thus, selecting correction candidates from the dictionary with distance $\leq 3$ to the input is sufficient for practical purposes. In practice, the frequency of edit operations that are found in OCR-errors depends on the symbols. A substitution of $i$ and $j$ is more likely than, say, a substitution of $i$ and $m$. Hence, for ranking of correction candidates is makes sense to refine the standard edit distance, using weights (penalties) that depend on the symbols. The selection of appropriate weights is an interesting point of research [7, 5, 8].

In our experiments we found a new source of OCR errors for documents which contain both – Latin and Cyrillic letters. The problem is that there are a number of Latin-Cyrillic letter pairs which have the very same graphical representation. For example the Latin letter $p$ has the same graphical representation as the Cyrillic letter $r$. There are approximately 12 such pairs depending on the font. In many cases whole Cyrillic words are recognized as Latin strings. For example the Bulgarian word *paca* is often mistaken as a Latin string. Although graphically the recognized Latin string looks exactly the same like the Cyrillic word, the resulting encoding is entirely different. Because of that we have problems applying electronic processing tasks like indexing, searching, etc on the OCR documents. As a result we had a significantly higher error rate for the OCR on documents with mixed Latin-Cyrillic

[1] The *edit distance* or *Levenshtein-distance* between two words $u$ and $v$ is the minimal number of symbol insertions, deletions or substitutions that are needed to transform $u$ into $v$. Variants of this distance, also considered in our experiments, treat also splits and merges of symbols as primitive edit operations.

[2] In our experiments for ICDAR03 [10, 7] more than 99% of the errors came with a Levenshtein distance $\leq 3$ and more than 95% with a distance $\leq 2$.

alphabets. For the tested corpus we have an error rate of 14.33%. Most of the errors are due to the Latin-Cyrillic similarity.

For handling the Latin-Cyrillic similarity problem we extended our correction method by treating those symbol pairs as equivalent symbols. This means that words like *paca* are considered as written in Cyrillic as well as in Latin. We look up the Cyrillic encoding in the Bulgarian dictionary and the corresponding Latin encoding in the English dictionary.

## 3 Coverage of conventional static dictionaries

An important parameter for the quality of lexical postprocessing systems is the coverage of the dictionaries that are used for the correction task. Given a corpus $C$, the lexical coverage of a dictionary $D$ is defined as the percentage of normal tokens of $C$ that occur in $D$. By a "normal" token, we mean a token that is only composed of standard letters. Unnormal in this sense, are e.g. "class_definition", "#0049-89-2180-2715".[3] Small dictionaries with the most frequent words can only help to improve the quality of texts where the OCR has a low recognition rate. For random samples of general corpora as Trec5 (English) or for a large newspaper corpus (German) with the lexica of the 10,000 most frequent words[4] only a coverage of 89.97% (English) and 79.50% (German) is reached. In typical applications, only a few errors per page are observed. In these cases, lexical postcorrection can only improve results if a coverage of almost 100% is reached with the background dictionary. In our experiments, we first used conventional "static" dictionaries for English (330.000 entries), German (2.300.000 entries) and Bulgarian (975.000 entries). With this equipment we reached a coverage of 97.24% (English), 92.87% (German) and 96.59% (Bulgarian) on the aforementioned documents. As a first improvement, we added special dictionaries for geological entities, proper names, acronyms and abbreviations leading to sizes of English (750.000 entries) and German (2.700.000 entries). The effect of this supplementation in coverage heavily depends on the specific vocabulary of each single text. For the tests on the standard texts we finally reached a coverage of 98.11% (English) and 96,51 (German).

