Optical character recognition applied on receipts printed in Macedonian language

Martin Gjoreski, Gorjan Zajkovski, Aleksandar Bogatinov, Gjorgji Madjarov, Dejan Gjorgjevikj
Faculty of Computer Science and Engineering
Skopje, Macedonia

Hristijan Gjoreski
Department of Intelligent Systems
Jožef Stefan Institute
Ljubljana, Slovenia

***Abstract*— The paper presents an approach to Optical Character Recognition (OCR) applied on receipts printed in Macedonian language. The OCR engine recognizes the characters of the receipt and extracts some useful information, such as: the name of the market, the names of the products purchased, the prices of the products, the total amount of money spent, and also the date and the time of the purchase. We used the publicly available OCR framework Tesseract, which was trained on pictures of receipts printed in Macedonian language. The results showed that it can recognize the characters with 93% accuracy. Additionally, we used another approach that uses the original Tesseract to extract the features out of the picture and the final classification was performed with k-nearest neighbor's classifier using dynamic time warping as a distance metrics. Even though the accuracy achieved with the modified approach was for 6 percentage points lower than the original approach, it is a proof of concept and we plan to further research it in future publications. The additional analysis of the results showed that the accuracy is higher for the words which are prescribed for each receipt, such as the date and the time of the purchase and the total amount of money spent.**

Keywords—OCR; Receipt digitalization; Tesseract; DTW;

I. Introduction and Related work

Optical Character Recognition (OCR) is conversion of photographed or scanned images, which contain printed or typewritten text, into machine readable characters (text). The basic idea origins since 1929 when the first OCR patent is obtained by Tausheck [1]. It is based on template matching by using optics and mechanics. After the first commercial computer (UNIVAC I) is installed (1951), the era of converting images of text into computer readable text has started. In 1956 the first approach to convert images of text into computer readable text was presented [2]. At that time hardware and software are strong limitations, so the OCR approaches are based on template matching and simple algebraic operations. Since then a lot research has been done on OCR and with the advancement of the technology more complex OCR approaches are developed. Today OCR is done in much more intelligent way, but it also requires more computational power, which can be a problem for smartphone implementations.

OCR can be used in common industries and applications including date tracking on pharmaceutical or food packaging, sorting mail at post offices and other document handling applications, reading serial numbers in automotive or electronics applications, passport processing, secure document processing (checks, financial documents, bills), postal tracking, publishing, consumer goods packaging (batch codes, lot codes, expiration dates), and clinical applications. Also OCR readers and software can be used, as well as smart cameras and vision systems which have additional capabilities like barcode reading and product inspection.

In recent years, numerous OCR-based smartphone applications were also introduced. A successful example application is the Google's Goggles application [3], which has more than 10 million downloads. Beside the OCR functionality it has several others such as: image search, text translation, bar code scanner. Their OCR engine can analyze text in several languages, not including the Macedonian language. Additionally, the implementation of the OCR engine is not on the smartphone itself, but on a server and therefore it requires internet connection in order to perform an OCR action. Recently Google has allowed public and freely available API for their OCR engine [4], which resulted in numerous smartphone OCR-based applications. However they can only be used with internet connection and furthermore, the API does not provide support for the Macedonian language. Finally, there are some examples of OCR-based applications that claim to support the Macedonian language, e.g., Translang [5]. However, none of them supports an OCR for Cyrillic script, which is the official script of the Macedonian language.

In this paper we present an application of OCR on receipts printed in Macedonian language. The next section presents the methodology used for the process of OCR. Then, in the Experimental Results section, the recognition accuracy is presented. Finally, the conclusion and a brief discussion about the approach and the results are given.

II. Methodology

Figure 1 shows the whole process of the OCR. First, the user takes a photo of a receipt that he/she received from a market. Then, the OCR engine recognizes the characters printed on the receipt and therefore extracts some useful information out of the receipt, e.g., the name of the market, the names of the products purchased by the user, the prices of the products, the total amount of money spent, and also the date and the time of the purchase. For the process of the OCR, the open source OCR engine called Tesseract [6] is used.

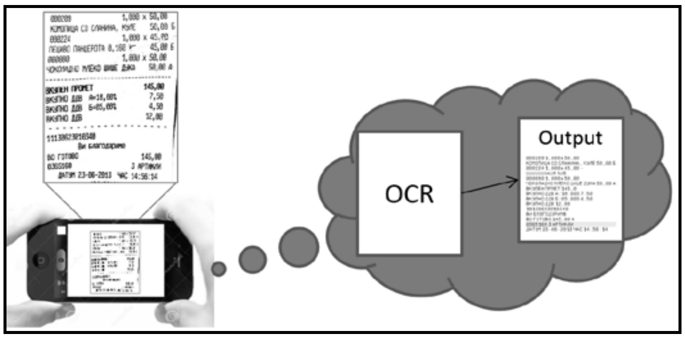

Figure 1. The OCR process applied on a receipt printed in Macedonian language (Cyrillic script).

