Low-resource heuristics for Bahnaric Optical Character Recognition improvement

Phat Tran[1], Hung Trinh[2], Phuoc Pham[3] and Tho Quan[4]

Abstract—**The Bahnar people are an indigenous ethnic minority living in Vietnam. Research on them is still in its infancy and is quite limited due to the lack of information and available data, particularly about their language. Therefore, this study was conducted to find an effective solution in accessing Bahnar language to digitize paper-to-electronic documents from images or scans. Currently, this research focuses on an approach to applying Optical Character Recognition (OCR) technology for recognizing characters in images and then using heuristics for post-processing to enhance and improve accuracy.**

***Index Terms*—Bahnar language, OCR, text recognition, language modeling, heuristic**

1 Introduction

According to a report by the General Statistics Office of Vietnam in 2019, the population of Bahnar is 286.910 people, accounting for about 3% of the total population of the country [1]. In the context of modern life, preserving and promoting the cultural identity of ethnic minorities is an urgent task. The rich morphological nature of Bahnar but few data resources make language modeling and spelling correction a difficult task. For Bahnar language, correcting spelling errors is a prerequisite because most data sources are from raw text that is noisy and has many typos. Therefore, digitizing those unprocessed texts is crucial and required in order to produce some of the first data sets pertaining to Bahnar language. This is a necessary background to have more deep research about this low-resource language, which is our research target.

1.1 Motivation

The goal of this study is to create more corpus repositories using Bahnar language data sources. These repositories will be used for data analysis, automatic error correction, and the creation and training of language models. Utilize the gathered data set to analyze the affecting variables and choose the best deep learning techniques in order to achieve high accuracy for the mistake repair support and language model building system. From there, the language model and automatic error correction for Bahnar language are provided at the character level.

We attempt to integrate contemporary scientific advances with Bahnar language processing. It is therefore feasible to understand the language model of Bahnar and be able to enter words, sentences or paragraphs for automatic error correction assistance. This may aid in preserving the languages and writing systems of ethnic minorities.

1.2 Challenges

The trend of integration is giving rise to the risk of decline in the mother tongue of many ethnic minorities. The preservation and promotion of ethnic minority languages and scripts is essential to preserve cultural identities and realize equal rights among ethnic groups [2]. Some ethnic minority languages such as Bahnar have not been studied much. In order to achieve the goal to preserve and promote ethnic minority languages and scripts in order to preserve their cultural identity, it is urgent to restore writing for ethnic minority languages.

In recent years, machine learning (ML)/deep learning (DL) techniques have made great progress and resulted in extremely good performance. However, these methods are only extensively researched for languages with high levels of resources, which makes it impossible to apply on low-resource languages. This is a tremendous challenge for researchers and communities to solve on the path to applying

[1] Phat Tran is with the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam.
[2] Hung Trinh is with the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam.
[3] Phuoc Pham is with the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam.
[4] Tho Quan is with the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam.

natural language processing (NLP) techniques on ethnic minority languages.

Additionally, reading and processing data in tabular form is crucial for Bahnar language processing at the lexical level, since this is how most dictionary data is often written (usually 1 key column is the keyword and 1 value column is its meaning). Dealing with tabular data proved to be a significant obstacle for us as we worked on this study. Tables are used to deliver important information to the reader in a systematic manner, making table detection a critical step in many document analysis applications. Due to the many table layouts and encodings, it is a challenging problem [3].

1.3 Quick discussion about methods to handle the problem

Because all the tables that appear in the dataset have separators, we approach the table extraction problem with traditional image processing using OpenCV. More specifically, based on the characteristics of the vertical and horizontal edges of the table to calculate and find the contours containing the cells of the table.

Regarding the solution, we will determine the line equation of the horizontal and vertical edges, then calculate the coordinates of the intersection of these lines to find the coordinates of the 4 corners of the table and cells.

After the cells are identified, the next work involves text recognition problems using cell-by-cell OCR.

Finally, with the detected character string, we perform an additional post-processing step to improve accuracy.

1.4 Contribution

The contribution of our paper is the introduction of an approach to applying OCR techniques and heuristic procedure on Bahnar language - a low-resource language, and our experiments on this language is a valuable demonstration for further research on other undeveloped languages, especially minority languages.

Besides, with the results of our work, we can digitize documents in Bahnar language to provide the community with some first datasets for this language, which may shed light on various related research in the future.

2 Related works

For a long time, numerous methods to identify and fix errors have been offered in work that deals with texts extracted using OCR.