## 4 Use of dynamic web dictionaries

In a series of preliminary experiments we found that most tokens occurring in a text of a given area can be found in web pages with a direct relationship to the given topic. We thus developed a method for automatically computing "dynamic" dictionaries for the correction of specific documents via an analysis of suitable web pages. 25 non-function words of the given thematic OCR-corpus were extracted and sent as a disjunctive query to a standard web search engine.[5] The vocabulary of the 1000 top-ranked pages in the answer set was used to build a dynamic thematic dictionary. Obviously this procedure is simple and naive and can be improved in many respects - a point of ongoing research. Nevertheless, in each case we obtained a better lexical coverage than in the parallel experiments with maximal static lexica [10]. Later, using these kind of dynamic dictionaries for lexical postcorrection (see section 7) we again observed a considerable improvement. This means that the negative influence of incorrect words occurring in web pages can be widely ignored.

The average size of the dynamic dictionaries for our experiments on a thematic corpus was about 450.000 entries. As they are much smaller than the combination of the static lexica and at the same time have a higher coverage, they combine the advantages of "small lexicons" described by Peterson in [6] with the advantages of "large lexicons" as stated by Damerau/Mays in [1].

## 5 Using Levenshtein automata for efficient selection of correction candidates

Given a possibly garbled input word $w^{ocr}$ received from the OCR, our postcorrection component first computes a list of possible correction candidates from the background dictionary $D$ selected for correction. In our approach,

[3] Note that usually only normal tokens are collected in dictionaries. In our research, correction of unnormal tokens has yet not been investigated.

[4] using web frequencies

[5] For a more detailed description of the lexicon acquisition procedure see [10]. In this paper we also report on experiments using an OCR corpus of different thematic fields in German and English. We comment on lexical coverage and accuracy of postprocessing results with and without dynamic dictionaries.

we select as correction candidates all words $w^{cand}$ in the dictionary where the Levenshtein distance between $w^{ocr}$ and $w^{cand}$ does not exceed a given bound $k$. In practice, bounds $k > 3$ lead to an unacceptable number of correction candidates. For most applications, bounds $k = 1, 2$ are useful, in some cases a bound $k = 3$ makes sense.

In order to meet industrial requirements, the selection of such a list of correction candidates must be very fast. This leads to the following algorithmic problem: Given a dictionary, $D$, and a bound $k \in \{1, 2, 3\}$, find an efficient method to select for a given input string $u$ all words $v$ of $D$ where the Levenshtein distance between $u$ and $v$ does not exceed $k$.

In [9] and [4], we introduced several solutions to this problem. We assume that the dictionary is implemented as a deterministic finite state automaton. The basic method in [4] depends on the concept of a universal deterministic Levenshtein automaton of fixed degree $k$. The automaton of degree $k$ may be used to decide for arbitrary words $u$ and $v$ if the Levenshtein distance between $u$ and $v$ does not exceed $k$. The automaton is "universal" in the sense that it does not depend on $u$ and $v$. The input of the automaton is a sequence of bitvectors computed from $u$ and $v$. The distance satisfies the bound iff the automaton accepts the sequence. The selection of the list of candidates from the dictionary is realized as a parallel backtracking traversal of the universal Levenshtein automaton and the dictionary automaton. Under one perspective, the universal Levenshtein automaton is used to control the walk in the dictionary automaton. Blind paths that cannot lead to correction candidates are recognized soon, which means that in practice only a small part of the dictionary automaton is traversed. From a more formal point, the joint traversal can be considered as the computation of the intersection of two regular languages: (1) the language of the dictionary, and (2) the language of all alphabetical tokens in distance $\leq k$ to the input word.

Even if many blind paths are recognized soon, the number of useless traversal steps that do not lead to correction candidates is high when using the above basic method. One reason is the high number of outgoing transitions for states of the dictionary automaton that are close to the input state. This leads to many choice points and a high amount of backtracking. In [4], a refinement of the basic methods is suggested that leads to drastic improvements concerning correction times. In this approach, an additional "backwards" dictionary $D^R$ (representing the set of all reverses of the words of a given dictionary $D$) is used to reduce approximate search in $D$ with a given bound $k \geq 1$ to related search problems for smaller bounds $k' < k$ in $D$ and $D^R$. To this end, the given input word is split into two parts. We cannot completely avoid search with bound the original bound $k$. However, search with bound $k$ only starts when we have already passed through the first part of the input word. In the dictionary automaton we have then reached a state not too close to the start state. For these states the branching degree is small in most cases. Hence, the danger of useless backtracking steps is considerably reduced, which leads to smaller search times. Reversed dictionaries are necessary in order to cover the cases where $k$ errors occur in the first part of the input string. These cases are treated using reverse dictionaries and reversed input words, which means that the errors again occur in the second part of the joint traversal. The dictionaries at our disposal, we have to manage the use of different information resources during the correction process.