A. Tesseract

Creating an OCR engine is a challenging research task and requires great knowledge in image processing, feature extraction and machine learning. However, there are several open source projects that provide OCR framework and are widely used in the creation of OCR-related applications. In order not to reinvent the wheel and also to save time for development, in this study we decided to use an OCR framework which is freely available. After studying several frameworks, we decided to use the Tesseract. Tesseract is OCR engine that is developed by HP between 1984 and 1994 to run in a desktop scanner, but it is never used in an HP product [7]. Since then it has a lot of improvements. In 2005 it becomes open source and is managed by Google since then. The last stable version (V3.02) is released in 2012 and V3.03 is expected to be released in 2014. Tesseract is written in C and C++ but it also has Android and iOS wrappers which make it useful for smartphone application.

1) Tesseract Architecture

The first approach that is tested in the process of character recognition is the original Tesseract engine. Tesseract has traditional step-by-step pipeline architecture (shown in Figure 2). First image preprocessing is done with adaptive thresholding where a binary image is produced. Then connected component analysis is done to provide character outlines. Next techniques for character chopping and character association are used to organize the outlines into words. In the end two-pass word recognition is done by using methods of clustering and classification. For the final decision about the recognized word, Tesseract consults with both language dictionary and user defined dictionary. The word with smallest distance is provided as an output. This is just brief overview of the Tesseract architecture, more details can be found in the authors' literature [7].

2) Training Tesseract

For the training phase, Tesseract needs a photograph (tiff or pdf file) of a text written in the same language as the one that it is trying to recognize. For each character from the learning text Tesseract extracts 4 different feature vectors. Then it uses clustering technique to construct a model for each character and those models are later used in the classification phase for decision of which character should be recognized.

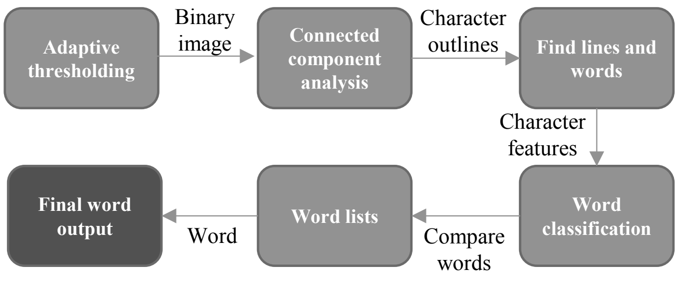

Figure 2. Tesseract OCR engine architecture.

For preparing the training text, several different approaches were tested regarding the font of the training text, the size of the characters in the training text and the content of the training text. Tests were done for each of the three problems. In the first approach the training text was written with a font that was made of quality photographs of single characters. In the second approach the training text was written with a font that is similar to the font of the receipts. For the size of the characters in the training text tests were done with different font sizes starting from 16px to 48px. Regarding the content of the training text two different approaches were tested. With the first approach for each character that the model is trying to recognize there are 10 to 25 different instances with respect to the frequency of the character in the Macedonian language. For example the count of the vowels was 20-25 and the count of the special characters or very infrequent characters such as Н or Ѕ was 10-15. In the second approach the training text was consisted of 1300-1500 random sampled words from different receipts.

The tests showed that the engine is most accurate if the size of the letters in the training text is similar to the text on the photographed receipts. In this case the size that is used is 40px. Also it was concluded that better results can be achieved if the training text is consisted of random sampled words from different receipts the second approach. After all the testing done on Tesseract, the training text that was used for further analysis consisted of 1300-1500 random sampled words from different receipts, it was written with a font similar to the font of the receipts and the size of the characters was 40px.

B. Tesseract-DTW

For the process of character recognition we also tried another approach that uses Dynamic Time Warping (DTW) [8] and K-Nearest Neighbors (KNN) classifier [9]. This approach, Tesseract-DTW (shown in Figure 3), uses the original Tesseract only for feature extraction; the final classification is performed by the KNN classifier using the DTW as a distance metrics. The DTW metric was chosen because the size of each feature vector extracted by the Tesseract varies, and is not the same for each character. Please note that applying a standard classifier such as decision tree, SVM, etc., was not an option because of the varying size of the feature vectors.