The majority of current work on OCR post-processing was completed as part of contests held in 2017 and 2019 by ICDAR (International Conference on Document Analysis and Recognition). The competitions comprise of two tasks, error detection and error correction, with evaluation sets for English and French language. During these two competitions, the majority of the participating teams used machine learning (ML)/deep learning (DL) approaches [4][5][6][7] which resulted in extremely good performance. However, those approaches are just well-studied for high-resource languages, which makes it impossible to apply on Bahnar language - a lower-resourced language.

Fortunately, it is encouraging that there were also simpler, dictionary-based approaches among the good performers, achieving e.g. a 13% character error rate (CER) reduction for English monographs and a 23% CER reduction for French monographs (with relatively low initial CERs of 1-4%). Besides, spelling correction is used in traditional statistical and/or rule-based methods to increase word accuracy from 80% to 90%, producing one-digit word error rate (WER) [8][9][10]. This type of results inspired us to pursue an approach along the same lines that is a combination between dictionary-based and statistical and/or rule-based approaches - a heuristic method for post-processing.

Another challenge occuring in the process of text extraction is table detection. Since documents normally contain a variety of table-based information with variations in appearance and layouts, an automatic table information extraction method is necessary to recognize characters in tables' cells.

There have been several prior works on identifying and extracting the tabular data inside a document. Before the development of deep learning, most table detection research relied on heuristics or metadata. TINTIN [11] used structural data to locate tables and the fields that make them up. T. Kasar et al. [12] employed an SVM classifier to determine whether or not an image region was a table region by identifying crossing horizontal and vertical lines as well as low-level features. Coinciding with the rise of Deep Learning and object detection, Azka Gilani et al. [3] was the first to suggest a Deep learning-based technique for table detection by utilizing a faster R-CNN based model. They also introduced distance-based augmentation to detect tables in an effort to increase model accuracy.

3 Data preprocessing

3.1 *Image cropping*

Since our dataset is a scanned copy of Bahnar Dialect Dictionary [13], there are several outliers existing in the images. Most of them appear in the images because we used a smartphone to scan Bahnar Dialect Dictionary whose camera cannot focus on a single page of the dictionary, which makes the images include some superfluous parts.

Therefore, before applying OCR, we need to crop the images to remove unnecessary parts and get the best possible data for the next steps. Three samples of original images and the results after cropping them are demonstrated in Figure 1 and Figure 2.

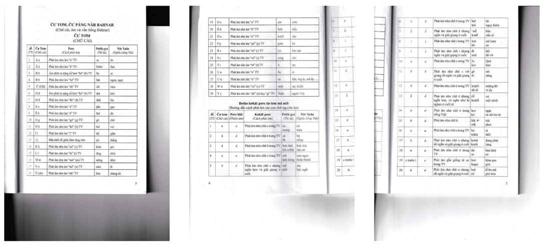


Fig. 1. Original images of Bahnar Dialect Dictionary.

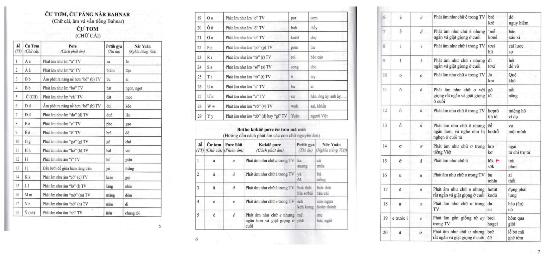


Fig. 2. Cropped Images.

3.2 *Binary image transforming*

In our approach, we convert the image into its binary form. To do this, we need a threshold to divide two values for each input pixel as 0 or 1. If the pixel value is smaller than the threshold, it is set to 0, otherwise it is set to a maximum value. Normally, in a global threshold, we often use an arbitrarily chosen value as the threshold. To avoid this selection, we use the Otsu method to be able to determine the threshold price automatically. Consider a bimodal image, which has just two different image values and its histogram with just two peaks. Between those two numbers would be a reasonable threshold. Similar to this, Otsu's approach uses the image histogram to estimate the ideal global threshold value [14].

Otsu's algorithm tries to find a threshold value (t) which minimizes the weighted within-class (background and foreground) variance given by the relation:

$$\sigma^2(t) = \omega_{bg}(t)\sigma^2_{bg}(t) + \omega_{fg}(t)\sigma^2_{fg}(t)$$

where $\omega_{bg}(t)$ and $\omega_{fg}(t)$ represents the probability of the number of pixels for each class at threshold t and σ^2 represents the variance of color values. The algorithm repeatedly looks for the threshold that reduces the within-class variance, which is determined by a weighted sum of the variances of the two classes.

