Croatian OCR Error Correction Using Character Confusions and Language Modelling

Mladen Marović, Mladen Mikša, Jan Šnajder, Bojana Dalbelo Bašić
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
{mladen.marovic, mladen.miksa, jan.snajder, bojana.dalbelo}@fer.hr

Abstract. *Manual correction of errors produced by optical character recognition (OCR) is a time-consuming task. This paper presents an automatic post-processing system that utilizes various methods for improving the OCR results of Croatian language texts. The system relies on knowledge of general characteristics of OCR errors, as well as language-specific knowledge. Used methods include character confusions, a character n-gram model, and word-splitting. A statistical language model is used for ranking the generated candidates depending on the sentential context. Experimental evaluation, performed on newspaper texts supplied by the Croatian News Agency, shows an error rate reduction of above 20%. These results amount to about 36% of the performance of manual correction.*

Keywords. Natural language processing, OCR, character confusions, character n-grams, word merge errors, language model, Croatian language

1 Introduction

With the advent of the information age a need has arisen for large-scale digitalization of vast quantities of data, with the dominant type being that in written form. A simple solution for converting written data into its appropriate electronic form is manual input, but – considering the volume of data that still exists primarily (or only) in print – this approach can be discarded as highly inadequate. A much more effective solution is to apply OCR (Optical Character Recognition), which is a process that translates images of handwritten, typewritten, or printed text into machine-encoded text. Unfortunately, such a process is not error-proof. While most errors appearing in an OCR-ed text can be easily corrected by a human reader, in automated text processing and IR (information retrieval) tasks the presence of such errors leads to a decrease in performance and the deterioration of results.

A substantial improvement of OCR results can be achieved by manual correction. However, for anything but a small quantity of text, manual correction is expensive and time consuming [10]. Another potential solution is correction via existing spelling checkers, such as the ones included in various word processors. Upon closer inspection, this approach can be deemed as unsatisfactory due to its semi-automaticity and the differences between typing errors and OCR errors. Therefore, a specific solution is needed for automatic correction of OCR errors.

Depending on the language of the corrected texts, some linguistic properties can also present either a difficulty or a useful source of information in correcting OCR errors. A high degree of inflection, such as in Croatian or Arabic languages, as well as the absence of specific word boundaries, such as in Chinese language, can present an additional challenge during OCR correction. Syntactic parsing techniques using regular or context-free grammars can increase the efficiency of the correction process, as was shown in [2].

In this paper, we describe and evaluate a post-processing system that utilizes various methods for improving OCR results of Croatian texts. The system is based on character confusion and character

n-gram models used to generate correction candidates, a combinatorial search method for splitting merged words, and a statistical language model for context-dependent candidate selection.

The rest of the paper is structured as follows. A brief overview of related work is given in the next section. Section 3 describes the OCR error correction system. Evaluation results are presented in Section 4, while Section 5 concludes the paper and outlines future work.

2 Related work

Research in correcting OCR errors has produced a set of diverse methods for tackling the problem. Many different ways of detecting and correcting erroneous words were developed, usually integrating some form of a dictionary and the information concerning the OCR process. Here we present some of the correction methods and the systems implementing them.

Tong and Evans [10] built a system based on retrieving correction candidates by character n-grams. The observed word is divided into n-grams and a list of candidate words that contain at least one of its n-grams is retrieved. Candidates are ranked by the likelihood of character confusions that could have resulted with the given string. A statistical language model is used to include context information and provide more accurate final ranking of candidates. The system assumes that the words in the OCR document are separated by blanks and therefore does not correct word boundary errors. They reported a result of 60.1% error reduction on the tested documents.

Another method used in correcting OCR documents is based on the idea that different OCR devices produce different errors. Klein and Kopel [3] have exploited this idea in their system using outputs from two OCR devices. In cases when the outputs differ the system uses additional information from the dictionary, character confusion matrix, and local context to decide on the correct word. The system achieved a decrease in the error rate of the two devices from 3.7% and 5.9% to 3.2% for English, from 22.1% and 2.1% to 1.5% for French, and from 14.1% and 11.0% to 3.2% for Hebrew.