1) DTW

DTW also known as dynamic programming matching is a well-known technique to find an optimal alignment between two given sequences [8]. It finds an optimal match between two sequences of feature vectors by allowing stretching and compression of sections of the sequences. DTW first has been used by Sakoe and Chiba [10] to compare different speech patterns in automatic speech recognition. In fields such as data mining and information retrieval, DTW has been successfully applied to automatically cope with time deformations and different speeds associated with time-dependent data. Also it successfully has been used both for online [11] and offline signature verification [12].

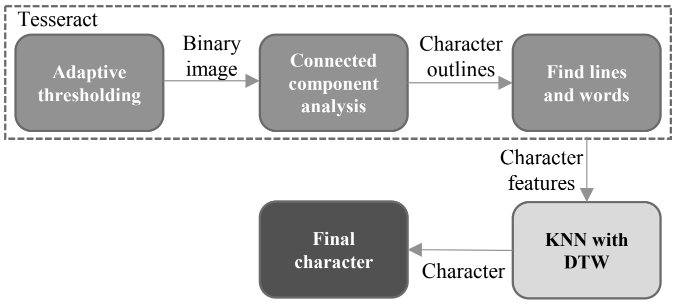

Figure 3. Tesseract-DTW architecture.

2) DTW distance

To calculate the distance between two vectors *X1 = (x11, x12, ..., x1i),* and *X2 = (x21, x22, ..., x2j)*, DTW needs a local cost measure, sometimes also referred to as local distance measure. In this study an Euclidean distance is used as cost measure, see equation (1). By evaluating the local cost measure for each pair of elements of the sequences *X1* and *X2*, cost matrix *M* is calculated, see equation (2). The goal is to find an alignment between *X1* and *X2* having minimal overall cost. For calculating the minimal overall cost three conditions must be satisfied: boundary condition, monotonicity condition and step size condition. The minimal overall cost is the output of the DTW algorithm, shown in equation (4).

$$Cost\ (x1i, x2j) = Euclid\ (x1i, x2j) \quad (1)$$
$$M[i][j] = Cost\ (x1i, x2j) \quad (2)$$
$$DTWdist\ (X1, X2) = M_{[1][1]} + S_{min} + M_{[i][j]} \quad (3)$$

Where, $S_{min} = \sum (\min (M_{[k+1][t]}, M_{[k][t+1]}, M_{[k+1][t+1]}))$, $k \epsilon \{1, 2, ..., i\text{-}2\}$ and $t \epsilon \{1, 2, ..., j\text{-}2\}$.

3) Evaluating Tesseract-DTW

For evaluating the Tesseract-DTW approach 6 photographs of different receipts were used. 5 of them were used as training samples and 1 as a test sample. This is repeated 6 times so each of the receipts was used once as a test sample.

First each character of the training receipts is labeled. Then feature extraction is done by using Tesseract. After the feature extraction each character of the learning receipts is described with 4 feature vectors (4). *X* and *Z* are with variable size (5) and *Y* and *W* are with constant size (6).

$$C_1 = (X^1, Y^1, Z^1, W^1) \quad (4)$$
$$X^1 = (x_1, x_2, ..., x_m), Z^1= (z_1, z_2, ..., z_j) \quad (5)$$
$$Y^1 = (y_1, y_2, y_3), W^1 = (w_1, w_2, w_3) \quad (6)$$

In the classification phase KNN classifier was used. For calculating the distance between two characters C_1 and C_2 combination of DTW and Euclidean distance measurement is used. DTW is used for calculating the distance between the vectors with varying size (7) and Euclidean distance is used for calculating the distance between the vectors with the no varying size (*8*). After DTW and Euclidean distance is calculated between the corresponding vectors of the two characters the final distance between the two characters is calculated with Euclidean distance based on the four distances (*d1, d2, d3, d4*) calculated in the previous step (9). The character with the smallest distance to the test character is chosen as the output of the classifier.

$$d1 = DTW_{dist}\ (X^1, X^2),\ d3 = \mathrm{DTW}_{\mathrm{dist}}\ (Z^1, Z^2) \quad (7)$$
$$d2 = Euclid\ (Y1, Y2),\ d4 = Euclid\ (W1, W2) \quad (8)$$
$$Distance\ (C_1, C_2) = Euclid\ (d1, d2, d3, d4) \quad (9)$$

III. Experimental results

Figure 4 shows an accuracy comparison for the two approaches used for character recognition, Tesseract and Tesseract-DTW. The comparison is performed using the number of correctly recognized characters from 6 photographed receipts. One can note that only for the third photograph, the Tesseract-DTW is better than the original Tesseract. In all other cases the original Tesseract approach is better. On average, the Tesseract is better for 6 percentage points. Also compared by time of execution Tesseract was better that Tesseract-DTW, which was in a way expected given the complexity of the DTW and the usage of the so called “lazy” (instance-based) classifier – KNN.