If we denote P_{all} be the total count of pixel in an image, $P_{BG}(t)$ be the count of background pixels at threshold t, $P_{FG}(t)$ be the count of foreground pixels at threshold t. So the weight are given by:

$$\omega_{bg}(t) = \frac{P_{BG}(t)}{P_{all}}$$

$$\omega_{fg}(t) = \frac{P_{FG}(t)}{P_{all}}$$

The variance σ^2 can be determined by using this equation:

$$\sigma^2(t) = \frac{\sum_{i=1}^{t}(x_i - \bar{x})^2}{N-1}$$

where x_i is the value of pixel at i in the group (background or foreground), $\bar{x}$ is the means of pixel values in the group, N is the number of pixels.

3.3 *Edge detecting*

3.3.1 *Lines detecting*

First, after obtaining the binary image in the previous step, we can determine the horizontal edges and vertical edges using a simple function cv2.getStructuringElement(). Then erode and dilate these segments, we get more reasonable edges.

The method we use here is based on [15] that uses HoughLinesP with OpenCV in Python.

After there are patchwork lines that create horizontal edges, grouping these edges into main edges, we will get the necessary edge lines.

3.3.2 Image rotating

Due to the fact that scanned images may be skewed, which affects the detection of lines and cells in the next step images or pdf file. We need to straighten the image based on the vertical edge of the table if it exists. Fortunately, based on the data set, we can see that the vertical edges of the table are all parallel. Therefore, table rotation can be performed using the first vertical edge. A vertical edge has coordinates (x_1, y_1, x_2, y_2) where (x_1, y_1) and (x_2, y_2) are the coordinates of 2 points creating that line, respectively.

Having known the coordinate of 2 points of the line, we can calculate the angle between that vertical edge and Y-axis using this formula:

$$\alpha = arctan \frac{x_1 - x_2}{y_1 - y_2}$$

Then we can easily straighten the image by rotating it by an angle $-\ \alpha$ using library cv2.

3.3.3 Cell table detecting

Once we straighten the image, we detect the list of the vertical and horizontal edges of the table again.

Then we will find intersect points of lines and from there find the cells. And the way we use that is to use a system of 2 hidden equations to determine the intersection of these vertical and horizontal edges to give the position (specifically the 4-corner coordinates) of the table and cells.

By taking care of the terminal values to find the equation of the line, this method can automatically fill in areas where the edges are broken or too blurred. In addition, cells can also be guaranteed 99.99% will be fully captured, not missed and automatically sorted from top to bottom, left to right.

Suppose we have 2 line segments, respectively $A(a_1, a_2)$ and $B(b_1, b_2)$. Where a_1, a_2 is the coordinates of the 2 termination points of line A and the same for line B.

Direction vector of line A:

$$n_a = a_2 - a_1 = (x_1, x_2)$$

Normal vector of line A: $u_a = (-\ x_2, x_1)$

Direction vector of line B: $n_b = b_2 - b_1$

Direction vector of line a_1b_1: $n_p = a_1 - b_1$

The interaction point is calculated by using this equation:

$$(\frac{u_a.n_p}{u_a.n_b}).n_p + b_1$$

Having calculated all the intersections of the lines, we have a list of the coordinates of this intersection. The next issue is figuring out which four points will make up a cell. To solve this, one logic that we apply to find the 4 corners of a cell is spreading points. For each point (x_0, y_0) in the list, we proceed to search for the intersection on the right that is closest to it (x_1, y_0) and the intersection below it closest to it (x_0, y_1). We only need to check if the point (x_1, y_1) exists in the list of points or not. If it exists, then we can determine the 4 corners of the cell, otherwise, we skip and consider the next intersection. The method for locating a cell's four corners is shown in Figure 3.

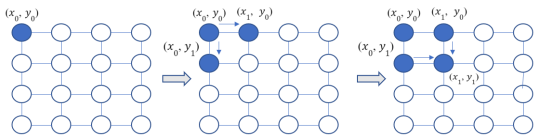


Fig. 3. Steps to spread points to find 4 corners of a cell.

3.3.4 Non-table detecting

In our dataset, the table appears in the image on a single line with no text to the left or right.

Therefore, we simply use the cv2 function to whiten the area between the two horizontal lines to remove the part of the image containing the table. From the coordinates of the top horizontal line and the bottom horizontal line, we can crop to have two images of the upper table and the below table, which are the inputs for the OCR model.

4 Heuristics for post-processing

In order to increase the accuracy, we process the output of the OCR model with the probability method.