With the refined method, the correction times per word are between a few microseconds and a millisecond, depending on the bound $k$, even for dictionaries with millions of entries.

## 6 Postcorrection model, training and parameter optimization

In the following we describe our approach to lexical postcorrection using a single OCR-engine and a collection of dictionaries $\mathcal{D}$.[6] For each token $w^{ocr}$ recognized by the given OCR engine and each dictionary $D \in \mathcal{D}$ we preselect a list of $n$ *correction candidates* in $D$. Here $n$ is an (adjustable) parameter. For preselection, the standard Levenshtein distance $d_0$ is used. Entries $v$ of $D$ where $d_0(w^{ocr}, v)$ is small are preferred. If $d(w^{ocr}, v) = d(w^{ocr}, v')$ for two distinct entries $v, v'$ of $D_i$, then the more frequent entry is preferred. Preselection of correction candidates only represents a filtering step that narrows down the search for good correction suggestions. The choice of the value $n$ yields a compromise between recall and computational efficiency.

The next step is split into rounds. Each round begins with the selection of a configuration. A configuration is given by a subcollection $\mathcal{D}' \subseteq \mathcal{D}$ of active dictionaries and an active distance measure $d \in \mathcal{M}$. In order to select a configuration, a subcollection $\mathcal{D}'$ and a distance measure $d \in \mathcal{M}$ may be activated using buttons of a graphical user interface. For a given configuration $(\mathcal{D}', d)$, an interactive and semi-automated process is used to determine a unary correction model which leads to optimal correction accuracy. With a unary correction model we mean

[6] For the $k$-ary case using more than one OCR-engine see the forthcoming [11].

any deterministic procedure that computes for each recognized token $w^{ocr}$ a unique correction suggestion $w^{cor}$ which is $w^{ocr}$ itself or an(other) entry of an active dictionary. Correction accuracy is defined as the percentage of tokens $w^{ocr}$ where $w^{cor}$ coincides with the correct token $w$. Values for the correction accuracy reached for distinct configurations are compared in order to determine a configuration that leads to optimal accuracy. In our system, unary correction models are completely determined by the actual configuration $(\mathcal{D}', d)$ and values for two parameters $\alpha$, $\tau$ that are described below. In the sequel we assume that distance values $d(v, w^{ocr})$ are translated into similarity values $s(v, w^{ocr}) \in [0, 1]$ where high values mean high similarity.

*Balance.* The *score* of a correction candidate $v$ for a token $w^{ocr}$ is $sc(v) := \alpha \cdot s(v, w^{ocr}) + (1 - \alpha)f(v)$. Here $f(v) \in [0, 1]$ is a normalized frequency value for candidate $v$. The *balance parameter* $\alpha$ is a value in $[0, 1]$ that determines the relative weight of similarity versus frequency. Since frequency and distance values are of distinct nature and normalized in distinct ways, this gives only a basic intuition. For selecting a value for $\alpha$ the graphical user interface offers two options. The value can be defined interactively, using a scroll bar. Alternatively we may automatically compute a value $\alpha$ that leads to optimal correction accuracy.

