applied sciences

MDPI

Article

Exploiting Script Similarities to Compensate for the Large Amount of Data in Training Tesseract LSTM: Towards Kurdish OCR

Saman Idrees and Hossein Hassani *

Department of Computer Science and Engineering, University of Kurdistan Hewlêr, 30 Meter, Kurdistan Region, Erbil 44001, Iraq; saman.idrees@ukh.edu.krd
* Correspondence: hosseinh@ukh.edu.krd

Featured Application: This work helps in the preparation of OCR for the Kurdish language. In particular, its focus is on Kurdish texts written in Persian-Arabic script. Currently, Kurdish OCR is in its early stages. This work can assist in preparing the environment for a full-fledged OCR application for Kurdish.

Abstract: Applications based on Long-Short-Term Memory (LSTM) require large amounts of data for their training. Tesseract LSTM is a popular Optical Character Recognition (OCR) engine that has been trained and used in various languages. However, its training becomes obstructed when the target language is not resourceful. This research suggests a remedy for the problem of scant data in training Tesseract LSTM for a new language by exploiting a training dataset for a language with a similar script. The target of the experiment is Kurdish. It is a multi-dialect language and is considered less-resourced. We choose Sorani, one of the Kurdish dialects, that is mostly written in Persian-Arabic script. We train Tesseract using an Arabic dataset, and then we use a considerably small amount of texts in Persian-Arabic to train the engine to recognize Sorani texts. Our dataset is based on a series of court case documents in the Kurdistan Region of Iraq. We also fine-tune the engine using 10 Unikurd fonts. We use Lstmeval and Ocreval to evaluate the outputs. The result indicates the achievement of 95.45% accuracy. We also test the engine using texts outside the context of court cases. The accuracy of the system remains close to what was found earlier indicating that the script similarity could be used to overcome the lack of large-scale data.

Keywords: optical character recognition; tesseract; printed-document OCR; Kurdish-OCR system; offline character recognition system

Citation: Idrees, S.; Hassani, H. Exploiting Script Similarities to Compensate for the Large Amount of Data in Training Tesseract LSTM: Towards Kurdish OCR. *Appl. Sci.* **2021**, *11*, 9752. https://doi.org/10.3390/app11209752

Academic Editor: Agnese Magnani

Received: 8 August 2021
Accepted: 27 September 2021
Published: 19 October 2021

1. Introduction

The less-resourced languages face various issues from the language technology perspective. The lack of resources often is one of the main obstacles in the forefront of the resolution of those challenges. The Optical Character Recognition (OCR) systems play a crucial role in resource preparation and language processing. However, it is not feasible to develop or adapt an appropriate OCR for those languages if the required data does not exist. Kurdish is considered a less-resourced language that is facing several issues [1] including lack of OCR systems that are accurate and widely available. Tesseract Long-Short-Term Memory (LSTM) is a popular OCR engine that has been adapted for various languages, but LSTM-based methods require large amounts of data for their training. In this research, we suggest a resolution for that issue by using a training dataset for a language with a similar script and then applying a considerably small amount of data from the target language. Our target language is Kurdish, and we choose Sorani, one of its dialects that is mostly written in Persian-Arabic script. Therefore, we use an Arabic dataset as the base of the training, and we use a Kurdish dataset based on a series of court case

documents in the Kurdistan Region of Iraq. We also investigate the fine-tuning phase of Tesseract using 10 Unikurd fonts and evaluate the approach by Lstmeval and Ocreval.

The LSTM-based methods are not only used in image recognition systems, but they are also applicable to address and solve a wide range of problems. For example, they are used in time-series forecasting [2], energy consumption forecasting [3], commercial vacancy prediction [4], and various other areas. Although this paper focuses on suggesting a solution for the requirement of large amounts of data in one aspect of the applications of LSTM-based approaches, the suggestion could be extended to similar cases that have similarities of some sort in their training data.

The rest of this paper is organized as follows. Section 2 provides a background about the Kurdish language focusing on its alphabets and script similarities with Arabic, and then briefly introduces the OCR history focusing on Tesseract. Section 3 reviews the related work and literature. In Section 4, we discuss our approach. Section 5 presents and discusses the results. Finally, Section 6 concludes the paper.

2. Background

In recent years, research on Kurdish language technology has gained more attraction. Kurdish is spoken by approximately 30 million people [1]. It is a multi-dialect language, and its two most widely spoken dialects, Kurmanji (Northern Kurdish) and Sorani (Southern Kurdish), are spoken by roughly 75% of native Kurds [5,6]. Sorani usually is written in an adapted Persian-Arabic script with a cursive style and from Right To Left (RTL). Kurmanji mostly uses Latin for writing, except in the Kurdistan Region of Iraq and Kurdish areas of Syria, where they use the same script that Sorani uses [1]. The alphabet of the mentioned script includes 34 letters that appear in different shapes according to their position in words (see Table 1). Kurdish language technology and its challenges have been addressed and categorized as: dialect heterogeneity, script variety, lack of standards, lack of resources, and lack of investment [1,7,8].

Table 1. Kurdish alphabet forms, both Ligature and Isolated.

	Contextual Form			
#	**Isolated**	**Initial**	**Medial**	**Final**
1	ء	ئـ	ـئـ	ـئ
2	ا		ـا	
3	ب	بـ	ـبـ	ـب
4	پ	پـ	ـپـ	ـپ
5	ت	تـ	ـتـ	ـت
6	ج	جـ	ـجـ	ـج
7	چ	چـ	ـچـ	ـچ
8	ح	حـ	ـحـ	ـح
9	خ	خـ	ـخـ	ـخ
10	د	د	ـد	ـد
11	ر	ر	ـر	ـر
12	ڕ	ڕ	ـڕ	ـڕ
13	ز	ز	ـز	ـز
14	ژ	ژ	ـژ	ـژ
15	س	سـ	ـسـ	ـس
16	ش	شـ	ـشـ	ـش
17	ع	عـ	ـعـ	ـع
18	غ	غـ	ـغـ	ـغ
19	ف	فـ	ـفـ	ـف
20	ق	قـ	ـقـ	ـق

Table 1. *Cont.*

	Contextual Form			
#	**Isolated**	**Initial**	**Medial**	**Final**
21	ق	قـ	ـقـ	ـق
22	ك	كـ	ـكـ	ـك
23	گ	گـ	ـگـ	ـگ
24	ل	لـ	ـلـ	ـل
25	ڵ	-	ـڵـ	ـڵ
26	م	مـ	ـمـ	ـم
27	ن	نـ	ـنـ	ـن
28		هـ	ـهـ	-
29		ه		ـه
30		و		ـو
31		ۆ		ـۆ
32		وو		ـوو
33	ی	یـ	ـیـ	ـی
34	ێ	ێـ	ـێـ	ـێ

Table 2 shows the order of the alphabet of the Persian-Arabic script based on the Kurdish Academy proposal. The Arabic alphabet has 28 letters with a cursive style (see Table 3). The Persian writers augmented the Arabic alphabet to include graphemes for the phonemes that did not exist in Arabic. The Kurdish writers also did the same on the Persian-Arabic version [9]. The augmented scripts are known as Persian-Arabic scripts though the Kurdish one includes more new graphemes for the phonemes than the one that is used in Persian (Farsi) writing.

Table 2. Central Kurdish alphabet.

ئ	ا	ب	پ	ت	ج	چ	ح	خ	د	ر	ڕ	ز	ژ	س	ش	ع
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
غ	ف	ڤ	ق	و	گ	ل	ڵ	م	ن	ه	ک	ه	ۆ	وو	ی	ێ
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

The Kurdish Persian-Arabic script uses 21 out of 28 Arabic letters, keeps all four extra letters added by Persians, and adds nine letters. Seven Arabic letters were found unsuitable for Kurdish because they have no corresponding sound (phoneme) in the language. Unsuitable letters for the Kurdish language are ,ث ,ذ ,ص ,ض ,ط ,ظ and ع. Five new letters have been formed out of the residual letters, as shown in Tables 4 and 5. As a result, the current Kurdish (Sorani) alphabet consists of 34 characters.

Table 3. Arabic alphabet.

