Multimed Tools Appl (2017) 76:24549–24567
DOI 10.1007/s11042-016-4185-5

Learning string distance with smoothing for OCR spelling correction

Daniel Hládek[1] · Ján Staš[1] · Stanislav Ondáš[1] · Jozef Juhár[1] · László Kovács[2]

Received: 30 June 2016 / Revised: 11 November 2016 / Accepted: 18 November 2016 /
Published online: 7 December 2016

Abstract Large databases of scanned documents (medical records, legal texts, historical documents) require natural language processing for retrieval and structured information extraction. Errors caused by the optical character recognition (OCR) system increase ambiguity of recognized text and decrease performance of natural language processing. The paper proposes OCR post correction system with parametrized string distance metric. The correction system learns specific error patterns from incorrect words and common sequences of correct words. A smoothing technique is proposed to assign non-zero probability to edit operations not present in the training corpus. Spelling correction accuracy is measured on database of OCR legal documents in English language. Language model and learning string metric with smoothing improves Viterbi-based search for the best sequence of corrections and increases performance of the spelling correction system.

Keywords OCR · Spelling correction · Learning string distance · Hidden Markov model

✉ Daniel Hládek
daniel.hladek@tuke.sk
http://nlp.web.tuke.sk

Ján Staš
jan.stas@tuke.sk

Stanislav Ondáš
stanislav.ondas@tuke.sk

Jozef Juhár
jozef.juhar@tuke.sk

Lászlo Kovács
kovacs@iit.uni-miskolc.hu

[1] Department of Electronics and Multimedia Communications, Technical University of Košice, Letná 9, 042 00, Košice, Slovakia

[2] Institute of Information Technology, University of Miskolc, H-3515 Miskolc-Egyetemváros, Hungary

1 Introduction

Optical character recognition (OCR) is the process of converting a digitized image into text. Its main areas of application are automatic processing of hand-written business documents entries and forms, converting text from hardcopy, such as books or documents, into electronic form, and multimedia database searching for letter sequences, such as license plates in security systems.

OCR assists multimedia database processing – multimedia information retrieval and extraction from video or static digitized images. Additional information about improving its precision is presented in paper [12]. Examples of application for OCR in multimedia databases are:

- spelling correction system in scientific databases [19];
- recognition of characters in Chinese calligraphy [20];
- text extraction from video databases [36];
- preservation and processing of cultural heritage in digital humanities [28, 34];
- management of medical texts [13].

Natural language processing of digitized texts requires error correction for improved searching in digitized document databases, document indexing, document sorting into categories, and business data acquisition. Correction of OCR spelling errors improves the precision of information retrieval [8].

Spelling correction recovers the original form of the word in consideration of surrounding words and typographical errors in the word without taking the original image of the page into account. OCR errors create error patterns characteristic to the recognition algorithm, print format, paper, font, and language of the document. Any error correction method must be adapted to individual cases.

This paper proposes a correction system automatically adjustable for OCR errors. The learning algorithm adapts a parametrized string metric to specific error types manually corrected in the past. The language model improves suggestions for correction candidates taking context of the word into account. The Viterbi approach uses dynamic programming to find the best matching sequence of correct words according to the available problem information.

The algorithm operates if it is not possible to prepare each statistical component. The correction system functions if it is not possible to train a learning string metric or to work only with the language model without losing much precision.

Section 2 gives a brief overview of the current literature on the task after the problem statement in the introductory section. Common components for better disambiguation consistent with context are language models or more advanced machine-learning techniques, such as a maximum entropy classifier or hidden Markov model. Custom string distance is present in some cases.

Components of the proposed system are described in Section 3. The presented approach contains an error detection and spelling suggestion component, a language model, a learning string metric with parameter smoothing, and a Viterbi algorithm for the best correct word sequence search. Each component is described in its own subsection.

Section 4 describes the experimental evaluation. The proposed system is evaluated on aligned sequences from a database of OCR scanned images in the TREC-5 Confusion Track [10]. String distance with calculated parameters is used as observation probability for the hidden Markov model of the spelling correction system. The language model of a training set is used as state-transition probability. The best sequence of correction is found using the Viterbi algorithm.

Section 5 gives summary of the paper.

2 State of the art of spelling correction for OCR

Spelling correction has a long history of research. Theoretical foundations of spelling correction were presented in [11]. Spelling error types can be divided according to vocabulary into:

– non-word errors, where tokens are a sequence of characters not present in vocabulary;
– real-word errors, where the error is a valid word, but not the one that was meant.

A dictionary of valid words or a formal grammar of language morphology can detect non-word errors. However, real-word errors can be detected with deeper context analysis or word sense disambiguation.

Spelling correction is identifying incorrect words and sorting the set of correction candidates. The best candidate for correction is selected interactively or non-interactively.

Common use of spelling correction is in interactive mode, where the system identifies possibly incorrect parts of text and the user selects the best matching correction. The paper [34] describes an interactive post-correction system for historical OCR texts. Authors in [14] propose a crowd-sourcing, web based interface for spelling suggestions. The paper [28] evaluates the precision of common OCR for recognizing historical documents.

This paper is focused on non-interactive spelling correction of text produced by an OCR system, where the best correction candidates are selected automatically, taking context and string distance of the correction candidate into account. A non-interactive spelling correction system can be part of a multimedia database, a security system, or an information retrieval system.

One of the most recent contributions in the field of OCR spelling is correction of Chinese medical records in [13]. A morphological analyzer and named entity recognizer are important parts of this system because words in Chinese are not separated by spaces. Presence of a word in a medical dictionary and n-gram language model is used as a feature for the Maximum Entropy classifier to determine the best possible correction.