*Threshold.* If the error rate of the OCR is low, it does not make sense to automatically replace every OCR-token that is not found in the dictionary by the best correction candidate. Instead, we override the OCR-result only in the presence of additional confidence. In the following, assume that a balance parameter $\alpha$ has been fixed. Consider a recognized token $w^{ocr}$. Using the correction files, among all correction candidates for $w^{ocr}$ that have been preselected in *active* dictionaries we select the entry $v^{cand}$ with the highest score. Collecting the entries $v^{cand}$ for all tokens $w^{ocr}$ we create a list *Cands* with tuples of the form $\langle w, w^{ocr}, v^{cand}, sc(v^{cand})\rangle$. *Cands* is sorted in descending score order, The sorted list is divided into two parts. The "active part" contains an initial interval where entries have high scores. For the tokens $w^{ocr}$ of tuples in the active part we define $w^{cor} := v^{cand}$. As a matter of fact, very often $w^{ocr}$ is found in an active dictionary and we get $w^{cor} := v^{cand} = w^{ocr}$. The remaining "passive part" contains tuples with low scores. Here we generally define $w^{cor} := w^{ocr}$. The *threshold parameter* $\tau$ defines the minimal score for elements of the sorted list *Cands* that belong to the active part. The value of the threshold parameter can be assigned interactively, using a scroll bar. Alternatively we may automatically determine a value $\tau$ that leads to optimal correction accuracy.

*Optimization.* Appropriate values for the parameters $\alpha$ and $\tau$ can be computed using a simple hill climbing procedure. For an initial pair of values $(\alpha_0, \tau_0)$, which is selected arbitrarily, the system automatically computes the correction accuracy that is obtained. Fixing the value for $\tau_0$ we use the system to compute a value $\alpha_1$ that leads to optimal correction accuracy for the given threshold $\tau_0$. Fixing then $\alpha_1$ we compute a value $\tau_1$ that leads to optimal correction accuracy for the given value of the balance parameter, $\alpha_1$. In this way we continue until a local maximum is found.

*Training.* For the postprocessing of a long document or a bigger corpus of similar documents we run the described optimization process on a suitable training corpus. In this case additional improvements can be achieved by processing document specific symbol weights for a generalized Levenshtein distance. If only a short document is to be processed we use standard values for the parameters and a standard symbol weight table.

## 7 Evaluation Experiments: Trec5, Thematic Corpus, Mixed Alphabet Corpus

The usefulness of our approach is demonstrated in a first experiment for an extract of a standard OCR-corpus the TREC-5 [3] with 95% OCR-accuracy. For lexical postprocessing we use the English standard dictionary and for comparison we run the same tests first with a smaller dictionary of the most frequent 10.000 English words and then with a dynamic crawl dictionary of 35.000 entries. In a second experiment we show postprocessing for a sample of a Bulgarian OCR-corpus. Due to the domain specific case we give in the third experiment, a survey of our experiments on a thematic corpus for ICDAR03 [10] showing that successful automatic correction with dynamic lexica and a careful correction strategy is possible even for very accurate OCR-documents. An other result here is that even very big standard lexica, are inferior to the dynamic web lexica, generated for the specific domains.

### 7.1 Experiments on TREC-5: English

*Method.* Here we give an experiment on the quality of our parameterized postprocessing approach on a small sample of TREC-5 documents[7] using the English standard language dictionary. The documents were divided in training- and testdata in a ratio of 3 to 7. An optimal parameter $\alpha$ determining the relative weight between length sensitive Levenshtein distance versus word-frequency and an optimal correction threshold $\tau$ were computed by optimizing the correction accuracy of the training data (see section 6). This parameter values were then used for the test-phase.
*Results.* The experiments showed an improvement of the percentage of accuracy using the English standard lexicon between 3.3 and 5.5 for the training case and of 3.6 to 8.8 for the test case.[8] The improvement for the test data are naturally not as high as they would be for the overtrained case.[9] As all correction candidates were selected by a Levenshtein automaton with a filter distance ($\leq 2$) it can not be deduced from the values of $\alpha$, that frequency information generally is more important than distance information. The threshold values $\tau$ in their absolute values depending on the chosen $\alpha$ can be translated into a percentage of the possible corrections actually executed. The percentages for this experiment stay in a corridor between 75% using the dictionary of the 10,000 most frequent English words and over 90 % using the crawled dictionary. This corridor could be a clue for estimating a default value for $\tau$ in cases were training is not appropriate (ongoing research). An interesting result are the figures of the correction accuracies rising from using a small dictionary, the most frequent 10,000 English words, over that achieved by our English standard dictionary with 330,000 entries and finally the highest accuracies with the crawled lexicon of only 35.000 entries but a slight better coverage than the standard lexicon.[10] This improvement in correction accuracy reflects the coverage of the used dictionary ("recall") on the one hand and the absence of superfluous words ("precision") on the other hand, finding it's upper bound in the document perfect dictionary, the groundtruth document itself.