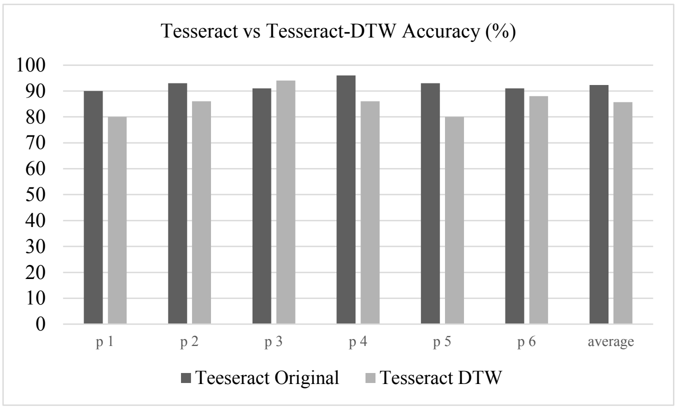

Figure 4: Accuracy for correctly recognized characters by using Tesseract and Tesseract-DTW.

IV. Discussion and Concusion

The paper presented an approach of OCR for receipts printed in Macedonian language. The main OCR engine that was used is Tesseract. In the process of character recognition two approaches were tested. In the first approach the original Tesseract was tested. Tests showed that Tesseract is most accurate when the training consists of random sampled words from different receipts and is written with similar font and size as the characters that we are trying to recognize. In the second approach modified version of Tesseract was used (Tesseract-DTW). In this approach the feature extraction was again performed by the Tesseract, however the final classification was done with KNN classifier using DTW as distance metrics. Tests showed that the first approach by using original Tesseract engine outperformed the second approach by 6 percentage points. Further analysis showed that the accuracy is higher for the numbers and words which are prescribed for each receipt, such as the date and the time of the purchase and the total amount of money spent. This was in a way expected because the classifier has more examples to train on, i.e. they are present in each receipt and the numbers are limited only to 10 characters. On the other hand, the names of the products are more difficult to recognize mainly because there are names that are not from Macedonian language. In general, the more data is used for training, the better the model should be. In future we plan to collect much more data samples by providing a free smartphone application.

To the best of our knowledge, this is the first attempt to apply OCR on receipts printed in Macedonian language using the Cyrillic script, and moreover the first attempt to modify the original Tesseract by applying KNN algorithm using DTW distance metrics. Even though, the modified version of the Tesseract achieved slightly worse results, it gives promising results and we plan to further improve it in the future work. We are also considering an approach that will combine the both methods, e.g. by using meta-learning, and eventually improve the recognition accuracy.

Acknowledgment

The authors would like to thank the developers of the Tesseract OCR framework and making it freely available to the research and developers community.

References

[1] G. Tauschek, “Reading machine” U.S. Patent 2026329, Dec. 1935

[2] S. Mori, C.Y. Suen, K. Yamamoto“Historical Review of OCR research and development”, Proceeding of the IEEE (Volume:80, Issue 7), Jul 1992

[3] Google’s Goggle application.
https://play.google.com/store/apps/details?id=com.google.android.apps.unveil

[4] Google’ API for OCR.
https://developers.google.com/google-apps/documents-list/#uploading_documents_using_optical_character_recognition_ocr

[5] Thanslang application.
https://play.google.com/store/apps/details?id=icactive.app.translang

[6] Tesseract-ocr. Mar-2012. URL: http://code.google.com/p/tesseract-ocr/.

[7] Ray Smith. “An overview of the Tesseract OCR engine". In: Document Analysis and Recognition, 2007. ICDAR (2007).

[8] M. Müller, “Information Retrival for music and motion”, 2007, XVI, 318 p. 136 illus. 39.

[9] D. Aha, D. Kibler (1991). Instance-based learning algorithms. Machine Learning. 6:37-66D. Aha, D. Kibler (1991). Instance-based learning algorithms. Machine Learning. 6:37-66.

[10] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” Acoustics, Speech and Signal Processing, IEEE Transactions on, vol. 26, no. 1, pp. 43–49, 1978.

[11] Y. Qiao, X. Wang, C. Xu, “Learning Mahalanobis Distance for DTW based Online SignatureVerification”, Information and Automation (ICIA), 2011 IEEE International Conference, June 2011

[12] A. Piyush Shanker, A.N. Rajagopalan, “Off-line signature verification using DTW”, Journal Pattern Recognition Letters Volume 28 Issue 12, September 2007