#	Contextual Form			
	Isolated Form	**Initial**	**Medial**	**Final**
1	ء (used mainly in the medial and final position, which is an unlinked letter)		أ	ـأ
			إ	ـإ
			ؤ	ـؤ
		ئـ	ـئـ	ـئ ـيـ
2	ا		ـا	
3	ب	بـ	ـبـ	ـب
4	ت	تـ	ـتـ	ـت
5	ث	ثـ	ـثـ	ـث
6	ج	جـ	ـجـ	ـج
7	ح	حـ	ـحـ	ـح
8	خ	خـ	ـخـ	ـخ
9		د		ـد
10		ذ		ـذ
11		ر		ـر
12		ز		ـز
13	س	سـ	ـسـ	ـس
14	ص	صـ	ـصـ	ـص
15	ض	ضـ	ـضـ	ـض
16	ط	طـ	ـطـ	ـط
17	ظ	ظـ	ـظـ	ـظ
18	ع	عـ	ـعـ	ـع
19	غ	غـ	ـغـ	ـغ
20	ف	فـ	ـفـ	ـف
21	ق	قـ	ـقـ	ـق
22	ك	كـ	ـكـ	ـك
23	ل	لـ	ـلـ	ـل
24	م	مـ	ـمـ	ـم
25	ن	نـ	ـنـ	ـن
26	ه	هـ	ـهـ	ـه
27		و		ـو
28	ي	يـ	ـيـ	ـي

Table 4. Modified Arabic letters to Kurdish letters.

#	Modified Arabic Letters to Kurdish		
	Arabic Letters	**TO**	**Kurdish Letters**
1	ب		پ
2	ج		چ
3	ز		ژ
4	ف		ڤ
5	ک		گ

As indicated by Hashemi [10], the Kurdish alphabet has seven vowels, which are presented with seven corresponding letters (see Table 6).

Table 5. Diacritics added to the Arabic letters to create Kurdish letters.

Arabic Letters with Kurdish Diacritics			
#	Arabic Letter	TO	Kurdish Letter
1	ر		ڕ
2	ل		ڵ
3	و		ۆ
4	ی		ێ

Table 6. Vowel letters in Kurdish alphabet.

Kurdish Alphabet Vowel			
#	Kurdish Vowel	API	Example
1	ا	**ä**	بـا (air)
2	ه	**є**	سەر (head)
3	و	**u**	کورد (Kurd)
4	ۆ	**ö**	تۆ (You)
5	وو	**ü**	دوور (far)
6	ی	**ï**	شین (blue)
7	ێ	**ë**	دێ (village)

The Kurdish alphabet also uses a specific character named Zero-Width Non-Joiner (ZWNJ) for digital writing. ZWNJ is a non-printing character placed between two characters to avoid inappropriate concatenation. For example, after the character "ه". ZWNJ is encoded in Unicode as U+200C. It is illustrated in the following words: هەروەها - بەهار - هەژار - هەولێر. If ZWNJ was not used, those words appeared as هەژار - بههار - هەروەها - هەولێر.

Optical Character Recognition

Optical Character Recognition (OCR) is a research area that benefits from several computing fields, such as machine learning, computer vision, and natural language processing [11]. OCR essentially converts two types of documents into texts: handwritten and machine-typed [12]. Those documents may hold different kinds of data, such as passport documents, invoices, bank statements, digital receipts, business cards, mails, and newspapers, and OCR can make them ready for text processing.

Early character recognition systems followed the telegraphy applications technology. That allowed the industry to develop devices that could help blind and visually impaired to read texts [13]. In 1914, Emanuel Goldberg developed a machine that could read and convert characters into standard telegraph codes [14]. Concurrently, Edmund Edward Fournier developed what was a handheld scanner used to move across a printed page to produce tones that corresponded to a specific letter or character [15]. In the 1930s, Emanuel Goldberg built a statistical machine using optical code recognition for searching microfilm archives. Furthermore, in the 1950s, the US Department of Defense developed Geographic Information Systems and Mapping Operations (GISMO) that could read Morse Codes and words on printed pages, character by character [14].

In 1974, Rey Kurzweil developed the Omni-font OCR machine for blind users. The device was able to recognize printed text in any font. In the early 1990s, A. G. Ramakrishnan built a print-to-braille device that could convert scanned images of printed books to braille books. In the new millennium, OCR became widely available both as an online and offline service under different computing platforms [12].

3. Related Work

In this review, our focus is on OCR systems, but we also review some other LSTM-based applications that could benefit from our approach. For the OCR applications, the research focuses on machine-typed documents in the Kurdish language. However, because

the work on Kurdish OCR is in its early stages, we concentrate on OCRs for languages with Arabic-based scripts particularly, Arabic, Urdu, and Farsi (Persian). We also look into available platforms and technologies for OCR development.

3.1. OCR Studies for Persian-Arabic Based Scripts

Yaseen and Hassani [16] worked on OCR for Kurdish (Sorani) texts in Persian-Arabic script. They discussed the features of the script in terms of its special diacritics and cursive style, and accordingly, they proposed a new set of rules for the segmentation process. They used optimized horizontal histogram projection for line segmentation and contour-based segmentation for character segmentation. The research used the Gamera [17] toolkit for classification and reported a 90.82% of accuracy for their approach.

While conducting this study, we also noticed work on Sorani (Central Kurdish) OCR using Tesseract OCR engine (more detail could be found on its GitHub page at https://github.com/Shreeshrii/tesstrain-ckb [accessed on 7 August 2021]). The mentioned work created a language model for Sorani using synthetic data generated over 26 fonts (see Table 7). However, at the time of writing this paper, we could not find any papers, whether peer-reviewed or as work in progress, to explain more than what the GitHub page describes.

Table 7. Fonts used by Shreeshrii to fine-tune the Arabic language model for Sorani.

#	Font Name	#	Font Name	#	Font Name
1	Abd Akre	10	Arial	19	Segoe UI Bold
2	Abd Gare	11	Arial Bold	20	Speda
3	Abd Halabja OLD	12	NRT Bold,	21	Speda Bold
4	Abd Hewler	13	NRT Reg	22	Tahoma
5	Abd Metin Bold	14	Peshang Des 1,	23	Tahoma Bold
6	Abd Umed	15	RudawRegular	24	Ubuntu Kurdish 0.81 met
7	Adobe Arabic Bold	16	Sarchia_Banoka_1	25	Unikurd Chimen
8	Arabic Typesetting	17	Segoe UI	26	Unikurd Goran
9	Unikurd Web	18	Shasenem-kurd		

Radhiah et al. [18] used Artificial Neural Network (ANN) and Hidden Markov Model (HMM) models in the segmentation process of their Arabic OCR. Their focus was on both isolated and concatenated forms of Arabic characters. They reported a 100% recognition accuracy with ANN and 69% with HMM. For HMM, they reported an accuracy rate of 71% and 50% for isolated and concatenated letters, respectively. According to their results, the ANN method misidentified five letters, while HMM approach misidentified nine. However, it is not clear how they achieved a 100% accuracy with ANN while the method wrongly identified five letters.

Nashwan et al. [19] conducted a study for isolated Arabic letters using ANN with Backpropagation and Learning Vector Quantization (LVQ) method. The research showed a 98.81% accuracy. However, it reported an issue about the confusion between isolated and concatenated letter recognition.

The Urdu language is also written from right to left except for the numerals that follow the similar languages wherein they are written from left to right [20]. Urdu consists of 38 letters and ten numeric characters.

A study by Naz et al. [21] proposed preprocessing techniques for OCR systems using four steps. The first step includes document scanning, orientation detection, skew correction, noise removal, binarization, and then other standard preprocessing methods. The second step is segmentation which separates the paragraphs and text lines. The third step is to extract features such as the right, left, top, and bottom points, curves, loops, crosses, and the height of characters in the text. The last step is the classification that uses fuzzy logic rules in two phases to recognize the diacritics and visible features. They used Nasta'liq font of the size of 36 for the proposed method. They reported an accuracy rate

of 100% and 94% for the baseline/single character identification and ligature/concatenated identification, respectively.