OCRSpell, built by Taghva and Stofsky [9], is a complex system that exploits information gathered from multiple knowledge sources. It is semi-automatic in that it requires user interaction for some problematic cases. The system uses dynamic and static device mappings, n-gram analysis, and heuristics to improve its performance. It relies on heuristics to discern word boundaries, making correction of split and merged words possible. Evaluation of the system was performed on two OCR documents and it displayed an increase in word accuracy from 98.18% to 99.79% for one document, and from 98.46% to 99.85% for another document.

The work presented in this paper combines several described approaches, but with some modifications. We have modified character n-grams to include the information about its position in a string to make them more efficient. Word merge errors are handled by a combinatorial search.

3 System description

The system attempts to correct character confusion and word merge errors. Split word errors are neglected because of the small probability of their occurrence. The correction process starts by splitting the input text into distinct tokens representing possible words, punctuation marks, html tags, etc. Word tokens are processed further, whereas the rest is simply copied to the output. Word tokens are then checked against the dictionary; if the word is found in the dictionary, it is copied to the output, otherwise it is processed in three steps as follows. The first step generates correction candidates for the word token. The second step deals with merged words correction on candidates that do not exist in the dictionary, while the third step adjusts each candidate's probability via a language model. These three steps, depicted in Figure 1, are described next.

3.1 Character confusions

Candidate generation begins with a method based on character-level correction If character sequences known to be the result of an error are found in a word token, they are corrected in an attempt to produce valid word candidates. This type of correction is based on a noisy channel model, which relates an observable string O to an underlying se-

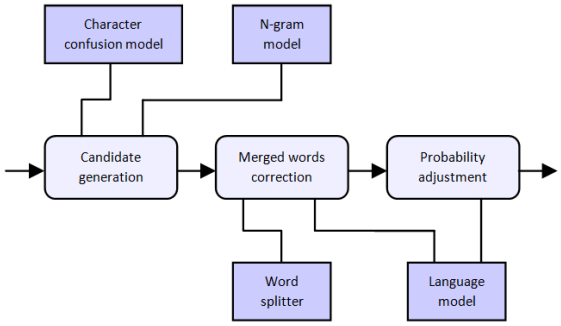

Figure 1: System workflow

quence, in this case recognized character strings to underlying word sequences. Such a model is used to learn how a chosen OCR device corrupts single characters or character sequences, producing a character-level confusion model specific to the device it was trained on.

The error model may be formalized as follows [6]. Let $W_v = \#V_1 \dots V_i \dots V_j \dots V_n\#$ represent a valid word, while $W_c = \#C_1 \dots C_k \dots C_l \dots C_m\#$ is a corrupt word produced by the OCR process, with # marking word boundaries. The $C_k \dots C_l$ is a character sequence misinterpreted from $V_i \dots V_j$. Let there also be three types of confusions: insertion and deletion of a character sequence, and substitution of a character sequence with a different one. Then, estimates for the probabilities of these confusions are given by the following formulas:

$$P_{sub}(V_i \dots V_j \to C_k \dots C_l) = \frac{c(V_i \dots V_j \to C_k \dots C_l)}{c(V_i \dots V_j)}, \quad (1)$$

$$P_{del}(V_i \dots V_j \to \varepsilon) = \frac{c(V_i \dots V_j \to \varepsilon)}{c(V_i \dots V_j)}, \quad (2)$$

$$P_{ins}(\varepsilon \to C_k \dots C_l) = \frac{c(\varepsilon \to C_k \dots C_l)}{c(\langle \text{all\_characters} \rangle)}, \quad (3)$$

with $c(x)$ being the number of observations of x in the training data set, and ε being the empty string. The probability of the entire word W_v being corrupted into W_c, with n confusions, is given by the formula:

$$P(W_c|W_v) = \prod_{m=1}^{n} P_{err}(V_{i_m} \dots V_{j_m} \to C_{k_m} \dots C_{l_m}). \quad (4)$$

The above described model assigns zero probabilities to unknown confusions. This can be avoided by smoothing these probabilities, i.e., by assigning a small uniform probability to unknown confusions, in accordance to Lidstone's law [6]. The uniform probability was chosen to be 100 times smaller than the probability of the rarest character confusion observed in the data set.