The method [8] improves information retrieval from OCR documents in Indian languages with a data-driven (unsupervised) approach. This paper proposes a novel method for identifying erroneous variants of words in the absence of a clean text. It uses string distance and contextual information to multiple variants of the same word in text distorted by OCR. Identical terms are identified by classical string distance with fixed parameters and term co-occurrence in documents.

The paper [6] applies an n-gram language model, trie, and A* search for correction candidate proposal, word similarity weights, and manual sentence alignment. The method estimates the probability of an edit operation from a training corpus. A* search provides a weighted list of correction candidates. The language model reorders correction candidates to find the best correction. The paper concludes that information about context has a bigger impact on classification precision than the weight of edit operation.

A method in [23] discovers the typical errors of an OCR system in scanned historical German texts and provides list of possible original and modern word forms. It uses approximate string matching from a lexicon and the Bayes rule to identify possible corrections.

OCR systems for printed Tamil texts are compared in [22] and a post-processing error correction technique for the tested OCR system is proposed.

The approach in [15] uses a language model and custom string metric. An old style of writing is seen as a distorted current form. A machine translation system, language model,

and string distance is used to transform a 17th century English historical text into current language.

The paper [26] improves segmentation of paragraphs and words by correcting the skew angle of a document. The method in [30] uses a combination of the longest common subsequences (LCS) and Bayesian estimates for a string metric to automatically pick the correction candidate from Google search. The method in [5] uses a bigram language model and WordNet for better spelling correction. The paper [31] proposes a hidden Markov model for identification and correction. The approach [7] uses morphological analysis of suffixes for candidate word proposal. The paper [21] incorporates a language model of errors into the weighted finite state transducer of OCR.

2.1 String distance in spelling correction

It is necessary to determine the similarity of two strings to solve the problem of spelling correction. A widely-used concept of string similarity is the edit distance: the minimum number of insertions, deletions, and substitutions required to transform the string into the other string [24]. According to edit operations, spelling errors are divided into:

- substitution with one letter changed into another;
- deletion with one letter missing;
- insertion with one letter extra.

Edit operation z is an ordered pair of symbols from source and target alphabet or the termination symbol #,#.

$$E \in (A \times B) \cup \{\#, \#\}, \tag{1}$$

A zero string ϵ in edit operation z means insertion of a character from the target alphabet B or deletion of a character from the source alphabet A.

The minimum number of these edit operations necessary to transform a string is called the Levenshtein string distance. Each edit operation has a value of 1 and the sum of edit operations is the Levenshtein edit distance.

The distance between between two strings x^T with length T and y^V with length V $d_c(x^T, y^V)$, is defined recursively as [24]:

$$d_c(x^t, y^v) = \min \left\{ \begin{array}{l} \delta(x^t, y^v) + d_c(x^{t-1}, y^{v-1}) \\ \delta(x^t, \epsilon) + d_c(x^{t-1}, y^v) \\ \delta(\epsilon, y^v) + d_c(x^t, y^{v-1}) \end{array} \right\}, \tag{2}$$

x^t is the prefix of string x^T with length t, y^v is the prefix of string y^V with length v. δ are parameters of the string metric. If values δ of edit operations are just zero or one, the Levenshtein edit distance can be calculated using dynamic programming algorithm [35].

Weights δ in a letter similarity matrix express the probability of error types. [1] defines the error model of spelling correction, where parameters of the model are weights of edit operations. Parameters of the model can take both theoretical expectations and experimental observation of human behavior. Real values of the string distance parameters, where each edit operation can have different weight δ, are a generalization of the Levenshtein distance.

3 The proposed system

Each spelling correction system (and natural language processing system) is based established on human knowledge. Performance and quality of the system depends on the quality

and methods of its processing. Some systems have decision rules encoded directly into the source codes, others can learn from human-annotated corpora, transform examples into parameters of a statistical model and generalize a model for events not seen in the training data.

This knowledge can be divided according to its form into:

– implicit (latent, hidden) knowledge - based on examples in the form of annotated data;
– explicit knowledge - expressed in the form of rules, programs or equations.

Implicit knowledge, hidden in the corpus of manually annotated examples, requires statistical processing or machine learning techniques to estimate the parameters of the model. Explicit knowledge can be used directly by the processing system.

Our correction system uses both types of knowledge. The proposed system continues from our previous paper [9], where an HMM-based correction system was presented.

The spelling correction system consists of these components:

1. *Error detection and correction suggestion* describes how ”incorrect” words look and proposes correction candidates.
2. *A parametrized string metric* reorders correction candidates according to the error model.
3. *A language model* expresses ”correct” language and the common contexts of correct words.
4. *A disambiguator* uses evaluated correction candidates and their probability according to context to find the best sequence of correct words for the current sentence.

Components of the systems and respective knowledge sources are depicted in Fig. 1.

3.1 Error detection and correction suggestion

The problem of spelling correction distinguishes two types of words: correct and incorrect. The first part of spelling correction is identifying the incorrect word boundaries according to a dictionary of given language. Explicit rules are necessary to omit correct parts of the text not in the dictionary, but are considered correct, such as numbers or acronyms. The second part of this problem is creating a list of possible corrections. A finite state acceptor or Levenshtein automaton [27] searches the correction lexicon in the form of a suffix trie. Correction candidates form a list of correct words in a given Levenshtein distance from the incorrect word. The best correction candidate can be selected interactively (in text editors or office systems) or in a non-interactively (information retrieval or extraction).