### 7.2 Experiments on a Thematic Corpus: English and German

*Method.* Here we describe an experiment on the quality of our parameterized postprocessing approach on very accurate OCR-documents using dynamic web dictionaries. The experiment was run on four randomly generated thematic corpora of the topics *Holocaust, Roman Empire, Mushrooms* and *Neurology* in English and German, OCRed by two leading OCR-engines. All corpora were divided in training- and testdata in a ratio of 3 to 7. As lexical basis we used the crawled dynamic dictionaries and for a counterexperiment a union of all static dictionaries. An optimal parameter $\alpha$ determining the relative weight between length sensitive Levenshtein distance versus word-frequency and an optimal correction threshold $\tau$ were computed by optimizing the correction accuracy of the training data (see section 6). This parameter values were then used for the test-phase.
*Results.* In the training phase all experiments showed an improvement of accuracy up to one percent and in no case despite high base accuracies the use of the dynamic lexica lead to a misscorrection. This supports our hypothesis of the usefulness of crawled dynamic dictionaries. In a counterexperiment we could show that for the training data in no case the union of the static lexica lead to a better accuracy than the use of the dynamic lexica.[11]
The improvement for the test data using the dynamic dictionaries is in general a not as good as the training results but holding for all experiments. As all correction candidates were selected by a Levenshtein automaton with a filter distance ($\leq 2$) it can not be deduced from the values of $\alpha$, that frequency information generally is more important than distance information. Comparing the results for the two languages we noticed by different values of $\alpha$ that the Levenshtein distance is more important for German than for English. If this can be confirmed by further research it could be a valuable information for postprocessing without document specific training.

[7] We took three samples with 3335, 3943 and 3919 "normal" Tokens of fr940202.0 and one sample with 4820 "normal" tokens of fr940328.2.

[8] The higher improvement in absolute numbers of the testdata compared with the training data explains in most of the cases by the negative correlation between ground accuracy and average improvement by postprocessing.

[9] The losses are e. g. 0.22,3.16,0.08,0.03 for the English standard dictionary and 0.40, 0.04,0.21,0.00 for the crawl dictionary, with the 3.16 obviously as an outlier. The quantification of test/training losses for our method is an ongoing research thus the given figures show only first trends.

[10] The dynamic dictionary was crawled by Web pages found by a query with the 15 most frequent content terms of document "Trec-5-1". Here we used only the first 100 results hoping to get a smaller but more precise dictionary than in our other experiments where we used the first 1000 results of the search engine.

[11] For details see [10].

### 7.3 Experiments on a Mixed Alphabet Corpus: Bulgarian

For evaluating our postcorrection method for mixed alphabets we used a Bulgarian corpus consisting of the translated European law texts. It contains many English words distributed in the documents. The texts were printed with the Universum font. This font is often used in the informative prose and causes many OCR errors due to the Latin-Cyrillic letter similarities. We have tried both OCR engines but the second one performed very poorly on the documents. On Table 6 we present the results which show a drastic improvement especially in the case of the second OCR engine. The significant post-correction is achieved mainly because of the successful correction of the Latin-Cyrillic similarity errors.