We start by creating a Bahnar vocabulary dataset, which needs to be accurate, structured, and syntactically correct. Periods, commas, dashes, occipitals, colons, round brackets, and quotes will be ignored and replaced with spaces during preprocessing. Phrases will be broken up into individual words, and if a word only has one character, it will be deleted. However, one-letter words containing accent ˘ will be treated as two characters, for example, ‘č’ will be treated as a word that contains two characters. For some more special cases, such as ‘ ê̆’, ‘ô̆’, ‘Ê̆’ or ‘Ô̆’, they will be counted as three characters. Then, the dataset will be updated with the newly created valid terms.

The next step is to create a Bahnar dictionary from the vocabulary dataset. This dictionary is created by dividing each word in the dataset into 2, 3, and 4 character clusters. These clusters then come together to produce a new word, which is combined with the length of the word to enter into a dictionary to determine how many times it has been used.

Afterward, we use two loops to check. The first loop is to determine the starting point and the second is to determine the number of substring characters from that point. We do some general case corrections in the first loop, such as changing “]” to “l”, “Ö” and “š” (which do not exist in Bahnar alphabet) to “Č” and “č”, respectively. For each of those substrings, if they are not in the list of occurrences or have fewer occurrences than the threshold (default is 5), the error will be corrected.

Words that are included in the need to be replaced will be replaced according to the phrases with the highest probability (the highest frequency) and with the priority level from 4 characters decreasing to 2 characters.

5 Experimental results

5.1 Dataset

In this section, we will describe our dataset for Bahnar language, named Bahnar Dialect Dictionary. The dictionary is published by Gia Lai Department of Education and Training which is a good resource to learn and research into Bahnar language. This dataset contains 115 images and it includes table and non-table structures that hold a large number of words in both Bahnar language and Vietnamese. Therefore, using only normal OCR techniques is not enough to recognize special characters appearing in Bahnar language including “ƀ”, “č”, “ĕ”, “ê̆”, “ĭ”, “ñ”, “ŏ”,“ô̆”, “ơ̆”, “ŭ”, “ư̆”, “Ƀ”, “Č”, “Ĕ”, “Ê̆”, “Ĭ”, “Ñ”, “Ŏ”, “Ô̆”, “Ơ̆”, “Ŭ”, “Ư̆”. This has motivated us to use heuristic to improve the results of OCR. Besides, since there are several outliers and anomalies existing in the images, it is also crucial to perform data preprocessing on the dataset before working on it.

5.2 Tesseract

To apply OCR on our dataset, we used Tesseract - an optical character recognition engine with open-source code, this is the most popular and qualitative OCR-library up to now [16].

The main role of Tesseract in our pipeline is to recognize characters in the table and non-table structures of the images. However, it needs traineddata files which support Legacy and LSTM engines. Unfortunately, these traineddata files currently limit themselves to only high-resource languages such as French, English, Bulgarian, Vietnamese, etc. Therefore, Tesseract is unable to recognize special characters of minority languages, which motivated us to use heuristics for post-processing [17].

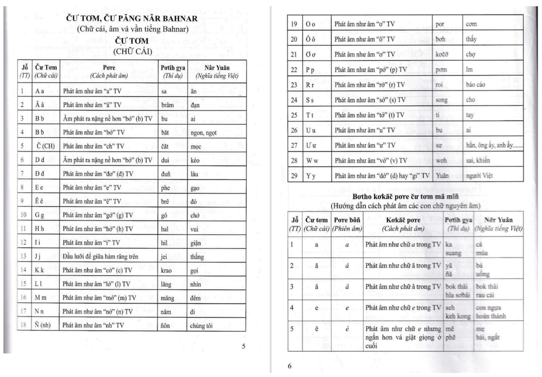
There are 14 Page Segmentation Modes (PSMs) in Tesseract. We have experimented with all of them to find out an appropriate PSM for our dataset which can significantly improve the accuracy of the OCR. PSM 6 turns out to be the best mode for our case. PSM 6 performs well when OCRing pages of simple books (e.g., a paperback novel) that follows a simplistic page structure and a single, consistent font throughout their pages. This has led to the configuration of ‘-l vie+en --oem 1 --psm 6’ in our implementation.

5.3 Experiment with language modelling

Additionally, we have conducted experiments on using language modeling to post-process the results of OCR. The model that we used was specifically developed for Bahnar language [18] which employs both Character Left to Right (L2R) Model and Character Right to Left (R2L) Model.

However, we found that this language model produced subpar outcomes. Further investigation led us to the conclusion that this model's poor performance was mostly caused by the low-resource data used for its training, which inspired us to implement heuristics in our pipeline.