Our system uses a set of rules for correcting the aforementioned confusions based on the above described model. An excerpt from this set is given in Table 1. Rules are defined by a corrupt character sequence, $C_i \dots C_j$, and its valid replacement, $V_k \dots V_l$. The rule set is used to generate correction candidates from the word token. Because insertion rules can be applied to all positions in a word token, their use would cause a large increase in the number of generated candidates. Therefore, they are neglected in this step (note that insertion rules are produced from deletion confusions, so in fact the appearance of deletion confusions is neglected). The candidates generated using this method are then ranked by their respective probabilities and used in subsequent correction steps.

3.2 Character n-grams

Candidates generated via the character confusion model contain only corrections to known insertion or substitution errors. In principle, to deal with deletion errors and all unknown errors, the word token should be compared to each word in the dictionary to compute the conditional probability of that word being the correct candidate. However, the computational costs associated with such an exhaustive search are too high to be of any practical use [10]. Instead, similar to [10], a character n-gram model is used to retrieve words orthographically similar to the word token, presuming that these are the most likely correction candidates.

All words are indexed by their character n-grams at system startup. During word token processing, the token is split into character n-grams, which are then used to retrieve all the words in the dictionary

Table 1: Character confusion rules

Sequence		
Corrupt	Valid	Probability
v	*y*	0.4090
ii	*ü*	0.3750
u	*ü*	0.3125
tb	*TB*	0.2857
	⋮	
\	*v*	0.0001
J	*l*	0.0001
o	*p*	0.0001
	⋮	
š	*ε*	10^{-6}
šn	*ε*	10^{-6}
TM	*ε*	10^{-6}
un	*ε*	10^{-6}

containing at least one of these n-grams. An example of a word split into its character trigrams, with # acting as a word boundary marker, is given by:

$$nGrams(\#zamjena\#) = \{\#za, zam, amj, mje, jen, ena, na\#\}. \quad (5)$$

The method described so far greatly reduces the number of candidates retrieved from the dictionary. In order to further reduce this number, another problem with character n-grams should be avoided. This problem arises when two completely unrelated words have a number of n-grams in common. For example, words *vjerojatnost* (*probability*) and *postojanje* (*existence*) share the trigram *ost*, which would cause one word to be retrieved as a candidate for the other. Therefore, a modification to the existing character n-gram model is proposed by using each n-gram's relative position in a word as an additional information for candidate retrieval. Let W be a word of length n, and p the position of the observed n-gram G in the given word, with zero marking the position of the first n-gram. Then, a measure of the relative position, R, of the n-gram G in the word W can be expressed as:

$$R(G, W) = \frac{p}{n-1}, \quad (6)$$

with n being equal to the number of n-grams in W expanded with word boundary markers. For example, $R(amj, \#zamjena\#) = 2/6$. Based on this measure, for each n-gram, words are distributed into k classes of equal interval width (with k being 10 in our system). During candidate retrieval, only words in the same or the surrounding l classes are retrieved, with l being inversely proportional to the length of the word token.

Additionally, candidates are filtered using either of the two different measures: the number of common n-grams or minimum edit distance. Minimum edit distance is the minimum number of single character insertions, deletions, and substitutions needed to produce some new word from a given one [5]. Using either of these measures eliminates candidates least likely to be correct and thus further improves processing time. Retrieved candidates are then matched to the word token and ranked by their respective probabilities using the character confusion model described in 3.1.

3.3 Merged words correction

After the candidate generation step has finished, those candidates that are not contained in the dictionary are additionally processed by the word-splitting step. The base of our approach to word-splitting is a combinatorial search algorithm. It searches the word token trying to split it between each character and keeping the N best results of such splits. The probabilities of the resulting splits are calculated using a statistical language model (explained in Section 3.4). The decision whether to perform the split at a given point is governed by dictionary lookup and several heuristics.

Dictionary lookup is based on a dictionary organized into a trie [4]. As the algorithm progresses through the token, the split candidate descends down the trie. If the current node indicates that we have found a valid word, a split is performed at the corresponding point. An example of a trie is presented in Fig. 2. On the word token *onje*, consisting of valid words *on* (*he*) and *je* (*is*), the algorithm starts with the candidate positioned at the root node $\langle s \rangle$. As the algorithm progresses the candidate descends down the branch $\langle s \rangle \rightarrow o \rightarrow n$. Whenever a valid word is encountered (designated by a double-ringed node), a new candidate is generated containing the split after the valid word.