Developers of Persian OCR have adopted several methods, such as movement on edge to capture character features, HMM, and morphology of characters. For example, using a chain-code-based approach and nearest neighbor, they trained a classifier, font by font, and then compared the result with a different font as the test set. The method, on average, showed an accuracy rate of 97.4% [22]. Using morphological operators, Jelodar et al. [23] applied hit/miss operators to describe all sub-words combined with the template matching. The study used one font (Lotus) of two different sizes. The test set contained 3000 words (15,000 characters), and the experiments achieved an accuracy rate of 99.9%.

3.2. OCR Technology and Platforms

A variety of commercial companies along with non-commercial communities provide OCR services and applications. The services are available online or offline. However, currently, a functional and reliable Kurdish OCR is not available. In this section, we review some available OCR services in which we concentrate on Tesseract as one of the widely used engines. We also address some other engines that are available as open-source.

3.2.1. Tesseract

Tesseract is one of the most widely used OCR engines that provides a high accuracy rate compared with other available engines [11]. Figure 1 shows a brief history of Tesseract. Tesseract has been adapted for many languages (up to 140 different languages) [24]. Since its version 4.0, it presented a new engine based on Long Short-Term Memory (LSTM). LSTM, as a specific form of Artificial Recurrent Neural Network (RNN), provides substantially higher accuracy on image recognition than Tesseract's earlier versions. Tesseract can be trained from scratch or be fine-tuned based on already trained languages. In 2005, it became open-source, and it is freely available on http://code.google.com/p/tesseract-ocr (accessed on 7 August 2021).

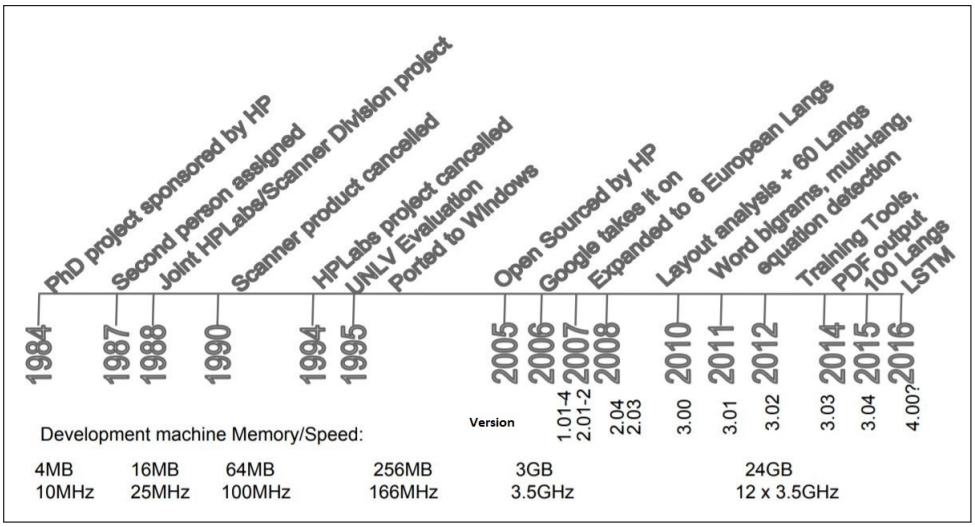


Figure 1. Tesseract Timeline [25].

3.2.2. Other OCR Technologies

Other OCR technologies also exist and are used for different purposes. Gamera is a toolkit for building document image recognition systems [26]. The images or symbols must be assigned to the classes manually, and the result of this step creates a database in the Extensible Markup Language (XML) format. Gamera uses the K Nearest Neighbors (KNN) algorithm as its classifier. Gamera is available on https://gamera.informatik.hsnr.de/index.html (accessed on 7 August 2021).

Google Drive OCR is a multilingual and online OCR service provided by Google Incorporation and supporting up to 200 languages in almost 25 writing systems. According to Genzel and Ashok Popat [27], Hidden Markov Model (HMM) has been employed to work potentially with all languages and writing systems. Although it is a powerful OCR system, it does not support the Kurdish language [28]. Google drive can partially recognize Kurdish documents based on its Arabic model, but it misses the special Kurdish characters that do not exist in the Arabic alphabet.

GOCR/JOCR is an open-source OCR tool under the GNU license [29] which can read and convert images files such as PNM, PGM, PBM, and PP into text format. GOCR does not require training data that results in a fairly low accuracy rate for different languages. The recognition process happens in two stages. In the first stage, GOCR takes the whole document as input, and in the second one, it processes misidentified characters from the same document [29].

Tafti et al. [11] provided a comparative analysis on various OCR engines, their capability, working platform, and other attributes. Recently, a new survey with a broader coverage was also published [30].

Finally, literature reports on research about handwritten Kurdish texts. For example, Zarro and Anwer [31] and Ahmed [32] have reported on handwritten character recognition in Kurdish, focusing on single and isolated characters. However, as we mentioned, our focus is on machine-type documents, and therefore, we are not reviewing those studies in detail.

To summarize, although OCR is a well-developed technology in language processing, most less-resourced languages have not benefited from the advances in the field. Furthermore, as most modern OCR engines use artificial intelligence methods such as LSTM, they require a large amount of data for their training. We are interested in showing that leveraging the script similarity between languages is a solution to that issue. Although the recent literature presents some work on using limited data in OCR adaption, particularly on LSTM-based engines [33–36], it does not show script-similarity-based approaches based on using well-resourced language to the advantage of less-resourced ones. That is what we intend to show in this research. In our method, the emphasis and concentration are on a specific engine (Tesseract).

3.3. Using LSTM-Based Approaches in Other Fields or Study

Peng et al. [2] used LSTM-based methods to forecast time-series forecasting. They used LSTM with fruit FOA-Fly Optimization Algorithm with LSTM (LSTM) to solve time series problems. Peng et al. [3] applied LSTM-based approaches on energy consumption forecasting. They suggested that their model could be retrained with data collected after the COVID-19 pandemic. As these data might not be as large as enough for an LSTM-based system, the idea of using a small dataset based on a trained model over a similar one, as we suggest in this paper, could be advantageous in the mentioned study too. Commercial vacancy prediction is another example of using LSTM-based methods [4]. In this study, Lee et al. [4] applied LSTM-based methods to predict commercial vacancy. Again, a dataset from the after COVID-19 era that could be reasonably smaller than the original dataset could be used for retraining the model.

4. Proposed Method

In this research, we train the Tesseract engine using an Arabic dataset and fine-tune it with a small amount of Kurdish (Sorani) texts. The LSTM-based systems require a large amount of training data. That is typically expensive, labor-intensive, and time-consuming.

Tesseract provides various training methods such as fine-tuning the existing language model and training from scratch. It follows two different approaches for its training: training with synthetic data/image lines and providing ground truth transcriptions. Augmentation could also be applied, and it would be useful when a pre-trained dataset is available.

We train Tesseract and fine-tune it by augmenting the Arabic language model by providing image lines and ground truth transcriptions as input datasets. Using Kurdish documents, we introduce the Kurdish-specific characters. We also examine to what extent a post-processing phase could resolve the issues regarding ZWNJ. Afterward, we retrain the fine-tuned language model with synthetic data to add new Unikurd fonts to compare the accuracy with the post-processing approach.

4.1. Tesseract Training Modes and Available Language Models

Tesseract 4 provides various data models for Latin languages. The models are trained on approximately 400,000 text lines covering roughly 4500 fonts [37]. Although the same number of fonts have not been available for non-Latin scripts, they have also been trained with a similar amount of text lines. The training phase is a long process, and depending on the computation power, it might take a few days or weeks. Despite having different training options, the training steps are identical. Tesseract training options are as follows:

1. Fine-tune: This method adds new data to an already trained model close to the target language model.
2. Cut off the top layer (or some arbitrary layers): If fine-tuning did not produce the desired outcome, this method cuts off the top layer and retrains it with additional data. It can train a new language or a language having script-similarity with a trained one.
3. Retrain from scratch: This is a method that starts from scratch. We do not recommend it if a large amount of data is not available.

According to Google [37], in Tesseract 4, old engines still exist, but they are deprecated and could be removed in the future. Tesseract has two OCR engines (Legacy and LSTM) that can be selected by OCR Engine Mode (–OEM) option, see Table 8. We used the LSTM engine that is faster and appears more promising to provide higher accuracy.

Table 8. Tesseract OCR engine modes.