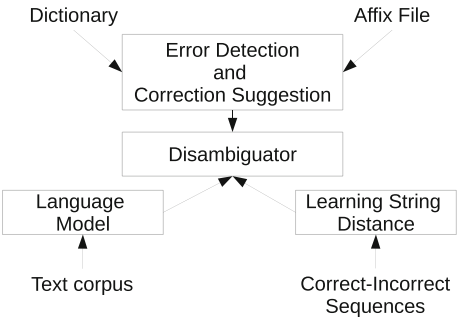


Fig. 1 Knowledge sources for the proposed system

The common spelling error detection and correction suggestion systems in use are Hunspell[1] and the older library Aspell.[2] Both libraries are used and evaluated in this paper. These error detection and error suggestion systems use explicit knowledge about correct words in the form of a correct word lexicon. Knowledge about incorrect words is encoded into the string metric. Correction candidates are sorted according to the Levenstein edit distance to the incorrect word.

3.2 Language model

The Levenshtein edit distance is, in most cases, a sufficient suggestion of correction candidates. The language model of the spelling correction system represents implicit knowledge about correct language. The context of the correction candidate helps distinguish better matching candidates by sorting them according to fitness with surrounding words.

The language model assigns the probability of occurrence of a word given a list of preceding words. It is estimated from a large set of training sequences. It is assumed that common word sequences are correct.

The n-gram model approximates the probability of a sequence of words y with length m:

$$P(y_1, y_2..y_m) = \prod_{i}^{m} P(y_i | y_{n-(i-1)...y_{i-1}}), \tag{3}$$

n means the size of the context - in the case of $n = 3$, the language model is called a trigram and takes the current word and two words from history, $n = 2$ is bigram model with only one word from history. A unigram model gives the probability of a word with no history. In the case of a spelling correction problem, the language model estimates the probability of a correction candidate according to a given history $P(x|y_{i-1}..y_{i-(n-1)})$.

Even if the training corpus for the language model is large, it does not contain enough valid sequences of language. Language model smoothing techniques are required to move part of the probability mass from events in the training corpus to events not present. Unseen events will have a non-zero probability. Common smoothing techniques for language models are summarized in [4].

The quality of a language model is expressed as perplexity. More on language modelling and how the perplexity of a language model depends on the theme of a text can be found in [29].

3.3 Learning string distance

The learning string distance is a generalization of the classic Levenshtein distance. The distance between two strings is calculated as the minimal sum of edit operations that transform the first string to the second string. The learning edit distance uses different weights for each possible edit operation – deletion, substitution, and insertion.

The weights of edit operations are stored in a letter similarity matrix. Each letter and zero length string ϵ has a row and column in the matrix. The weight of two letters express the probability of replacement, ϵ and a letter express the probability of insertion or deletion. The letter similarity matrix fully describes the learning string distance.

[1] https://hunspell.github.io

[2] http://aspell.net

Different parameters of the metric can be adjusted to specific problems. If an OCR system often replaces `i` and `l` or misses `f`, this feature can be expressed in the matrix. Parameters of the metric that express specific error patterns can be learned from a set of given examples of correct and distorted text.

The process of learning is a variant of an expectation-maximization algorithm. Firstly, the letter similarity matrix is assigned some initial values. A forward-backward algorithm is used to calculate the weight of each edit operation present in the training set with respect to the current distance parameters (letter similarity matrix). New parameters are the calculated weights of edit operations from the forward-backward algorithm. A smoothing step moves part of the probability mass to edit operations that were not observed in the training set.

The rest of this section describes in more detail how the string distance is calculated using a forward-backward algorithm and letter similarity matrix estimation using expectation maximization.

3.3.1 Distance calculation

The learning string edit distance is presented in papers [3, 32]. This method for estimating parameters of the string distance from a corpus of examples was first presented in [24]. Parameters of the string metric are seen as parameters of a memoryless stochastic transducer.

> Weighted transducers are automata in which each transition in addition to its usual input label is augmented with an output label from a possibly new alphabet, and carries some weight element of a semiring. Transducers are used to define mapping between two different types of information sources, e.g., word and phoneme sequences [16].

The distance between two strings is the negative log value of transduction probability from a target to a source string:

$$d_\phi = -\log p(x^T, y^V|\phi), \tag{4}$$

x^T is a source string of length T, y^V is a target string of length V. $\phi(A, B, \delta)$ is a memoryless stochastic transducer with parameters δ, where A is the source alphabet, B is the target alphabet. Zero length string ϵ is a part of both source and target alphabets.

The letter similarity matrix δ gives the probability of an edit operation z from a list of possible edit operations Z. Matrix δ consists of positive values less than or equal to one and their sum is one:

$$\sum_{z \in Z} \delta(z) = 1. \tag{5}$$

The probability of transduction $p(x^T, y^V|\phi)$ can be calculated as a forward probability $\alpha_{T,V}$:

$$p(x^T, y^V|\phi) = \alpha_{T,V}. \tag{6}$$

The forward probability matrix $\alpha_{t,v}$ for each prefix x^t, y^v of lengths t and v, $t \in \{0...T\}$, $v \in \{0...V\}$ of source and target strings x^T, y^V by a sequence of calculations is calculated using the forward algorithm [24]:

$$\begin{array}{lll} \alpha_{t,v} = & 1 & \\ \alpha_{t,v} = & 0 & if(v > 1 \vee t > 1) \\ \alpha_{t,v} += & \delta(\epsilon, y_v)\alpha_{t,v-1} & if(v > 1) \\ \alpha_{t,v} += & \delta(x_t, \epsilon)\alpha_{t-1,v} & if(t > 1) \\ \alpha_{t,v} += & \delta(x_t, y_v)\alpha_{t-1,v-1} & if(v > 1 \wedge t > 1) \end{array} \tag{7}$$

The last character of each input sequence (incorrect word and correction candidate) is a termination symbol #. The resulting transduction probability $p(x^T, y^V|\phi)$ is normalized with the inverse of the training sequence count $\delta(\#, \#)$.

$$p(x^T, y^V|\phi) = \alpha_{T,V}\delta(\#, \#) \tag{8}$$

3.3.2 Parameter estimation

Parameters δ are estimated using forward-backward algorithm that is a variant of expectation-maximization approach.