Figure 4 displays two examples of input images and Figure 5 illustrates the results after applying the language model on their OCR outputs.



ČƯ TƠM, ČƯ PĂNG NÂR BAHNAR
(Chữ cái, âm và vần tiếng Bahnar)
ČỤ TƠM
(CHỮ CÁI)

Jỗ *(TT)*	**Čư Tơm** *(Chữ cái)*	**Pơre** *(Cách phát âm)*	**Pơtih gya** *(Thí dụ)*	**Nâr Yuăn** *(Nghĩa tiếng Việt)*
1	A a	Phát âm như âm “a” TV	sa	ăn
2	Â â	Phát âm như âm “â” TV	ƀrâm	đạn
3	B b	Âm phát ra nặng nề hơn “bờ” (b) TV	bu	ai
4	Ɓ ƀ	Phát âm như âm “bờ” TV	ƀăt	ngon, ngọt
5	Č (CH)	Phát âm như âm “ch” TV	čăt	mọc
6	D d	Âm phát ra nặng nề hơn “bờ” (b) TV	dui	kéo
7	Đ đ	Phát âm như âm “đơ” (đ) TV	đuñ	lâu
8	E e	Phát âm như âm “e” TV	phe	gạo
9	Ê ê	Phát âm như âm “ê” TV	ƀrê	đỏ
10	G g	Phát âm như âm “gờ” (g) TV	gô	chờ
11	H h	Phát âm như âm “hờ” (h) TV	hal	vui
12	I i	Phát âm như âm “i” TV	hil	giận
13	J j	Đầu lưỡi để giữa hàm răng trên	jei	thắng
14	K k	Phát âm như âm “cờ” (c) TV	krao	gọi
15	L l	Phát âm như âm “lờ” (l) TV	lăng	nhìn
16	M m	Phát âm như âm “mờ” (m) TV	măng	đêm
17	N n	Phát âm như âm “nờ” (n) TV	năm	đi
18	Ñ (nh)	Phát âm như âm “nh” TV	ñôn	chúng tôi
19	O o	Phát âm như âm “o” TV	por	cơm
20	Ô ô	Phát âm như âm “ô” TV	ƀơh	thấy
21	Ơ ơ	Phát âm như âm “ơ” TV	kơčŏ	chợ
22	P p	Phát âm như âm “pờ” (p) TV	pơm	lm
23	R r	Phát âm như âm “rờ” (r) TV	roi	báo cáo
24	S s	Phát âm như âm “sờ” (s) TV	song	cho
25	T t	Phát âm như âm “tờ” (t) TV	ti	tay
26	U u	Phát âm như âm “u” TV	bu	ai
27	Ư ư	Phát âm như âm “ư” TV	sư	hắn, ông ấy, anh ấy.......
28	W w	Phát âm như âm “vờ” (v) TV	wơh	sai, khiến
29	Y y	Phát âm như âm “dờ” (d) hay “gi” TV	Yuăn	người Việt

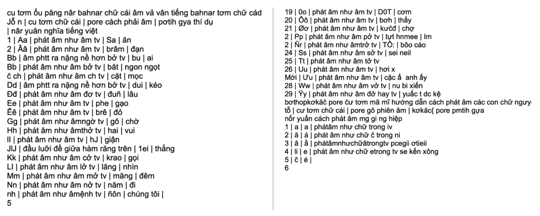
5

Bơtho kơkăč pơre čư tơm mă miñ
(Hướng dẫn cách phát âm các con chữ nguyên âm)

Jỗ *(TT)*	**Čư tơm** *(Chữ cái)*	**Pơre ƀôñ** *(Phiên âm)*	**Kơkăč pơre** *(Cách phát âm)*	**Pơtih gya** *(Thí dụ)*	**Nờr Yuăn** *(Nghĩa tiếng Việt)*
1	a	*a*	Phát âm như chữ *a* trong TV	ka suang	cá múa
2	ă	*á*	Phát âm như chữ ă trong TV	yă ñă	bà uống
3	â	*â*	Phát âm như chữ â trong TV	ƀok thâi hla sơƀâi	ƀok thâi rau cải
4	e	*e*	Phát âm như chữ *e* trong TV	seh keh kong	con ngựa hoàn thành
5	ĕ	*é*	Phát âm như chữ *e* nhưng ngắn hơn và giật giọng ở cuối	mĕ phĕ	mẹ hái, ngắt