## 8 A Classification of Post-Correction Errors

Consider a situation where we have fixed a unary correction model in terms of a configuration $(\mathcal{D}', d)$ and suitable values for the parameters $\alpha$, $\tau$. Let $\hat{D}$ denote the union of all active dictionaries. For simplicity here we assume that $w^{cor} = w^{ocr}$ as soon as $w^{ocr} \in \hat{D}$. With a correction error we mean any triple $(w, w^{ocr}, w^{cor})$ where $w \neq w^{cor}$. In our system, correction errors are classified using seven disjoint categories:

1. "false friends": $w \neq w^{ocr} \in \hat{D}$ and $w^{cor} = w^{ocr}$.
2. The OCR-result is actively corrected, with wrong result: $w^{ocr} \notin \hat{D}$ and $w^{cor} \neq w^{ocr}$. We distinguish three subcategories:
   (a) "wrong candidate" errors where $w \in \hat{D}$ (here $w \neq w^{ocr}$),
   (b) "infelicitous correction" errors where $w \notin \hat{D}$ and $w = w^{ocr}$, and
   (c) "no chance I" errors where $w \notin \hat{D}$ and $w \neq w^{ocr}$.
3. The OCR-result is left unmodified, with wrong result: $w^{ocr} \notin \hat{D}$ and $w^{cor} = w^{ocr}$. We distinguish three subcategories:
   (a) "too cautious" errors where $v^{cand} = w$,
   (b) "wrong candidate and threshold" errors where $v^{cand} \neq w$ but $w \in \hat{D}$, and
   (c) "no chance II" errors where $v^{cand} \neq w$ and $w \notin \hat{D}$.

Our system automatically provides the user with an error analysis where for each of the seven categories the number (and percentage) of correction errors is depicted. These kind of statistics can be used to recognize deficits of the current correction model. For example, "false friend" errors can only be avoided with a smaller dictionary. In contrast, a large number of "no chance" errors indicates that the coverage obtained by the set of active dictionaries is yet insufficient. A large number of "wrong candidate" errors typically indicates that the selected distance measure is not optimal or the number $n$ of preselected correction candidates per dictionary is too small. "Too cautious" errors can be avoided with a more liberal threshold. However, if the threshold is already optimal a larger value will lead to more "infelicitous corrections". For any combination of error categories, the list of all errors triples $(w, w^{ocr}, w^{cor})$ of the given error types can be displayed, separating triples from the active part from triples in the passive part, which may yield further insights about error sources and possible improvements.

## 9 Conclusion and Outlook

We argued that for lexical postcorrection of high-quality output texts of commercial OCR-systems (very large static lexica enriched with) dynamic web lexica for domain specific documents are most suitable. In our approach, the problem of efficiently computing a suitable set of correction candidates for a garbled input from the dictionary is solved with novel methods based on the concept of Levenshtein automata. For ranking of correction candidates, the actual system optimizes parameters that regulate the relative influence of word distances and frequencies in the ranking mechanism. In future versions of the system, further scores will be considered in the ranking mechanism (e.g., symbol-dependent edit weights, collocation information). We have also started to combine the results of two OCR-software systems in the postcorrection step, which leads to promising results.

## References


[1] F. J. Damerau and E. Mays. An examination of undetected typing errors. *International Journal of Document Analysis and Recognition*, 25(6):659–664, 1989.

[2] ISRI Staff. OCR Accuracy Produced by the current DOE Document Conversion System. Technical Report 2002-06, Information Science Research Institute(ISRI), University of Nevada Las Vegas, 2002.

[3] P. B. Kantor and E. M. Voorhees. The TREC-5 confusion track: Comparing retrieval methods for scanned text. *Information Retrieval*, 2(2/3):165–176, 2000.

[4] S. Mihov and K. U. Schulz. Fast approximate search in large dictionaries. *Computational Linguistics*, 2004. To appear.