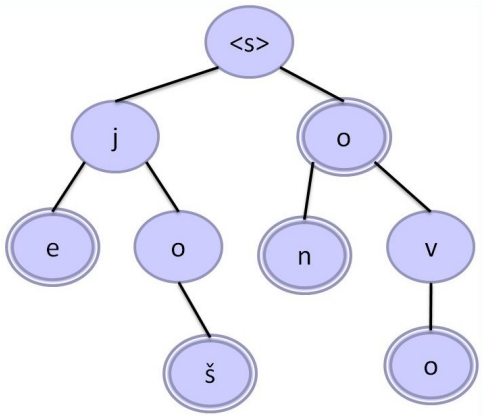

Figure 2: Trie example

Alternatively, there are several splitting heuristics employed in the system. We assume that upper case letters usually begin words and perform the split on their occurrence. Similarly, we split the word token on the occurrence of a sequence of digits, assuming that words do not contain digits.

Our system processes only the tokens that are not contained in the dictionary, so merge errors that result in a valid word can not be corrected. One way of addressing this issue could be by keeping a precompiled list of merges that result in a valid word and checking every token against that list. However, our preliminary results showed that this approach generates more errors than it corrects, so we currently decided to leave it out.

3.4 Statistical language model

A statistical language model is used as a discerning factor in choosing the most likely candidate for the error input, and also as a guide in the word-splitting step. The theory of language modelling [1] states that the probability of a word sequence can be calculated as follows:

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_1^{k-1}), \tag{7}$$

where w_s^t is a word sequence $\langle w_s, w_{s+1}, \ldots, w_t \rangle$.

In practice the model described by (7) has a high memory cost, so it is approximated by the Nth-order Markov model (N usually being one or two). In our system we use the first-order Markov model, thus only looking at the preceding word. That model, also called the bigram model, is represented by the equation:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1}), \tag{8}$$

with w_0 being a special token $\langle s \rangle$ meaning "Start of sentence" [1]. The equation calculates the probability of a word sequence as the product of conditional probabilities of each consecutive pair of words.

In the case of previously unseen bigrams we use Witten-Bell discounting. The idea behind it is that the probability of encountering a new bigram can be approximated by the ratio of the number of observed bigram types to the total number of bigrams. This can be incorporated into the equation for the probability $p^*(w_i|w_{i-1})$ of seeing an unknown bigram beginning with w_{i-1} as follows [1]:

$$p^*(w_i|w_{i-1}) = \frac{T(w_{i-1})}{Z(w_{i-1})(N(w_{i-1}) + T(w_{i-1}))}, \tag{9}$$

where $T(w_{i-1})$ is the number of bigram types, $Z(w_{i-1})$ is the number of zero-probability bigrams, and $N(w_{i-1})$ is the total number of bigrams starting with w_{i-1}. In order to preserve the probability mass, the probability of seen bigrams must be discounted using the following equation:

$$\sum_{i:c(w_x w_i)>0} p^*(w_i|w_x) = \frac{c(w_x w_i)}{c(w_x) + T(w_x)}. \tag{10}$$

4 Experimental evaluation

This section presents the evaluation of the performance of the given system. Two preliminary experiments were made: one with raw OCR output, and the other with the same OCR data corrected manually beforehand by the Croatian News Agency.

4.1 Training and test data

Experiments were performed on real OCR data acquired through the use of Abbyy FineReader[1] with Croatian language support. We refer to these data

[1] http://finereader.abbyy.com/

as *raw data*, although Abbyy FineReader probably performs some rudimentary OCR correction. The total data set consisted of 827 OCR-ed newspaper articles from various sources, of which 758 were used for extracting character confusions and determining their probabilities. The remaining 69 documents, which consisted of 32,541 words, were corrected manually by the authors of this paper and used as the ground truth for evaluation.