OEM	Type
0	Legacy engine only
1	Neural nets LSTM engine only
2	Legacy + LSTM engines
3	Default, based on what is available

4.1.1. Two Main Sources of Data for Training Tesseract LSTM Language Model

Below we describe the two main sources of data for training Tesseract LSTM. Other methods also exist but do not suit our approach.

1. Training with Synthetic data: Synthetic data is defined as artificially generated data from text files rather than being captured from actual events. It is used in data mining and other different systems such as fraud detection applications. Synthetic data is usually used in machine learning applications to create a model that could be applied to real data [38].
2. Training with Image lines: This is the process of generating data from actual documents. The documents are obtainable from printed sources, digital images (or scanned documents), or any other suitable and available formats. Ground truth descriptions for each image must be provided.

4.1.2. Data Collection and Preparation

We collect court cases from several courts in the Kurdistan Region of Iraq. We scan the documents with a resolution of 300 Dots Per Inch (DPI) according to Tesseract requirements. For each line of the text, we prepare an image line in PNG. Although various image formats such as Tagged Image File Format (TIFF) are acceptable for this purpose, we chose PNG that is more common for this process. Since no specific tool is recommended in the

literature, any image editor with the capability of cropping and rotation that supports the mentioned image formats can be used for the process. For the synthetic data we use available online sources.

4.2. Evaluation Method

Tesseract has an evaluation program named *lstmeval*. Using a list of *lstmf* files for evaluation, it performs its evaluations. However, other tools are also available for evaluating the accuracy of OCR language models, for instance, *ocreval*, ISRI-OCR's evaluation tool [39]. *Ocreval* is a powerful tool for evaluating character level and word level accuracy that supports all characters represented in UTF-8 encoding and provides more details in comparison to the Tesseract *lstmeval* program. Tesseract *lstmeval* evaluates LSTM-based networks. In our case, it evaluates the performance of the network in an optical character recognition model, while *ocreval* evaluates the overall performance of an OCR system by comparing the output of the OCR-ed document with the original one.

We initially use *lstmeval*, and we compare the results between the original Arabic language model with the fine-tuned *kur_ara* language model. We show the result after applying the post-processing method using Algorithm 1. Then we present the result after training/fine-tuning for adding new fonts. We also use the *ocreval* program to present more detail on results.

Algorithm 1 Post-processing method

if (ه) without ZWNJ exist **then**
  Replace all(ه) without ZWNJ to (ه) including ZWNJ ;
end if
if (ەا) exist **then**
  Replace all (ەا) to (هـا);
end if
if (ەە) exist **then**
  Replace all (ەە) to (هـهـ);
end if
if (ەە) exist **then**
  Replace all (ەە) to (هـه);
end if
if ا exist **then**
  Replace all (ا) to (ها);
end if
if (ەۆ) exist **then**
  Replace all (ەۆ) to (هۆ);
end if
if (ەێ) exist **then**
  Replace all (ەێ) to (هـێ);
end if
if (ەی) exist **then**
  Replace all (ەی) to (هـی);
end if

5. Results

We collected 110 documents in paper format from different town courts in three cities. They are court-registered complaints by civilians about various issues. Documents in the courts are prepared and typed on computer devices by court employees in the desired court format. There are different types of fonts and sizes among the collected documents. We only have limited data set, and therefore, we used 90%, equating to 99 papers, for training the system, and we used 10%, equating to 11 papers for testing and evaluation. A summary of the collected data is presented in Table 9. Table 10 shows the parameters we used in the training process.

Table 9. Summary of data utilized for the training and evaluation process.

#	Type of Data	Amount	Rate %	Total
1	Data utilized for training	99 printed documents	90 %	110 printed documents
2	Data utilized for evaluation	11 printed documents	10 %	
3	Image lines prepared from the documents	522 files	1044 files	
4	Ground Truth transcriptions prepared from documents	522 files		

Table 10. Training Parameters.

Parameter	Value
Page segmentation	Set to 13 default for RTL languages
OCR engine mode	1 (LSTM mode only)
Debug-Interval	−1
Max-Iteration	10,000
Lang-Type	RTL
Norm-Mode	1
Random-Seed	0
OMP-Thread-Limited	8

5.1. Dataset Preparation

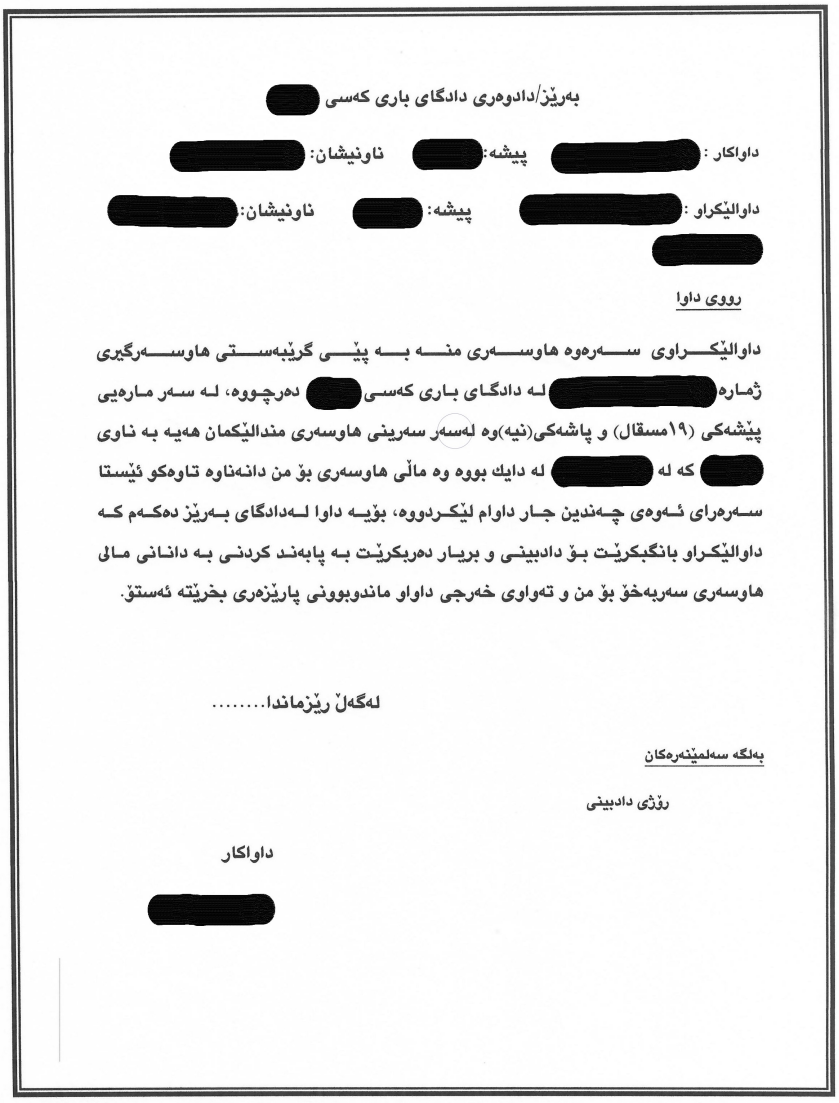
We removed all personally identifiable information (see Figure 2) and image lines with a naming protocol (e.g., image0.png) (see Figure 3). A summary of collected data is presented in Table 9.

Furthermore, we prepared the Ground Truth (GT) Transcription file, which is single-line plain text corresponding to the line images. We saved GT files with 8-bit encoding and named them with the identical name as the line image. We saved with the extension of *.gt.txt*,(e.g., image0.gt.txt), see Figure 4.

We used Notepad++ text editor for the preparation of the single-line plain text and created 522 image lines and 522 GT files, in total 1044 files.

The other preparation activities were performed as follows. Tesseract engine performed *binarization*. Basically, the process aims at converting color images into black and white equivalents. In our case, the documents were already in black and white. However, Tesseract still performed the process. *Dewarping* is another pre-process activity in data preparation. However, it is required when the images have been taken using different kinds of input devices, and therefore they might have distortions, skews, and other attributes that could negatively affect the result of the recognition. In our case, we scanned the documents manually and all with the same settings and a single scanner. Therefore, we did not find it necessary to apply *dewarping*. Because of a small set of data, we also applied *segmentation*, *cropping*, and image *rotation* manually.