6

Fig. 4. Input image of page 5 and 6 respectively.



```
cu tơm ốu păng nâr bahnar chữ cái âm vả vân tiếng bahnar tơm chữ cád
Jỗ n | cu tơm chữ cái | pore cách phải âm | pơtih gya thí dụ
| nâr yuăn nghĩa tiếng việt
1 | Aa | phát âm như âm tv | Sa | ăn
2 | Ââ | phát âm như âm tv | brâm | đạn
Bb | âm phtt ra nặng nề hơn bở tv | bu | ai
Bb | phát âm như âm bở tv | băt | ngon ngọt
č ch | phát âm như âm ch tv | cặt | mọc
Dd | âm phtt ra nặng nề hơn bở tv | dui | kéo
Đđ | phát âm như âm đơ tv | đuñ | lâu
Ee | phát âm như âm tv | phe | gạo
Êê | phát âm như âm tv | brê | đỏ
Gg | phát âm như âmngờ tv | gô | chờ
Hh | phát âm như âmthở tv | hai | vui
Il | phát âm như âm tv | hJ | giận
JIJ | đầu lưỡi đề giữa hàm răng trên | 1ei | thắng
Kk | phát âm như âm cở tv | krao | gọi
LI | phát âm như âm lở tv | lăng | nhìn
Mm | phát âm như âm mở tv | măng | đêm
Nn | phát âm như âm nở tv | năm | đi
nh | phát âm như âmệnh tv | ñôn | chúng tôi |
5
```

```
19 | 0o | phát âm như âm tv | D0T | cơm
20 | Ôô | phát âm như âm tv | bơh | thấy
21 | Øơ | phát âm như âm tv | kưčđ | chợ
2 | Pp | phát âm như âm pở tv | tựt hnmee | lm
2 | Ñr | phát âm như âmtrở tv | TÔ: | bôo cáo
24 | Ss | phát âm như âm sở tv | sei neil
25 | Tt | phát âm như âm tở tv
26 | Uu | phát âm như âm tv | hơi x
Mới | Ưu | phát âm như âm tv | cặc ấ  anh ấy
28 | Ww | phát âm như âm vở tv | nu bi xiến
29 | Ÿy | phát âm như âm đở hay tv | yuấc t dc kệ
bơthopkơkăč pore čư tơm mã mĩ hướng dẫn cách phát âm các con chữ nguy
tỗ | cư tơm chữ cái | pore gô phiên âm | kơkăc{ pore pmtih gựa
nốr yuấn cách phát âm mg gi ng hiệp
1 | a | a | phátâm như chữ trong iv
2 | ă | á | phát âm như chữ č trong ni
3 | â | ầ | phátâmnhưchữâtrongtv pcegii ơtieii
4 | li | e | phát âm như chữ etrong tv se kến xông
5 | č | é |
6
```

Fig. 5. OCR output of page 5 and 6 after applying language modeling respectively.

5.4 *Pipeline*

We create a pipeline as shown in Figure 6 to experiment with our dataset.

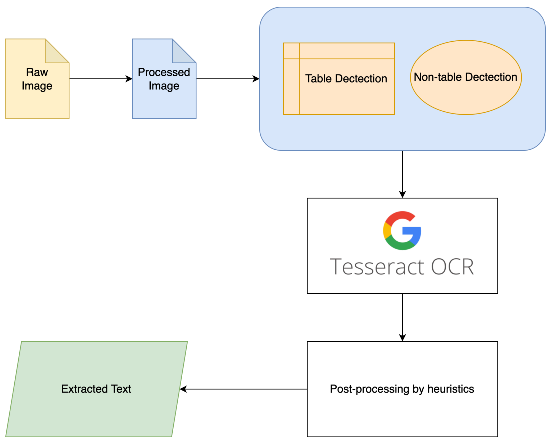


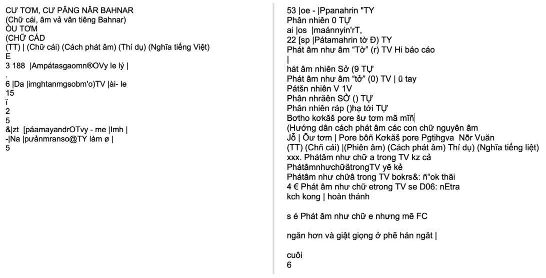
Fig. 6. An overall architecture of our approach that extracts text from digital images or scans for Bahnar language.

Figure 6 demonstrates the pipeline we used for extracting text from digital images or scans. First of all, we convert pdf scans to images, then crop them to have raw inputs. Then raw images will be preprocessed to eliminate outlier pixels as well as increase contrast between text and background image. Next, we erode, dilate, then houghline preprocessed images to find horizontal and vertical lines, which are necessary for detecting table cells and non-table parts, in more details are the upper and below the table. After OCR, text will be post-processing using heuristic, then all extracted texts from cells and outside the table will be formatted based on table structure to get the final result.