A corpus consisting of articles from the daily Croatian newspaper *Vjesnik* and the Official Gazette of the Republic of Croatia was used to build a dictionary and a language model. To account for various (and some rare) inflectional wordforms, we used the procedure described in [8] to first acquire an inflectional lexicon from the corpus, and then expanded this lexicon into a dictionary of nearly a million wordforms. Since this procedure is fully automatic, the dictionary is not error free and contains a number of morphologically invalid wordforms. This, however, should not be problematic because morphological errors in general do not coincide with OCR errors.

4.2 Performance measures

Two different measures were used to evaluate OCR error correction: text accuracy and error reduction rate. Text accuracy is the ratio of the number of correct words in the processed texts to the total number of words in the manually corrected text. This measure describes the percentage of correct text according to the ground truth. It equals $1 - ErrorRate$, where $ErrorRate$ is a measure given in [3]. Error reduction rate, a measure used in [10], gives the percentage of valid corrections penalized by the number of newly generated errors, as follows:

$$ERR = \frac{c(correct) - c(generated)}{c(correct) + c(incorrect)}. \qquad (11)$$

4.3 Results

The experiments were performed using different correction methods; the results of the evaluation are shown in Table 2. The first column shows the methods used in each experiment, with *conf.* standing for the use of character confusions for candidate generation, *n-gram* for the n-gram model, *split* for word merge errors correction, and *LM* for the use of the language model.

The first experiment shows an increase in overall text accuracy achieved by using the proposed system on raw OCR data. The best results are achieved using candidate generation based on character confusions, word splitting and language model (0.61% text accuracy improvement and 22.8% error reduction rate). The use of character confusions generated the majority of valid candidates. Word splitting and the language model improved the error reduction rate by about 2% each. However, word splitting had a greater effect on text accuracy because text accuracy increases with each valid word being split, whereas the number of valid corrections increases only if all the words in a word merge error are split correctly. For example, the splitting of the word token *dajebaruspjelodijeljenje* to words *da, je, baruspjelo, dijeljenje* generated three valid words and one invalid (*baruspjelo*). This correction increased the number of correct words by three, whereas the number of errors remained the same because the correction made was not entirely successful, thus increasing only text accuracy.

The results reveal that using the n-gram model degrades the performance of the system. This is because the n-gram model generates a large number of new errors, compared to the number of valid corrections it brings. If a valid word is processed that does not exist in the dictionary, a correction attempt will be made that may change the valid word and produce a new error. This is counted towards the number of generated errors. Also, during candidate generation a valid correction is often produced and later discarded because it does not exist in the dictionary, thus adding to the number of invalid corrections. A possible reduction of generated errors and invalid corrections might be achieved through the use of a larger dictionary.

The second experiment was performed on the hand corrected OCR output, which, as the results reveal, is still not 100% correct. By itself, manual correction achieved 99.31% text accuracy, which amounts to a 1.66% improvement over the raw OCR output. Further correction using the proposed system resulted in a minor increase of text accuracy in some cases, whereas a slight decrease was observed in others. The overall results show that manual correction currently outperforms our system; the system achieves up to 36.75% (i.e., 0.61/1.66) of the performance of manual correction.

The comparison of results with similar systems

Table 2: Text accuracy and error reduction rate

	Text accuracy (%)			Error reduction rate			
Input text/Correction method	Original	Corrected	Change	Correct	Incorrect	Generated	ERR (%)
OCR output							
conf.	97.65	97.98	+0.33	114	447	**8**	+18.9
conf. + LM	97.65	98.02	+0.38	123	438	**8**	+20.5
conf. + split + LM	97.65	**98.26**	**+0.62**	152	409	24	**+22.8**
conf. + n-gram	97.65	97.95	+0.31	122	439	32	+16.1
conf. + n-gram + LM	97.65	97.97	+0.33	123	438	27	+17.1
conf. + n-gram + split + LM	97.65	98.20	+0.56	**163**	**398**	59	+18.5
OCR output + Hand correction							
conf.	99.31	99.34	+0.03	16	180	6	+5.1
conf. + LM	99.31	99.35	+0.04	14	182	**1**	+6.6
conf. + split + LM	99.31	**99.39**	**+0.08**	19	177	2	**+8.7**
conf. + n-gram	99.31	99.27	−0.04	19	177	32	−6.6
conf. + n-gram + LM	99.31	99.28	−0.03	18	178	28	−5.1
conf. + n-gram + split + LM	99.31	99.31	+0.00	**22**	**174**	29	−3.6

is complicated by the fact that different sets and different measures are used for the evaluation, and that there are differences in what constitutes a valid and an invalid correction. Although extremely different in approaches, we have achieved similar results to Klein and Kopel [3] for the final text accuracy. OCRSpell was evaluated with user interaction, expectedly demonstrating better text accuracy than our system. The measure used by Tong and Evans [10] is the same as ours, but they only consider literal words containing letter sequences, whereas we consider all tokens, which additionally deteriorates our results and makes the comparison questionable.