In the first phase of training/fine-tuning the Arabic Trained data, the language model was not able to handle ZWNJ properly, and therefore, we replaced all "ہ" without ZWNJ character to "ہ" with ZWNJ character, to unbind wrong concatenated characters. After that, other replacement methods will be applied subsequently to correct common wrongly concatenated characters, see Algorithm 1.


بەڕێز/دادوەری دادگای باری کەسی [redacted]

داواکار : [redacted] پیشە: [redacted] ناونیشان: [redacted]

داوالێکراو : [redacted] پیشە: [redacted] ناونیشان: [redacted]

[redacted]

رووی داوا

داوالێکراوی سەرەوە هاوسەری منە بە پێی گرێبەستی هاوسەرگیری ژمارە [redacted] لە دادگای باری کەسی [redacted] دەرچووە، لە سەر مارەیی پێشەکی (١٩مسقال) و پاشەکی(نیە)وە لەسەر سەرینی هاوسەری منداليكمان هەیە بە ناوی [redacted] کە لە [redacted] لە دایك بووە وە ماڵی هاوسەری بۆ من دانەناوە تاوەکو ئێستا سەرەرای ئەوەی چەندین جار داوام لێکردووە، بۆیە داوا لەدادگای بەڕێز دەکەم کە داوالێکراو بانگبکرێت بۆ دادبینی و بریار دەربکرێت بە پابەند کردنی بە دانانی مالی هاوسەری سەربەخۆ بۆ من و تەواوی خەرجی داواو ماندووبوونی پارێزەری بخرێتە ئەستۆ.

لەگەڵ ڕێزماندا.....

بەڵگە سەلمێنەرەکان

ڕۆژی دادبینی

داواکار

[redacted]

Figure 2. Scanned court document image sample for data set preparation.

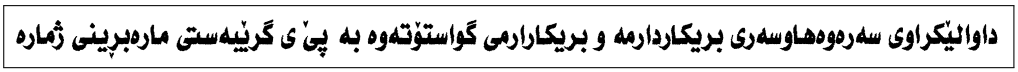

داوالێکراوی سەرەوەهاوسەری بریکاردارمە و بریکارارمی گواستۆتەوە بە پێ ی گرێبەستی مارەبڕینی ژمارە

Figure 3. Line Image sample.

```
داوالێکراوی سەرەوەهاوسەری بریکاردارمە و بریکارارمی گواستۆتەوە بە پێ ی گرێبەستی مارەبڕینی ژمارە
```

Figure 4. Ground Truth transcription sample.

5.2. Result after Training/Fine-Tuning Arabic Trained Data

We trained/fine-tuned the Arabic language model with the targeted dataset and created the *kur_ara* language model. We used *lstmeval* to evaluate and showed the result after training. We achieved a 32 percentage points difference in character error rate and a 51 percentage points difference in word error rate. The accuracy error rate for *kur_ara* language model is reported as follows, the character error rate is 2.93, and the word error rate is 13.53. The detailed results are presented in Table 11. As some Kurdish characters do not exist in the Arabic language model, that magnitude of difference was expected.

Table 11. The evaluation result of trained language model with the starter language model.

Language Models	Char Error Rate	Word Error Rate
Arabic.traineddata	35.10 %	65.03 %
kur_ara.traineddata	2.93 %	13.52 %
Improve rate	32.18 %	51.50 %

To evaluate the accuracy of the trained language model by actual court-printed document images, we used *ocreval* tool. A sample is shown in Figure 5.

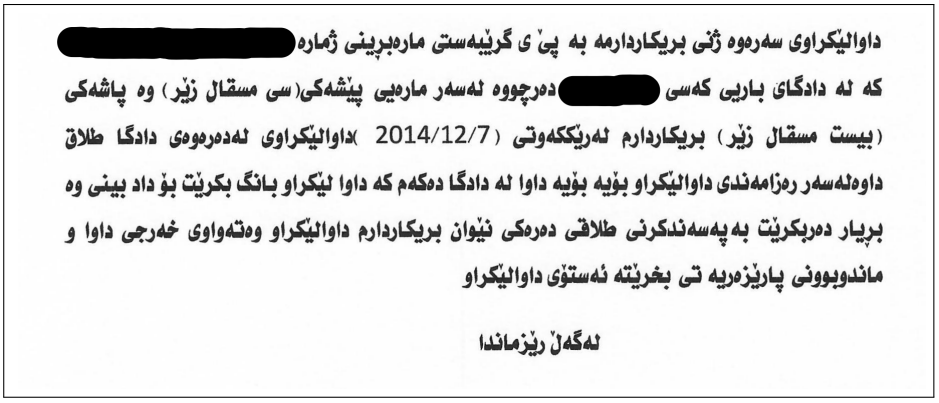


Figure 5. Actual court document image.

We OCRed the image with *kur_ara* language model, see Figure 6, and compared this with the original text file, see Figure 7.

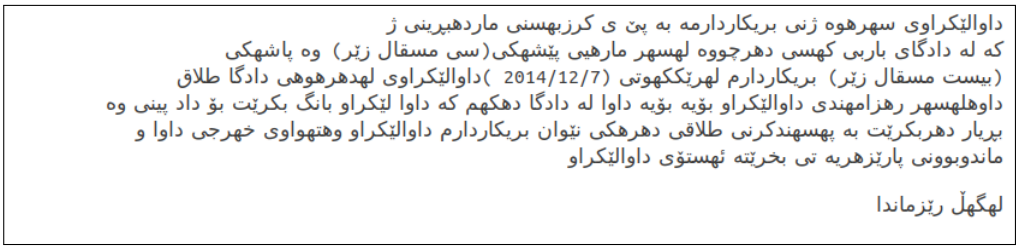


Figure 6. OCRed text of actual Court's document image.

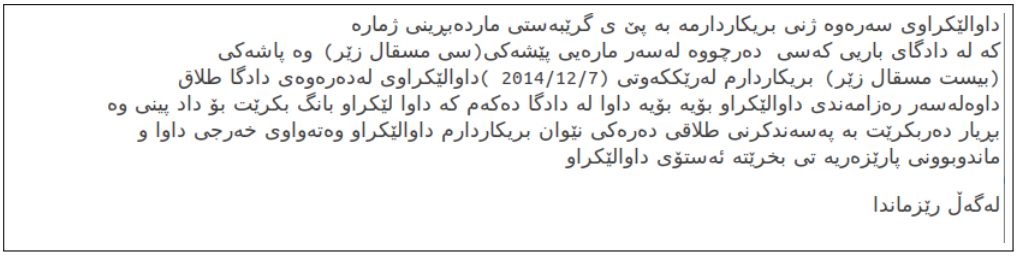


Figure 7. Original text of actual Court's document image.

The result showed an accuracy rate of 87.5%. Most of the errors that occurred are related to ZWNJ. The detailed report is shown in Figure 8. The result for the same document image after applying the post-processing method for text correction showed 97.75% accuracy.

```
ocreval Accuracy Report Version 7.0
-----------------------------------
   488  Characters
    61  Errors
 87.50%  Accuracy

     0  Reject Characters
     0  Suspect Markers
     0  False Marks
  0.00%  Characters Marked
 87.50%  Accuracy After Correction

  Ins   Subst    Del  Errors
    0      0      0       0  Marked
    5     55      1      61  Unmarked
    5     55      1      61  Total

 Count  Missed  %Right
    72       1   98.61  ASCII Spacing Characters
     8       0  100.00  ASCII Special Symbols
     7       0  100.00  ASCII Digits
   250       4   98.40  Basic Arabic
   150      55   63.33  Arabic Extended
     1       0  100.00  General Punctuation
   488      60   87.70  Total

Errors  Marked  Correct-Generated
    51       0  {ه}-{ه}
     4       0  {ماره}-{}
     1       0  { }-{}
     1       0  {}-{<\n>}
     1       0  {ن}-{ت}
     1       0  {ک}-{گ}
     1       0  {ب}-{ی}
     1       0  {ز}-{ێ}
```

Figure 8. Detailed report on actual Court's document image.