5.5 *Results*

We have obtained some satisfactory results from this approach which are shown in Figure 7.

5.5.1 *Results before applying table detection*



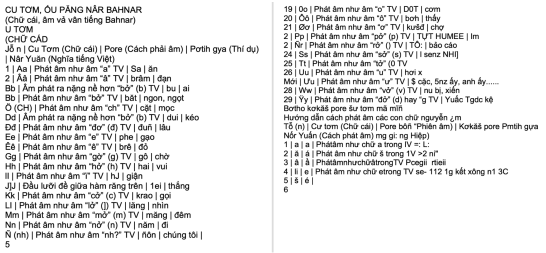
```
CƯ TƠM, CƯ PĂNG NÂR BAHNAR
(Chữ cái, âm vả vân tiêng Bahnar)
ÒU TƠM
(CHỮ CÁD
(TT) | (Chữ cái) (Cách phát âm) (Thí dụ) (Nghĩa tiếng Việt)
E
3 188  |Ampátasgaomn®OVy le lý |
.
6 |Da |imghtanmgsobm'o)TV |ài- le
15
ï
2
5
&|zt  [páamayandrOTvy - me |Imh |
-|Na |pưảnmranso@TY làm ø |
5
```

```
53 |oe - |Ppanahrin "TY
Phân nhiên 0 TỰ
ai |os  |maánnyin'rT,
22 [sp |Pátamahrin tờ Đ) TY
Phát âm như âm “Tờ” (r) TV Hi báo cáo
|
hát âm nhiên Sở (9 TỰ
Phát âm như âm “tở” (0) TV | ũ tay
Pátšn nhiên V 1V
Phân nhrăên SỞ () TỰ
Phân nhiên ráp ()hạ tới TỰ
Bơtho kơkăš pore šư tơm mã mĩñ|
(Hướng dân cách phát âm các con chữ nguyên âm
Jỗ | Ǒư tơm | Pore bôñ Kơkăš pore Pgtihgva  Nởr Vuăn
(TT) (Chñ cái) |(Phiên âm) (Cách phát âm) Thí dụ) (Nghĩa tiếng liệt)
xxx. Phátâm như chữ a trong TV kz cả
PhátâmnhưchữătrongTV yẽ kẻ
Phátâm như chữâ trong TV bokrs&: ñ“ok thãi
4 € Phát âm như chữ etrong TV se D06: nEtra
kch kong | hoàn thánh

s é Phát âm như chữ e nhưng mẽ FC

ngăn hơn và giật giọng ở phẽ hán ngăt |

cuôi
6
```

Fig. 7. OCR result of page 5 and 6 before applying table detection respectively.

We can observe that the OCR results on the cropped input are poor when there is no table detection. Particularly in the table part, the OCR result is found to be lacking and inconsistent with the input.