5 Conclusion

Correcting OCR errors is a difficult problem, whose solution would have positive effects on a diverse set of tasks, e.g., information retrieval. Unfortunately, achieving the accuracy of manual correction rate currently seems an unlikely task. We have presented an OCR correction system for the Croatian language utilizing different methods for correcting character and word merge errors. Our system demonstrated an increase of 0.62% in text accuracy and the error reduction rate of 22.8%, which amounts to 36.75% of the performance of manual correction. These results show that our system can not completely substitute manual correction, but it can be used to speed up the process. To the best of our knowledge, this paper presents the first OCR error correction system for the Croatian language.

As our evaluation was performed on a relatively small data set, we plan to perform further, more detailed evaluation in the future. Also, as the correction rate of any automatic system can not yet match manual correction, we are considering to implement a semi-automatic system. That way we could lower the time needed for correcting the text, while retaining high accuracy of manual correction. For correcting the word merge errors we are planning on looking into the forward-DP backward-A^* algorithm, as described in [7]. The implemented bigram language model on some occasions does not provide enough contextual knowledge, so some experimentation with a trigram or higher order model could provide us with better evaluation results. Also, we plan on investigating how results are affected by the quality of the wordforms dictionary.

Acknowledgements

The authors would like to thank their colleague Ognjen Lajšić for the initial work on the OCR error correction system providing some of the ideas

employed in the paper. We would also like to thank our colleagues Siniša Biđin, Sonja Grđan, Ante Kegalj, Tomislav Lombarović, Veljko Srdarević, and Ana Stopić, who have worked with us on the first version of the system. This work has been supported by the Ministry of Science, Education and Sports, Republic of Croatia and under the Grant 036-1300646-1986. The authors are grateful to the Croatian News Agency (HINA) for making available the OCR data set.

References

[1] D. Jurafsky, J.H. Martin, A. Kehler, K. Vander Linden, and N. Ward. *Speech and language processing*. Prentice Hall New York, 2000.

[2] K. Kise, T. Shiraishi, S. Takamatsu, and H. Kusaka. Improvement of Text Image Recognition Based on Linguistic Constraints. In *Proc. of the Conference for Machine Visions Applications, Tokyo*, pages 511–514, 1992.

[3] ST Klein and M. Kopel. A voting system for automatic OCR correction. In *Proc. of the SIGIR 2002 Workshop on Information Retrieval and OCR: From Converting Content to Grasping Meaning, Univ. of Tampere*, 2002.

[4] D.E. Knuth. *The art of computer programming. Vol. 3, Sorting and Searching*. Addison-Wesley Reading, MA, 1973.

[5] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 10(8), 1966.

[6] W. Magdy and K. Darwish. Arabic OCR error correction using character segment correction, language modeling, and shallow morphology. In *Proc. of the 2006 conference on empirical methods in natural language processing*, pages 408–414. Association for Computational Linguistics, 2006.

[7] M. Nagata. A stochastic Japanese morphological analyzer using a forward-DP backward-A^* N-best search algorithm. In *Proc. of COLING*, volume 94, pages 201–207, 1994.

[8] J. Šnajder, B. Dalbelo Bašić, and M. Tadić. Automatic acquisition of inflectional lexica for morphological normalisation. *Information Processing and Management*, 44(5):1720–1731, 2008.

[9] K. Taghva and E. Stofsky. OCRSpell: an interactive spelling correction system for OCR errors in text. *International Journal on Document Analysis and Recognition*, 3(3):125–137, 2001.

[10] X. Tong and D.A. Evans. A statistical approach to automatic OCR error correction in context. In *Proc. of the fourth workshop on very large corpora*, 1996.