The backward probability $\beta_{t,v}$ contains the probability $p(x_{t+1}^T, y_{v+1}^V|\phi)$ of generating terminated suffix pair x_{t+1}^T, y_{v+1}^V [24]. t, v start from T, V to zero.

$$\begin{array}{lll} \beta_{t,v} = & \delta(\#, \#) & \\ \beta_{t,v} = & 0 & if(v > V \vee t > T) \\ \beta_{t,v} += & \delta(\epsilon, y_{v+1})\beta_{t,v+1} & if(v > V) \\ \beta_{t,v} += & \delta(x_{t+1}, \epsilon)\beta_{t+1,v} & if(t > T) \\ \beta_{t,v} += & \delta(x_{t+1}, y_{v+1})\beta_{t+1,v+1} & if(v > V \wedge t > T) \end{array} \tag{9}$$

The results of forward and backward algorithms are matrices α, β of dimension $T + 1, V + 1$ with forward and backward probabilities of transduction for each pair of string prefixes x^T, y^V.

Future transducer parameters γ are calculated from matrices α, β for each training sentence with learning parameter λ in the expectation step of the learning algorithm.

The following update of γ is performed for each pair of training sequences in the training corpus.

Parameters γ are updated with the calculated α and β for each edit operation corresponding to a pair of prefixes x^t, y^v in the training sample:

$$\begin{array}{lll} \gamma_{x_t,\epsilon} += & \alpha_{t-1,v}\delta(x_t, \epsilon)\beta_{t,v}/\alpha_{T,V} & if(v > 1) \\ \gamma_{\epsilon,y_v} += & \alpha_{t,v-1}\delta(\epsilon, y_v)\beta_{t,v}/\alpha_{T,V} & if(t > 1) \\ \gamma_{x_t,y_v} += & \alpha_{t-1,v-1}\delta(x_t, y_v)\beta_{t,v}/\alpha_{T,V} & if(v > 1 \wedge t > 1) \end{array} \tag{10}$$

The final step is normalization of γ to fulfill constraint in the (5). The value for each edit operation $\gamma(z)$ is divided by the sum of γ. Calculated γ is set as a new set of parameters δ in the maximization step of the learning algorithm.

$$\delta(z) = \frac{\gamma(z)}{\sum_{e \in Z} \gamma(e)} \tag{11}$$

Learning continues for a fixed number of steps or until the difference between γ and δ is low. The result of the training are parameters δ of the stochastic transducer and parameters of the string distance metric.

3.4 Smoothing of LSM

Learning of the probabilistic string metric suffers from over-training [2]. The training database is always sparse - possible edit operations are not present in the training set. This kind of training assigns zero probability to unseen events, even if letter transduction is probable. If the number of edit operations in the training corpus is low, matrix γ of learned metric parameters is sparse. Some corrections will have infinite distance if it is used as a metric for spelling correction problem.

Part of the probability mass δ is moved from seen events (non-zero elements of the matrix) to unseen events in the training corpus in the process of smoothing. If unseen edit operations have non-zero probability, the distance of correction candidates cannot be infinite and may be included in the classification.

A linear interpolation with a matrix of constants and an interpolation parameter lambda is proposed for improving the learned parameters of string metric.

According to the maximum entropy principle, a general stochastic transducer that does not take any training data into account is a transducer with uniform distribution. Parameters of transducer with uniform distribution is a constant matrix δ_c. Its values are estimated as inverse square of each letter's transduction count C_l in the training corpus:

$$\delta_c(z) = \frac{1}{C_l^2}. \tag{12}$$

The learned transducer is interpolated with a transducer with uniform distribution to consider operations not in the training set (assign them non-zero probability). Smoothed parameters δ_s are equal to:

$$\delta_s(z) = \lambda\delta_c(z) + (1 - \lambda)\delta_l(z). \tag{13}$$

The linear interpolation parameter λ can be interpreted as the amount of purely random stochastic transducer behaviour.