[5] B. J. Oommen and R. K. Loke. Pattern recognition of strings with substitutions, insertions, deletions and generalized transposition. *Pattern Recognition*, 30(7):789–800, 1997.

[6] J. L. Petersen. A note on undetected typing errors. *Communications of the ACM*, 29(7):633–637, 1986.

[7] C. Ringlstetter. OCR-Korrektur und Bestimmung von Levenshtein-Gewichten. Master's thesis, LMU, University of Munich, 2003.

[8] E. S. Ristad and P. N. Yianilos. Learning string edit distance. In *Proc. 14th International Conference on Machine Learning*, pages 287–295. Morgan Kaufmann, 1997.

[9] K. U. Schulz and S. Mihov. Fast String Correction with Levenshtein-Automata. *International Journal of Document Analysis and Recognition*, 5(1):67–85, 2002.

[10] C. Strohmaier, C. Ringlstetter, K. U. Schulz, and S. Mihov. Lexical postcorrection of OCR-results: The web as a dynamic secondary dictionary? In *Proc. of the Seventh International Conference on Document Analysis and Recognition (ICDAR 03)*, pages 1133–1137, 2003.

[11] C. Strohmaier, C. Ringlstetter, K. U. Schulz, and S. Mihov. A visual and interactive tool for optimizing lexical postcorrection of OCR results. In *Proceedings of the IEEE Workshop on Document Image Analysis and Recognition, DIAR'03*, 2003.

| Corpus | Training-Improvement | | Test-Improvement | | $\alpha$ | $\tau$ |
|---|---|---|---|---|---|---|
| TREC-5-1 | 92.25%→97.22% | +4.97 | 86.86%→95.68% | +8.82 | 0.10 | 0.100894 |
| TREC-5-2 | 90.67%→94.03% | +3.36 | 89.54%→93.53% | +3.99 | 0.20 | 0.197556 |
| TREC-5-3 | 87.11%→92.62% | +5.51 | 91.68%→95.29% | +3.61 | 0.15 | 0.147604 |
| TREC-5-4 | 91.85%→96.43% | +4.58 | 87.08%→92.80% | +5.72 | 0.05 | 0.061515 |

**Table 1. Postprocessing results for 4 random samples of TREC-5 with English standard language lexicon**

| Corpus | Training-Improvement | | Test-Improvement | | $\alpha$ | $\tau$ |
|---|---|---|---|---|---|---|
| TREC-5-1 | 92.25%→95.12% | +2.87 | 86.86%→92.62% | +5.76 | 0.15 | 0.153618 |
| TREC-5-2 | 90.67%→92.18% | +1.51 | 89.54%→91.10% | +1.56 | 0.20 | 0.199409 |
| TREC-5-3 | 87.11%→91.04% | +3.93 | 91.68%→92.17% | +0.49 | 0.05 | 0.064788 |
| TREC-5-4 | 91.85%→94.28% | +2.43 | 87.08%→90.85% | +3.77 | 0.10 | 0.109521 |

**Table 2. Postprocessing results for 4 random samples of TREC-5 with the dictionary of the 10000 most frequent English words, with values for $\alpha$ and $\tau$ trained on the 10000 lexicon**

| Corpus | Training-Improvement | | Test-Improvement | | $\alpha$ | $\tau$ |
|---|---|---|---|---|---|---|
| TREC-5-1 | 92.25%→98.95% | +6.70 | 86.86%→97.03% | +10.17 | 0.10 | 0.088864 |
| TREC-5-2 | 90.67%→95.71% | +5.04 | 89.54%→97.89% | +8.35 | 0.10 | 0.087994 |
| TREC-5-3 | 87.11%→94.68% | +7.57 | 91.68%→96.59% | +4.91 | 0.10 | 0.088606 |
| TREC-5-4 | 91.85%→ 95.89% | +4.04 | 87.08%→ 94.51% | +7.44 | 0.10 | 0.088864 |