5.3. Result after Training/Fine-Tuning for Adding New Font

We Trained/Fine-tuned the kur_ara language model for adding new fonts. We used synthetic data, which includes AWN and AEN, and trained the model by utilizing 20 pages of approximately 1000 lines for each Unikurd font. We prepared those pages using a script that compiled the pages for each font based on the text at https://github.com/Shreeshrii/tesstrain-ckb/blob/master/langdata/ckb.training_text (accessed on 7 August 2021). We selected the 20 pages data from beginning of the document onward. Figure 9 shows a sample of the data that we prepared for Unikurd_Tishk font.

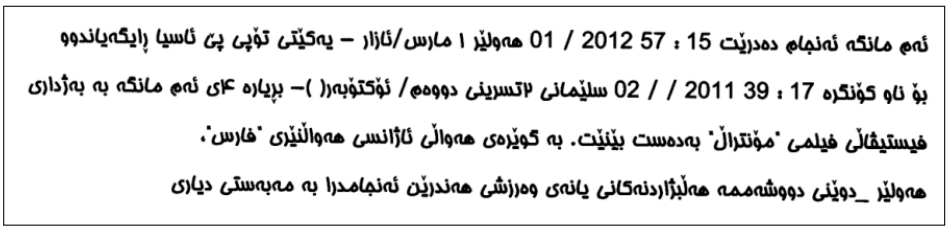


Figure 9. A Sample of Synthetic Data for Unikurd_Tishk font.

We evaluated and showed the result of the base *kur_ara* language model versus the fine-tuned version in Table 12. We started by training one font at a time and then repeated the process of training over ten fonts. The average error rate for characters before training/fine-tuning over ten fonts is 10.34%, while it is 22.61% for words. It is 1.60% for characters after training/fine-tuning, while it is 4.83% for words. That shows an improvement of 8.74 and 17.78 percentage points for characters and words, respectively. Table 12 presents a more detailed result for each font.

Table 12. The evaluation result of training/fine-tuning for adding new fonts.

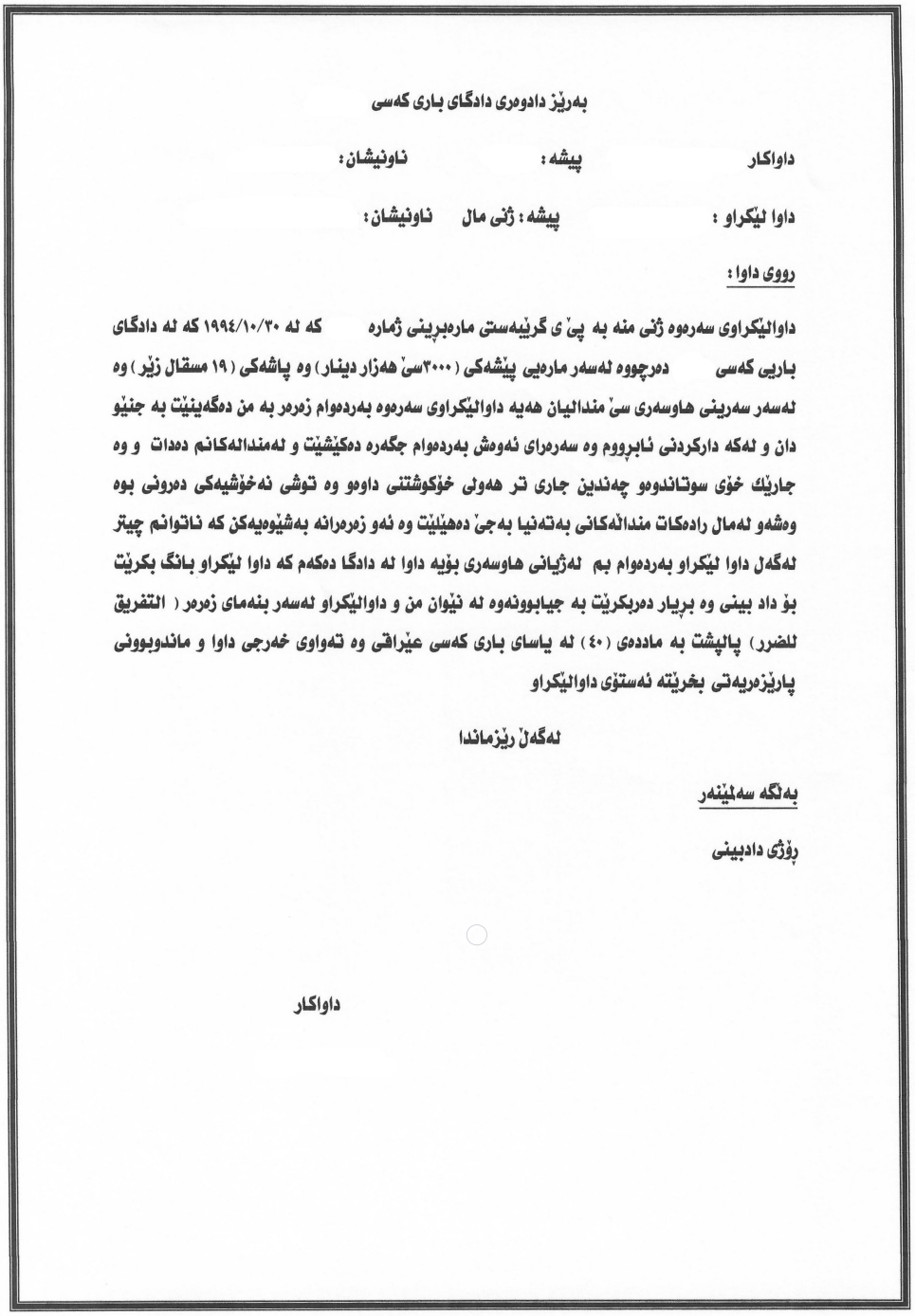
*2 —- #	Unikurd Font Name	Evaluation Before Training		Evaluation After Training	
		Char Err Rate %	Word Err Rate %	Char Err Rate %	Word Err Rate %
1	Jino	30.76	58.01	2.51	6.81
2	Web	6.15	15.51	1.35	4.78
3	Chimen	12.04	21.11	3.02	6.41
4	Hiwa	2.37	6.79	1.35	4.08
5	Goran	2.10	6.17	1.31	4.21
6	Kawe	11.94	33.65	1.35	4.70
7	Hejar	2.39	8.91	0.91	3.62
8	Hemen	11.88	30.22	2.05	6.12
9	Nali	5.52	16.83	0.73	3.52
10	Tishk	18.25	28.96	1.38	4.09
Average		10.34	22.61	1.60	4.83

We used the final *kur_ara* language model which was trained/fine tuned over ten fonts for recognizing actual court's documents images, see Figures 10 and 11.

The average accuracy rate for 11 documents reported as 95.45818182%. See details in Table 13.

Table 13. Accuracy rate for court's document images using ocreval.

#	Court's Documents	Accuracy Rate %
1	Image 1	95.88
2	Image 2	93.54
3	Image 3	93.20
4	Image 4	97.06
5	Image 5	96.34
6	Image 6	92.09
7	Image 7	97.24
8	Image 8	97.77
9	Image 9	92.83
10	Image 10	97.57
11	Image 11	96.52
Average accuracy rate		95.46



بەڕێز دادوەری دادگای باری کەسی

داواکار پیشە : ناونیشان :

داوا لێکراو : پیشە : ژنی مال ناونیشان :

رووی داوا :

داوالێکراوی سەرەوە ژنی منە بە پێ ی گرێبەستی مارەبڕینی ژمارە کە لە ١٩٩٤/١٠/٣٠ کە لە دادگای باریی کەسی دەرچووە لەسەر مارەیی پێشەکی (٣٠٠٠سێ هەزار دینار) وە پاشەکی (١٩ مسقال زێر) وە لەسەر سەرینی هاوسەری سێ منداڵیان هەیە داوالێکراوی سەرەوە بەردەوام زەرەر بە من دەگەینێت بە جنێو دان و لەکە دارکردنی ناپرووم وە سەرەرای ئەوەش بەردەوام جگەرە دەکێشێت و لەمنداڵەکانم دەدات و وە جارێك خۆی سوتاندوەو چەندین جاری تر هەوڵی خۆکوشتنی داوەو وە تۆشی نەخۆشیەکی دەرونی بوە وەشەو لەمال رادەکات منداڵەکانی بەتەنیا بەجێ دەهێلێت وە ئەو زەرەرانە بەشێوەیەکن کە ناتوانم چیتر لەگەڵ داوا لێکراو بەردەوام بم لەژیانی هاوسەری بۆیە داوا لە دادگا دەکەم کە داوا لێکراو بانگ بکرێت بۆ داد بینی وە بڕیار دەربکرێت بە جیابوونەوە لە نێوان من و داوالێکراو لەسەر بنەمای زەرەر (التفریق للضرر) پاڵپشت بە ماددەی (٤٠) لە یاسای باری کەسی عێراقی وە تەواوی خەرجی داوا و ماندووبوونی پارێزەریەتی بخرێتە ئەستۆی داوالێکراو

لەگەڵ ڕێزماندا

بەڵگە سەڵێنەر

ڕۆژی دادبینی

داواکار

Figure 10. Court document image for evaluation by final model (Image1).