The complete learning algorithm can be summarized as:

```
set delta = zero matrix
while Converegence:
    set gamma = zero matrix
    for each training sequence pair:
        calculate alpha using delta
        calculate beta using delta
        update gamma using alpha and beta
    normalize gamma
    smooth gamma
    set delta = gamma
```

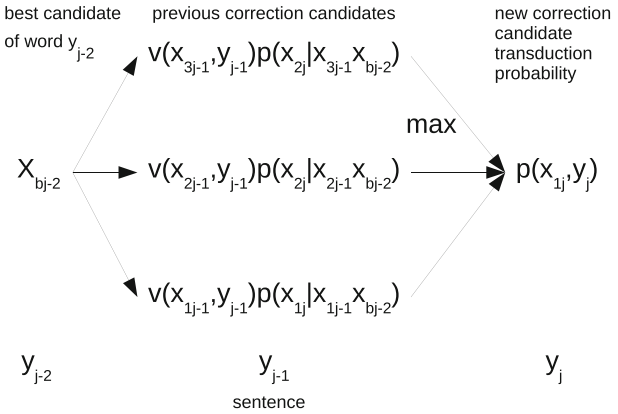


Fig. 2 Example of Viterbi value calculation from three previous correction candidates

3.5 Viterbi search

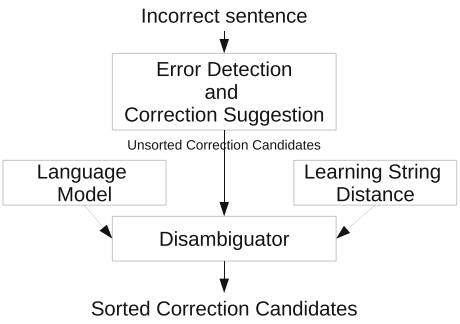
All these components compose a second order hidden Markov model [31], where state transition probability is the language model component and observation probability is the parametrized string metric.

The best sequence of output states (corrections or correct words) to the given hidden Markov model can be found using the Viterbi algorithm. Input of the algorithm is a sentence containing some incorrectly spelled words. Output is a sequence of best corrections for the given sentence.

The first part of the algorithm starts with the first word. Each possible correction is evaluated by a Viterbi value. The following word and its correction candidates are evaluated next. This procedure also determines the best previous state for each proposed correction candidate.

The best sequence of corrections is determined from evaluated words and corrections by a procedure called backtracking. It selects the correction with the best Viterbi value for the last word in the sentence. The best and last correction then determines the sequence of preceding corrections. Each correction candidate has its best predecessor calculated in the previous (forward) step.

Fig. 3 The proposed system



```
Federal Register

/ Vol. 59, No. 2 / Tuesday, January 4, 1994 / Rules and Regulations

Vol. 59, No. 2


Tuesday, January 4, 1994
-----------------------------------------------------------------------
Federal hegister ) Vol. 391 No. 2 )
Tuesday1 'anuary 41 1994 )           hules and

hegulations
Vol. 391 No. 2
Tuesday1 'anuary 41 1994
```

Fig. 4 Sample original and distorted documents

The Viterbi value is calculated recursively from previous words. The best product of state-transition probability and the previous Viterbi value is used to determine the next Viterbi value.

Each i-th correction candidate x_{ij} from a set of correction candidates $X(y_j)$ for the possibly j-th incorrect word y_j is evaluated by the value $v(x_{ij}, y_j)$ that is calculated recursively from v of the last word's correction candidates. The value v is the maximum of the product of transition probability $p(x_{ij}|x_{k(j-1)})$ and the previous Viterbi value $v(x_{k(j-1)}, y_{j-1})$ weighted by the observation probability $p(x_{ij}, y_j|\phi)$.

$$v(x_{ij}, y_j) = p(x_{ij}, y_j|\phi) \max_{x_{k(j-1)} \in X(y_{j-1})} p(x_{ij}|x_{k(j-1)}) v(x_{k(j-1)}, y_{j-1}) \tag{14}$$

Correction candidates $X(y_j)$ are determined by a error detection system (Hunspell or Aspell). The set of correction candidates $X(y_j)$ has a preliminary order defined by the system. The transition probability $p(x_{ij}|x_{k(j-1)})$ of two succeeding correct words is given by the language model. The observation probability $p(x_{ij}, x_j|\phi)$ is the probability of transduction from the correction candidate x_{ij} to an incorrect word y_j, when transducer ϕ has parameters δ.

An example of calculating the Viterbi value $v(x_{32}, y_3)$ of correction candidate x_{32} for words y_1, y_2, y_3 is depicted in Fig. 2. A block scheme of the whole correction system is depicted in Fig. 3.

In the case of a second-order HMM, a trigram language model can be used. The transition probability $p(x_{ij}|x_{k(j-1)}, x_{b(j-2)})$ now depends on correction $x_{b(j-2)}$ of the word y_{j-2} with the best Viterbi value that is constant for each previous correction candidate for word y_{j-1}.

$$b = \arg \max_{x_{k(j-2)} \in X(y_{j-1})} v(x_{k(j-2)}, y_{j-2}) \tag{15}$$

Using this technique the second-order HMM has the same computational complexity as the first order HMM.

```
Federal|Federal hegister|Register )_Vol.|/Vol. 391|59, No.|No.
2_)_Tuesday1|2/Tuesday, 'anuary|January 41|4, 1994_)_hules|1994/Rules and_|and
hegulations|Regulations Vol.|Vol. 391|59, No.|No. 2|2 Tuesday1|Tuesday,
'anuary|January 41|4, 1994|1994
```

Fig. 5 Sample aligned document

Table 1 Evaluation Corpus Characteristics

	Test Set	Train Set
Tokens	297 230	2 661 804
Sentences	22 920	206 275
Size (B)	3 491 924	31 346 181

4 Experiments

Data from the TREC-5 Confusion Track [10] were selected to evaluate the proposed approach. Other evaluation corpora are described in [18]. The TREC-5 Confusion corpus contains 55,600 legal documents from U.S. Federal Register, original electronic documents, and two sets degraded by an OCR system. This set is freely available and has already been used to evaluate OCR spelling correction system (e.g. in [8, 25]). The database contains original text document and text output degraded by an OCR system. Authors of the database did not make scanned images a part of the evaluation set, because the OCR process is not a part of the evaluation task.