**Table 3. Postprocessing results for 4 random samples of TREC-5 with a dynamic Web dictionary with 35,000 entries, values for $\alpha$ and $\tau$ trained on the web dictionary**

| | Training-Improvement | | Test-Improvement | | $\alpha$ | $\tau$ |
|---|---|---|---|---|---|---|
| Neurology (E) | | | | | | |
| OCR-Engine$_1$ | 98.81%→99.69% | +0.88 | 98.71%→98.95% | +0.24 | 0.10 | 0.102920 |
| OCR-Engine$_2$ | 97.84%→98.82% | +0.98 | 98.64%→99.10% | +0.46 | 0.10 | 0.100483 |
| Holocaust (E) | | | | | | |
| OCR-Engine$_1$ | 98.50%→98.87% | +0.37 | 99.01%→99.16% | +0.15 | 0.10 | 0.10771 |
| OCR-Engine$_2$ | 98.04%→98.46% | +0.42 | 98.65%→98.71% | +0.06 | 0.10 | 0.106948 |
| Rom. Emp. (E) | | | | | | |
| OCR-Engine$_1$ | 98.89%→99.53% | +0.64 | 98.80%→98.98% | +0.18 | 0.05 | 0.061064 |
| OCR-Engine$_2$ | 99.19%→99.62% | +0.43 | 98.93%→99.14% | +0.21 | 0.10 | 0.1031084 |
| Mushrooms (E) | | | | | | |
| OCR-Engine$_1$ | 99.08%→99.54% | +0.46 | 98.98%→99.38% | +0.40 | 0.10 | 0.10520 |
| OCR-Engine$_2$ | 98.76%→98.90% | +0.14 | 98.76%→98.88% | +0.12 | 0.10 | 0.10912 |

**Table 4. Postprocessing results of four English corpora**

| | Training-Improvement | | Test-Improvement | | $\alpha$ | $\tau$ |
|---|---|---|---|---|---|---|
| Neurology (G) | | | | | | |
| OCR-Engine$_1$ | 96.78%→97.52% | +0.74 | 97.78%→97.84% | +0.06 | 0.20 | 0.19783 |
| OCR-Engine$_2$ | 96.39%→96.72% | +0.33 | 96.94%→97.18% | +0.24 | 0.10 | 0.10854 |
| Holocaust (G) | | | | | | |
| OCR-Engine$_1$ | 98.07%→98.60% | +0.53 | 97.99%→98.22% | +0.23 | 0.20 | 0.19739 |
| OCR-Engine$_2$ | 97.48%→98.24% | +0.76 | 97.29%→97.97% | +0.68 | 0.15 | 0.15190 |
| Rom. Emp. (G) | | | | | | |
| OCR-Engine$_1$ | 98.56%→98.70% | +0.14 | 98.71%→98.73% | +0.02 | 0.20 | 0.20009 |
| OCR-Engine$_2$ | 98.47%→98.51% | +0.04 | 98.21%→98.29% | +0.08 | 0.20 | 0.20467 |
| Mushrooms (G) | | | | | | |
| OCR-Engine$_1$ | 98.40%→98.85% | +0.45 | 97.52%→97.97% | +0.45 | 0.10 | 0.10542 |
| OCR-Engine$_2$ | 98.02%→98.63% | +0.61 | 96.35%→96.94% | +0.59 | 0.10 | 0.10340 |

**Table 5. Postprocessing results of four German corpora.**

| | Training-Improvement | | Test-Improvement | | $\alpha$ | $\tau$ |
|---|---|---|---|---|---|---|
| European laws (B) | | | | | | |
| OCR-Engine$_1$ | 85.67%→94.65% | +8.98 | 83.70%→92.95% | +9.25 | 0.05 | 0.94014 |
| OCR-Engine$_2$ | 52.08%→91.14% | +39.06 | 51.20%→90.32% | +39.12 | 0.05 | 0.90300 |

**Table 6. Postprocessing results of Bulgarian corpus.**