بەڕێز/دادوەری دادگای باری کەسی

داواکاران ١_ پیشە :ژنی مال بریکارەکەی پاریزەر

پیشە : قوتابی

داوالێکراو/ پیشە :ژنی مال ناونیشان :

سەرەرای میراتی کۆچکردو

رووی داوا

کوری داوالێکراوی سەرەوە مێردی بریکاردارم وەباوکی داواکاری دووەم بووە بەناوی ( ) لەرێککەوتی (١٩٩٠/٣/١)لەگەل بریکاردارم هاوسەرگیریان ئە نجام داوە لەسەرمارەیی پێشەکی (٥, ١٩ ,نۆزدە مسقال ونیو زێر) پاشەکی (نیە)وەمێردی برکاردەرم لە(٢٠٠٢/١١/٩)کۆچی دوایی کردووەو و لەسەرسەرینی هاوسەرێدوو منداليان هەیەبەناوەکانی (موالید ١٩٩٣/٧/١)کەداواکاری دووەمە وکچەوە ( )موالید ١٩٩٨/٨/٥کە کچە)کە کوردن وموسوڵمان و عراقین بۆیەداوالەدگادەکەم کەداوالێکراوی سەرەوەسەرەرای میراتی کۆچکردوو بانگبکرێت بۆدادبینی و بریار دەربکرێت بەسەڵماندنی هاوسەری نێوان بریکادەرم و کوری داوالێکراوی کۆچکردوودانە پالی رەچەلەکی هەردوو منداڵەکان بۆبریکاردارم و کوری داوالێکراوی سەرەوە وەتۆمارکردنی لەتۆمارەکانی باری شارستانی بە پێی تاریخی ناماژە پێکراو لەسەرەوە وەبەشەمیراتی بریکاردەرم بە (١٠٠٠٠)دەهەزاردینار دەخەملێنم داوالێکراواری وەتەواوی خەرجی داواو ماندوبونی پاریزەرێتی بخرێەستۆی داوالێکراو.

لەگەل رێزماندا .....

داواکار(٢) لەجیاتی داواکار (١)

بریکارەکەی : پاریزەر

بە پێی بریکارنامەی گشتی ژمارە

کەلە٣٠ /٦/ (٢٠١٤) لەدادنوسی

Figure 11. Court document image for evaluation by final model (Image2).

5.4. Discussion

The research showed that using a training dataset that has a script similarity with a less-resourced language could increase the overall accuracy of the LSTM-based Tesseract engine. We also showed that two methods could be used to improve the output of the engine: using post-processing and fine-tuning the engine using different fonts. Although both provide similar improvement, the latter is more straightforward and preferable. However, using the latter approach depends on the fonts available for the target language. Highly accurate character recognition of the created model (*kur_ara*) is currently limited to documents written in ten *Unikurd* fonts, see Table 12. The approach can be applied to other Kurdish Unicode fonts by using synthetic data as well. Currently, the developed language model is restricted in recognizing only machine-typed documents.

The advantage of our approach is in the size of the data. A small amount of data could be used to train a Tesseract model that has been trained over an appropriate dataset for a well-resourced language with a similar script. The result language model of that process

provides a highly accurate output. However, this research did not intend to investigate documents that include tables and test older documents typed using mechanical type machines and other older typing tools.

6. Conclusions

The unavailability of OCR systems is an obstacle that hinders the text processing task in under-resourced languages. Tesseract is an engine that provides a proper basis to develop OCR models for various languages. However, its LSTM-based approach requires large amounts of data for their training. This research suggested a method to overcome this issue. For our experiment, we used Kurdish (Sorani), which is considered an under-resourced language. The study concluded that a small dataset for a target language along with a larger dataset for a second language that has script similarity with the target language could compensate for the requirement of a large amount of data in LSTM-based applications. The study showed that the resulted model could provide a reasonable result even in the absence of a large dataset. We also showed that, in the absence of fine-tuning, the output could be improved by post-processing. However, if fine-tuning using different fonts is possible, it could eliminate the post-processing phase.

The conclusion suggests three main areas for future work. First, the expansion of the training data set to cover more complex documents, for instance, documents with tables, newspapers, multicolumn documents, to name a few. That could be combined with an appropriate set from different types of Tesseract segmentation to extend the coverage of the adapted engine. Second, working on historical documents from the pre-digital era is paramount that could help many other sectors of Kurdish processing, particularly corpora development. Third, to investigate the applicability and efficiency of the suggested approach for OCR systems in other fields that use LSTM-based methods such as energy consumption, commercial vacancy prediction, and time-series forecasting.

Author Contributions: Conceptualization, H.H. and S.I.; Methodology, H.H. and S.I.; Software, S.I.; Validation, S.I.; Formal analysis H.H. and S.I.; Investigation, S.I. and H.H.; Resources, S.I.; Data curation, S.I.; Writing—original draft preparation, S.I.; Writing—review and editing, H.H.; Visualization, H.H. and S.I.; Supervision, H.H.; Project administration, H.H. Revision: H.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset will be publicly available on Kurdish-BLARK under the GPL upon the acceptance of the paper.

Acknowledgments: We appreciate the feedback we received from the anonymous reviewers. They were constructive, professional, valuable, and helpful. We are grateful to the reviewers for their opinion that significantly improved the paper. We are also grateful to Lesley A T Gaj for her generous assistance in proofreading the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AEN	Arabic Eastern Number
ANN	Artificial Neural network
AWN	Arabic Western Number
BLARK	Basic Language Resource Kit
DPI	Dots Per Inch
FOA	Fly Optimization Algorithm with LSTM
FOA-LSTM	Fly Optimization Algorithm with LSTM

GB	GigaByte
GIF	Graphics Interchange Format
GISMO	Geographic Information Systems and Mapping Operations
GNU	GNU's Not Unix
GUI	Graphical User Interface
GT	Ground Truth
HMM	Hidden Markov Model
ISRI	Information Science Research Institute
JOCR	Jorg Optical Character Recognition
JPEG	Joint Photographic Experts Group
KNN	K Nearest Neighbors
KRI	Kurdistan Region of Iraq
KRG	Kurdistan Regional Government
LR	language resources
LSTM	Long Short-Term Memory
LTR	Left To Right
LVQ	Learning Vector Quantization
OCR	Optical Character Recognition
OEM	OCR Engine Mode
PBM	Portable Bitmap
PGM	Portable Gray Map
PNG	Portable Network Graphics
PNM	Portable Any Map
PP	Pocket Physics
PPP	Public-Private Partnership
PSM	Page Segmentation Mode
RNN	Recurrent Neural Network
RTL	Right To Left
SDK	Software Development Kit
TIFF	Tagged Image File Format
UTF-8	8-bit Unicode Transformation Format
XML	Extensible Markup Language
ZWNJ	Zero-Width Non-Joiner