These degraded documents were printed and the scanned from paper. The first run of OCR was performed on images of documents in high resolution with a character error rate (ratio of incorrect characters to all characters) of approximately 5 %. This set is marked *Deg5* in experiments. The second run of OCR was performed on documents with low resolution and has a character error rate of approximately 20 %. This set is marked *Deg20*. Example of the original and distorted document is in Fig. 4.

4.1 Evaluation methodology

Performance of automatic OCR is word error rate, defined as the ratio of incorrect words to all words.

$$\mathrm{WER} = \frac{\text{incorrect words}}{\text{all words}}.100\ \% \tag{16}$$

Table 2 Effect of smoothing on WER

Interpolation Parameter λ	Deg5 WER [%]	Deg20 WER [%]
0	18.13	44.31
0.02	17.44	43.59
0.05	17.32	43.56
0.07	17.32	43.57
0.12	17.31	43.62
0.1	17.31	43.62
0.15	17.31	43.64
0.2	17.37	43.70

Table 3 Effect of Commponents of the Spelling Correction on WER

Setup	Deg5 WER [%]	Deg20 WER [%]
None	20.74	52.31
Aspell	21.44	51.80
Aspell+LM	17.02	42.18
Aspell+LD	17.64	44.00
Aspell+LD+LM	16.61	40.70
Hunspell	21.35	51.96
Hunspell+LM	17.37	41.98
Hunspell+LD	17.93	44.24
Hunspell+LD+LM	17.05	41.05

4.2 Experimental data preparation

Original and distorted versions of the document are aligned using the Needleman-Wunsch dynamic programming algorithm [17]. The result is pairs of incorrect and correct words. Figure 5 depicts aligned sequences of correct and incorrect words. Token boundaries are identified according to spaces.

Preliminary alignment is required, because using documents as training sequences is not computationally feasible. The size of matrices α and β calculated for each training sample depends on the size of the correct and incorrect part. Token alignment reduces computational complexity. Incorrect-correct pairs are training samples for the learning string metric. It is possible to train the system using unaligned documents, but it is computationally complex. The complexity of training is dependent on the length of input strings (correct and incorrect).

The training and evaluation set is constructed from aligned samples. The size of training and testing sets is summarized in Table 1.

The parameters of the learning string metric and the language model are estimated from the training set. String distance is trained using the forward-backward algorithm described in Section 3 in 5 iterations. The trigram language model is estimated using the SRILM Toolkit.[3] Witten-Bell smoothing method is used for unigrams, bigrams and trigrams. Experiments were performed using GNU Parallel script [33].

4.3 Effect of smoothing on the learning distance metric

The effect of smoothing on the learning string distance was examined in the first experiment. The learning string distance was learned with values of the interpolation parameter λ from (13) in Section 3.4. The correction system was run without a language model with Aspell detection and correction suggestion. Results of the experiment are in Table 2.

The best performance for the *Deg5* set was reached with value $\lambda = 0.12$. Experiment with $\lambda = 0$ demonstrates performance of the system with a learning string distance without smoothing. The results in Table 2 show that smoothing of the learning string metric has a positive impact on performance of spelling correction using a learning string metric.

[3] http://www.speech.sri.com/projects/srilm/.

4.4 Effect of individual components on performance

The effect on performance of system components is evaluated in the second experiment. The spelling correction system was run in different configurations:

- *None*: CC. Shows word error rate of the document without any correction.
- *Hunspell or Aspell*: Hunspell error detection and spelling suggestion is used. The first proposed correction candidate is selected as the best.
- *Hunspell or Aspell + LD*: Each suggested correction candidate is evaluated by a learning sting distance with smoothing. The candidate with the best string distance is selected as a candidate for correction. The first proposed correction candidate is the best.
- *Hunspell or Aspell + LM*: The best sequence of corrections is found with a Viterbi search. Observation probability of the correction candidate is determined only by its transition probability given by the language model.
- *Hunspell or Aspell + LD + LM*: A full Viterbi-based search is performed in this configuration as it is described in Section 3.5. The probability of transduction of an erroneous word and the correction candidate is used as an observation probability, and a language model is used as a state-transition probability.

Experiments in Table 3 show that based on explicit rules, the classic spelling correction systems Aspell and Hunspell cannot be used for the task of OCR correction. However, the proposed correction candidates are feasible for classification according to context, edit distance, or both.

The context of a word impacts strongly on the word error rate of OCR spelling correction. The effect of a language model is comparable to the effect of the learning string distance with smoothing. If they are used together, they bring even more improvement of WER. A stronger, positive effect of the language model compared with the parametrized string distance is consistent with findings in [6].

5 Conclusion

The approach presented in this paper uses state-of-the art spelling correction (Hunspell, Aspell) and can handle additional knowledge sources. Along with the learning string distance, the proposed correction system can use a language model and correction lexicon, if they are available. Experiments show that learning string distance and language model improve the spelling correction precision.

5.1 Contribution summary

The novelty of the approach is designing and evaluating the system performance with both language model and learning string distance (*spell[4]+LM+LD in experiment 2). Two independent knowledge sources are incorporated into a hidden Markov model as observation and transition probabilities. Their impact on correction system is measured in Table 3. Approaches where only a language model is used (*spell+LM) for better correction

[4]Hunspell or Aspell.

candidate disambiguation have been presented in previous papers [6, 21, 33] (and others) and evaluated in terms of other problems.

The other innovation of this paper is proposing and evaluating the smoothing technique for learning string distance. Its single parameter is described as an amount of random stochastic transducer behavior and can be easily estimated. It is shown that this kind of smoothing has a positive effect on accuracy.

5.2 Discussion of experiments

Performance of Hunspell and Aspell in the problem of automatic correction is measured in experiments (*spell). Experiments show that these correction systems based on a lexicon and Levenshtein string distance do not have satisfactory performance. Their negative impact on word error rate is caused by false positives in error detection, where correct parts of text are falsely marked as incorrect and changed. These values can be considered as a baseline for comparison.

It is interesting that the effects of context (*spell+LM) are comparable to the effect of a sole error model (learning string distance, *spell+LD). These two implicit knowledge sources are different and uncorrelated. It is possible that using another method of disambiguation, such as conditional random fields or maximum entropy, can produce better performance than the presented Viterbi algorithm (*spell+LM+LD).

Acknowledgements The work presented in this paper was supported by the Ministry of Education, Science, Research and Sport of the Slovak Republic under research project VEGA 1/0075/15 and by the Slovak Research and Development Agency under research project APVV-15-0517 and by the Technical University of Košice under research project DIALab.

References

1. Ahmad F, Kondrak G (2005) Learning a spelling error model from search query logs. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05, pp. 955–962. Association for Computational Linguistics, Stroudsburg, PA, USA. doi:10.3115/1220575.1220695
2. Bellet A, Habrard A, Sebban M (2012) Good edit similarity learning by loss minimization. Mach Learn 89(1-2):5–35. doi:10.1007/s10994-012-5293-8
3. Bilenko M, Mooney R (2003) Adaptive duplicate detection using learnable string similarity measures:39–48. doi:10.1145/956750.956759
4. Chen SF, Goodman J (1999) An empirical study of smoothing techniques for language modeling. Comput Speech Lang 13(4):359–394. doi:10.1006/csla.1999.0128
5. Eutamene A, Kholladi M, Belhadef H (2015) Ontologies and bigram-based approach for isolated non-word errors correction in OCR system. International Journal of Electrical and Computer Engineering 5(6):1458–1467
6. Evershed J, Fitch K (2014) Correcting Noisy OCR: Context Beats Confusion. In: Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, DATeCH '14, pp. 45–51. ACM, New York, NY, USA. doi:10.1145/2595188.2595200
7. Gerdjikov S, Mihov S, Nenchev V (2013) Extraction of spelling variations from language structure for noisy text correction. doi:10.1109/ICDAR.2013.72

8. Ghosh K, Chakraborty A, Parui SK, Majumder P (2016) Improving Information Retrieval Performance on OCRed Text in the Absence of Clean Text Ground Truth. Inf Process Manag. doi:10.1016/j.ipm.2016.03.006. Artice in press
9. Hládek D., Staš J., Juhár J. (2013) Unsupervised spelling correction for Slovak. Adv Electr Electron Eng Ser 11(5):392–397. doi:10.15598/aeee.v11i5.898
10. Kantor P, Voorhees E (2000) The TREC-5 confusion track: Comparing retrieval methods for scanned text. Inf Retr 2(2-3):165–176
11. Kukich K (1992) Techniques for automatically correcting words in text. ACM Comput Surv 24(4):377–439
12. Lin Y, Song Y, Li Y, Wang F, He K (2015) Multilingual corpus construction based on printed and handwritten character separation. Multimedia Tools and Applications:1–17. doi:10.1007/s11042-015-2995-5
13. Lv YY, Deng YI, Liu ML, Lu QY (2016) Automatic error checking and correction of electronic medical records. Frontiers in Artificial Intelligence and Applications 281:32–40. doi:10.3233/978-1-61499-619-4-32
14. Mühlberger G., Zelger J, Sagmeister D (2014) User-driven correction of OCR errors. Combing crowdsourcing and information retrieval technology:53–56. doi:10.1145/2595188.2595212
15. Mitankin P, Gerdjikov S, Mihov S (2014) An Approach to Unsupervised Historical Text Normalisation. In: Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, DATeCH '14, pp. 29–34. ACM, New York, NY, USA. doi:10.1145/2595188.2595191
16. Mohri M (2004) Weighted Finite-State Transducer Algorithms. An Overview. In: D.C. Martín-Vide, D.V. Mitrana, D.G. Păun (eds.) Formal Languages and Applications, no. 148 in Studies in Fuzziness and Soft Computing, pp. 551–563. Springer Berlin Heidelberg
17. Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48(3):443–453. doi:10.1016/0022-2836(70)90057-4
18. Oard DW, Baron JR, Hedin B, Lewis DD, Tomlinson S (2010) Evaluation of information retrieval for E-discovery. Artif Intell Law 18(4):347–386. doi:10.1007/s10506-010-9093-9
19. Park JH, Park HH, Kwon YB (2014) Error correction of reference indexing system including multimedia journals. Multimedia Tools and Applications 74(7):2359–2370. doi:10.1007/s11042-014-1971-9
20. Pengcheng G, Jiangqin W, Yuan L, Yang X, Tianjiao M (2014) Fast Chinese calligraphic character recognition with large-scale data. Multimedia Tools and Applications 74(17):7221–7238. doi:10.1007/s11042-014-1969-3
21. Perez-Cortes JC, Llobet R, Navarro-Cerdan J, Arlandis J (2010) Using field interdependence to improve correction performance in a transducer-based OCR post-processing system. doi:10.1109/ICFHR.2010.99
22. Ramanan M, Ramanan A, Charles E (2015) A performance comparison and post-processing error correction technique to OCRs for printed Tamil texts. doi:10.1109/ICIINFS.2014.7036474
23. Reffle U, Ringlstetter C (2013) Unsupervised profiling of OCRed historical documents. Pattern Recog 46(5):1346–1357. doi:10.1016/j.patcog.2012.10.002
24. Ristad E, Yianilos P (1998) Learning string-edit distance. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(5):522–532
25. Sariev A, Nenchev V, Gerdjikov S, Mitankin P, Ganchev H, Mihov S, Tinchev T (2014) Flexible Noisy Text Correction. In: 2014 11th IAPR International Workshop on Document Analysis Systems (DAS), pp. 31–35. doi:10.1109/DAS.2014.12
26. Sawant A, Chougule D (2015) Script independent text pre-processing and segmentation for OCR. doi:10.1109/EESCO.2015.7253643
27. Schulz KU, Mihov S (2002) Fast string correction with Levenshtein automata. Int J Doc Anal Recognit 5(1):67–85. doi:10.1007/s10032-002-0082-8
28. Springmann U, Najock D, Morgenroth H, Schmid H, Gotscharek A, Fink F (2014) OCR of Historical Printings of Latin Texts: Problems, Prospects, Progress. In: Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, DATeCH '14. ACM, New York, NY, USA, pp 71–75. doi:10.1145/2595188.2595205
29. Staš J, Juhár J, Hládek D (2014) Classification of heterogeneous text data for robust domain-specific language modeling. EURASIP Journal on Audio, Speech, and Music Processing 2014(1):14. doi:10.1186/1687-4722-2014-14
30. Taghva K, Agarwal S (2013) Utilizing web data in identification and correction of OCR errors. In: Proc. SPIE 9021, Document Recognition and Retrieval XXI, 902109. doi:10.1117/12.2042403
31. Taghva K, Poudel S, Malreddy S (2013) Post processing with first- and second-order hidden Markov models. doi:10.1117/12.2006500
32. Takasu A (2009) Bayesian similarity model estimation for approximate recognized text search. doi:10.1109/ICDAR.2009.193
33. Tange O (2011) Gnu parallel - the command-line power tool. ;login: The USENIX Magazine 36(1):42–47. doi:10.5281/zenodo.16303