References

1. Hassani, H. BLARK for Multi-dialect Languages: Towards The Kurdish BLARK. *Lang. Resour. Eval.* **2018**, *52*, 625–644. [CrossRef]
2. Peng, L.; Zhu, Q.; Lv, S.X.; Wang, L. Effective long short-term memory with fruit fly optimization algorithm for time series forecasting. *Soft Comput.* **2020**, *24*, 15059–15079. [CrossRef]
3. Peng, L.; Wang, L.; Xia, D.; Gao, Q. Effective energy consumption forecasting using empirical wavelet transform and long short-term memory. *Energy* **2021**, *238*, 121756. [CrossRef]
4. Lee, J.; Kim, H.; Kim, H. Commercial Vacancy Prediction Using LSTM Neural Networks. *Sustainability* **2021**, *13*, 5400. [CrossRef]
5. Esmaili, K.S.; Salavati, S. Sorani Kurdish Versus Kurmanji Kurdish: An Empirical Comparison. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Sofia, Bulgaria, 4–9 August 2013; pp. 300–305.
6. Hassani, H.; Medjedovic, D. Automatic Kurdish Dialects Identification. *Comput. Sci. Inf. Technol.* **2016**, *6*, 61–78.
7. Esmaili, K.S. Challenges In Kurdish Text Processing. *arXiv* **2012**, arXiv:1212.0074.
8. Ahmadi, S. Why Does Kurdish Language Processing Matter? 2019. Available online: https://sinaahmadi.github.io/posts/why-kurdish-language-processing-matters.html (accessed on 2 September 2021).
9. Marouf, M. Kurdish Academia Journal NO. 16. 2015. Available online: https://govkrd.b-cdn.net/OtherEntities/Kurdish%20Academy/Kurdish/%D8%A8%DA%B5%D8%A7%D9%88%DA%A9%D8%B1%D8%A7%D9%88%DB%95%DA%A9%D8%A7%D9%86/%DA%AF%DB%86%DA%A4%D8%A7%D8%B1%DB%8C%20%D9%8A%D9%94%DB%95%DA%A9%D8%A7%D8%AF%DB%8C%D9%85%DB%8C%D8%A7/Govari%20Ekadimi%2016.pdf (accessed on 5 August 2021)
10. Hashemi, D. Kurdish Orthography. 2016. Available online: http://yageyziman.com/Renusi_Kurdi.htm (accessed on 2 September 2021).
11. Tafti, A.P.; Baghaie, A.; Assefi, M.; Arabnia, H.R.; Yu, Z.; Peissig, P. OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, Furthermore, Transym. In *Advances in Visual Computing. ISVC 2016*; Lecture Notes in Computer Science; Bebis, G., Ed.; Springer: Cham, Switzerland, 2016; Volume 10072.
12. Mithe, R.; Indalkar, S.; Divekar, N. Optical Character Recognition. *Int. J. Recent Technol. Eng.* **2013**, *2*, 72–75.

13. Herbert, H. *The History of OCR, Optical Character Recognition*; Recognition Technologies Users Association: Bennington County, VT, USA, 1982.
14. Dhavale, S.V. *Advanced Image-Based Spam Detection and Filtering Techniques*; IGI Global: Hershey, PA, USA, 2017.
15. d'Albe, E.F. On A Type-reading Optophone. *Proc. R. Soc. Lond. Ser. A Contain. Pap. A Math. Phys. Character* **1914**, *90*, 373–375.
16. Yaseen, R.; Hassani, H. Kurdish Optical Character Recognition. *UKH J. Sci. Eng.* **2018**, *2*, 18–27. [CrossRef]
17. Gamera. The Gamera Project. Available online: https://gamera.informatik.hsnr.de/ (accessed on 7 August 2021)
18. Radhiah, A.; Machbub, C.; Hidayat, E.M.I.; Prihatmanto, A.S. Printed Arabic Letter Recognition Based On Image. In Proceedings of the 2018 International Conference on Signals and Systems (ICSigSys), Bali, Indonesia, 1–3 May 2018; pp. 86–91.
19. Nashwan, F.; Rashwan, M.A.; Al-Barhamtoshy, H.M.; Abdou, S.M.; Moussa, A.M. A Holistic Technique For An Arabic OCR System. *J. Imaging* **2018**, *4*, 6. [CrossRef]
20. Husnain, M.; Saad Missen, M.M.; Mumtaz, S.; Jhanidr, M.Z.; Coustaty, M.; Muzzamil Luqman, M.; Ogier, J.M.; Sang Choi, G. Recognition of Urdu Handwritten Characters Using Convolutional Neural Network. *Appl. Sci.* **2019**, *9*, 2758. [CrossRef]
21. Naz, S.; Hayat, K.; Razzak, M.I.; Anwar, M.W.; Madani, S.A.; Khan, S.U. The Optical Character Recognition Of Urdu-like Cursive Scripts. *Pattern Recognit.* **2014**, *47*, 1229–1248. [CrossRef]
22. Izakian, H.; Monadjemi, S.; Ladani, B.T.; Zamanifar, K. Multi-font Farsi/Arabic Isolated Character Recognition Using Chain Codes. *World Acad. Sci. Eng. Technol.* **2008**, *43*, 67–70.
23. Jelodar, M.S.; Fadaeieslam, M.J.; Mozayani, N.; Fazeli, M. A Persian OCR System Using Morphological Operators. In Proceedings of the World Academy of Scienc, Engineering and Technology, Istanbul, Turkey, 25–27 February 2005; Volume 2, pp. 137–140.
24. Smith, R. An Overview Of The Tesseract OCR Engine. In Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, 23–26 September 2007; Volume 2, pp. 629–633.
25. Smith, R. *Motivation and History of the Tesseract OCR Engine*; Google Inc.: Menlo Park, CA, USA, 2020.
26. Dalitz, C. A Tutorial Introduction to the Gamera Framework. 2009. Available online: https://gamera.informatik.hsnr.de/docs/gamera-tutorial.pdf (accessed on 7 August 2021).
27. Genzel, D.; Ashok Popat, D.N. Paper to Digital in 200 Languages. 2015. Available online: https://ai.googleblog.com/2015/05/paper-to-digital-in-200-languages.html (accessed on 5 August 2021).
28. Google Drive Help. Convert PDF and Photo Files to Text. 2020. Available online: https://support.google.com/drive/answer/176692?hl=en&co=GENIE.Platform%3DDesktop (accessed on 5 August 2021).
29. Dhiman, S.; Singh, A. Tesseract vs. GOCR A Comparative Study. *Int. J. Recent Technol. Eng.* **2013**, *2*, 80.
30. Jain, P.; Taneja, K.; Taneja, H. Which OCR toolset is good and why: A comparative study. *Kuwait J. Sci.* **2021**, *48*. [CrossRef]
31. Zarro, R.D.; Anwer, M.A. Recognition-based Online Kurdish Character Recognition Using Hidden Markov Model Furthermore, Harmony Search. *Eng. Sci. Technol. Int. J.* **2017**, *20*, 783–794.
32. Ahmed, R.M. Kurdish Handwritten Character Recognition Using Deep Learning Techniques. Master's Thesis, University of Kurdistan Hewlêr, Erbil, Iraq, 2019.
33. Sinha, A.; Jenckel, M.; Bukhari, S.S.; Dengel, A. Unsupervised OCR Model Evaluation Using GAN. In Proceedings of the 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, NSW, Australia, 20–25 September 2019; pp. 1256–1261.
34. Martínek, J.; Lenc, L.; Král, P. Building an efficient OCR system for historical documents with little training data. *Neural Comput. Appl.* **2020**, *32*, 17209–17227. [CrossRef]
35. Hula, J.; Mojžíšek, D.; Adamczyk, D.; Čech, R. Acquiring Custom OCR System with Minimal Manual Annotation. In Proceedings of the 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2020; pp. 231–236.
36. Kiss, M.; Benes, K.; Hradis, M. AT-ST: Self-Training Adaptation Strategy for OCR in Domains with Limited Transcriptions. *arXiv* **2021**, arXiv:2104.13037.
37. Google. Tesseract Documentation. 2020. Available online: https://tesseract-ocr.github.io/ (accessed on 5 August 2021).
38. Patki, N.; Wedge, R.; Veeramachaneni, K. The Synthetic Data Vault. In Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, Canada, 17–19 October 2016; pp. 399–410.
39. Santos, E.A. OCR Evaluation Tools for the 21st Century. In Proceedings of the 3rd Workshop on the Use of Computational Methods in the Study of Endangered Languages Volume 1 (Papers), 3rd Workshop on Computational Methods for Endangered Languages, Honolulu, HI, USA, 26–27 February 2019; pp. 23–27.