34. Vobl T, Gotscharek A, Reffle U, Ringlstetter C, Schulz KU (2014) PoCoTo - an Open Source System for Efficient Interactive Postcorrection of OCRed Historical Texts. In: Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, DATeCH '14. ACM, New York, NY, USA, pp 57–61. doi:10.1145/2595188.2595197
35. Wagner RA, Fischer MJ (1974) The String-to-String Correction Problem. J ACM 21(1):168–173. doi:10.1145/321796.321811
36. Yang H, Quehl B, Sack H (2012) A framework for improved video text detection and recognition. Multimedia Tools and Applications 69(1):217–245. doi:10.1007/s11042-012-1250-6

Daniel Hládek was born in Košice, Slovakia in 1982. He graduated in 2008 at the Department of Cybernetics and Artificial Intelligence at the Technical University of Košice. He finished his PhD. focused on fuzzy logic and reinforcement learning with applications in robotics at the same department. He is currently working as Assistant Professor at the Department of Electronics and Multimedia Communications, Faculty of Electrical Engineering and Informatics, Technical University of Košice. His research is focused on natural language processing of Slovak language with focus on information retrieval systems and content analysis.

Ján Staš was born in Bardejov, Slovakia in 1984. In 2007 he graduated M.Sc. (Ing.) at the Department of Electronics and Multimedia Communications of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice. He received his Ph.D. degree at the same department in the field of Telecommunications in 2011. He is currently working as an Assistant Professor in the Laboratory of Speech and Mobile Technologies at the same department. He is a specialist in the field of computational linguistics, natural language processing and statistical language modeling.

Stanislav Ondáš was born in Prešov, Slovakia in 1981. In 2004 he graduated M.Sc. (Ing.) at the Department of Electronics and Multimedia Communications of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice. He received his Ph.D. degree at the same department in the field of Telecommunications in 2008. He is currently working as an Assistant Professor in the Laboratory of Speech and Mobile Technologies at the same department. He is a specialist in the field of human-machine interaction, dialogue modeling and management, natural language processing and semantic analysis.

Jozef Juhár was born in Poproc, Slovakia in 1956. He graduated from the Technical University of Košice in 1980. He received Ph.D. degree in Radioelectronics from Technical University of Košice in 1991, where he works as full Professor at the Department of Electronics and Multimedia Communications. His research interests include digital speech and audio processing, speech/speaker identification, speech synthesis, development in spoken dialogue and speech recognition systems in telecommunication networks. Prof. Juhár is a member of ISCA, AES and IEEE. He is a member of the editorial boards and reviewer of several international scientific journals.

László Kovács (1961) studied mathematics and physics at the University of Kossuth Lajos, Debrecen, Hungary. He obtained a PhD degree in technical sciences from University of Miskolc in 1998. Dr Kovács is an assistant professor at the Department of Information Technology of the University of Miskolc. He actively participates in the teaching and research activities of the department. He held or holds courses in the following subjects: Software Development, Programming Foundations, Database Systems, WEB-Technologies, OLAP and Data Mining, XMLdata management. He participated in several mobility programs on the field of software engineering and database management. The research interest of Dr. Kovács involves the following areas: soft computing, heuristic optimizations, ontology modeling in database systems and software engineering. He is a project leader of a Department-level research group on ontology-based reverse engineering at University of Miskolc.