5.5.2 *Result after applying table detection (without heuristic)*



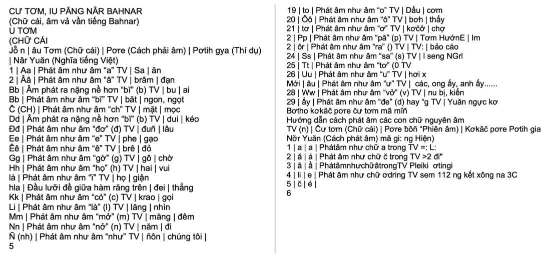
```
CU TƠM, ỐU PĂNG NÂR BAHNAR
(Chữ cái, âm vả vân tiếng Bahnar)
U TƠM
(CHỮ CÁD
Jỗ n | Cu Tơm (Chữ cái) | Pore (Cách phải âm) | Pơtih gya (Thí dụ)
| Nâr Yuăn (Nghĩa tiếng Việt)
1 | Aa | Phát âm như âm “a” TV | Sa | ăn
2 | Ââ | Phát âm như âm “â” TV | brâm | đạn
Bb | Âm phát ra nặng nề hơn “bở” (b) TV | bu | ai
Bb | Phát âm như âm “bở” TV | băt | ngon, ngọt
Ö (CH) | Phát âm như âm “ch” TV | cặt | mọc
Dd | Âm phát ra nặng nề hơn “bở” (b) TV | dui | kéo
Đđ | Phát âm như âm “đơ” (đ) TV | đuñ | lâu
Ee | Phát âm như âm “e” TV | phe | gạo
Êê | Phát âm như âm “ê” TV | brê | đỏ
Gg | Phát âm như âm “gờ” (g) TV | gô | chờ
Hh | Phát âm như âm “hở” (h) TV | hai | vui
Il | Phát âm như âm “ï” TV | hJ | giận
J]J | Đầu lưỡi đề giữa hàm răng trên | 1ei | thắng
Kk | Phát âm như âm “cở” (c) TV | krao | gọi
LI | Phát âm như âm “lở” (]) TV | lăng | nhìn
Mm | Phát âm như âm “mở” (m) TV | măng | đêm
Nn | Phát âm như âm “nở” (n) TV | năm | đi
Ñ (nh) | Phát âm như âm “nh?” TV | ñôn | chúng tôi |
5
```

```
19 | 0o | Phát âm như âm “o” TV | D0T | cơm
20 | Ôô | Phát âm như âm “ô” TV | bơh | thấy
21 | Øơ | Phát âm như âm “ơ” TV | kušđ | chợ
2 | Pp | Phát âm như âm “pở” (p) TV | TỰT HUMEE | lm
2 | Ñr | Phát âm như âm “rở” () TV | TÔ: | bảo cáo
24 | Ss | Phát âm như âm “sở” (s) TV | I senz NHI]
25 | Tt | Phát âm như âm “tở” (0 TV
26 | Uu | Phát âm như âm “u” TV | hơi x
Mới | Ưu | Phát âm như âm “ư” TV | $ cặc, 5nz ấy, anh ấy......
28 | Ww | Phát âm như âm “vở” (v) TV | nu bị, xiến
29 | Ÿy | Phát âm như âm “đở” (d) hay “g TV | Yuấc Tgdc kệ
Bơtho kơkăš pore šư tơm mã mĩñ
Hướng dẫn cách phát âm các con chữ nguyễn ¿m
Tỗ (n) | Cư tơm (Chữ cái) | Pore bôñ “Phiên âm) | Kơkăš pore Pmtih gựa
Nốr Yuấn (Cách phát âm) mg gi: ng Hiệp)
1 | a | a | Phátâm như chữ a trong IV =: L:
2 | ă | á | Phát âm như chữ š trong 1V >2 ni"
3 | â | ầ | PhátâmnhưchữâtrongTV Pcegii  rtieii
4 | li | e | Phát âm như chữ etrong TV se- 112 1g kết xông n1 3C
5 | š | é |
6
```

Fig. 8. OCR result of page 5 and 6 after applying table detection.

From results in the post-processing step (between Figure 7 and Figure 8), we are able to observe that the table extraction is better with the table structure being kept and text result consistent with the input.

5.5.3 *Results after heuristic*



CƯ TƠM, IU PĂNG NÂR BAHNAR
(Chữ cái, âm vả vần tiếng Bahnar)
U TƠM
(CHỮ CÁI
Jỗ n | âu Tơm (Chữ cái) | Pơre (Cách phải âm) | Pơtih gya (Thí dụ)
| Nâr Yuăn (Nghĩa tiếng Việt)
1 | Aa | Phát âm như âm "a" TV | Sa | ăn
2 | Ââ | Phát âm như âm "â" TV | brâm | đạn
Bb | Âm phát ra nặng nề hơn "bĭ" (b) TV | bu | ai
Bb | Phát âm như âm "bĭ" TV | băt | ngon, ngọt
Č (CH) | Phát âm như âm "ch" TV | mặt | mọc
Dd | Âm phát ra nặng nề hơn "bĭ" (b) TV | dui | kéo
Đđ | Phát âm như âm "đơ" (đ) TV | đuñ | lâu
Ee | Phát âm như âm "e" TV | phe | gạo
Êê | Phát âm như âm "ê" TV | brê | đỏ
Gg | Phát âm như âm "gờ" (g) TV | gô | chờ
Hh | Phát âm như âm "họ" (h) TV | hai | vui
là | Phát âm như âm "ĭ" TV | họ | giận
hla | Đầu lưỡi đề giữa hàm răng trên | đei | thắng
Kk | Phát âm như âm "có" (c) TV | krao | gọi
Li | Phát âm như âm "là" (l) TV | lăng | nhìn
Mm | Phát âm như âm "mở" (m) TV | măng | đêm
Nn | Phát âm như âm "nở" (n) TV | năm | đi
Ñ (nh) | Phát âm như âm "như" TV | ñôn | chúng tôi |
5

19 | to | Phát âm như âm "o" TV | Dấu | cơm
20 | Ôô | Phát âm như âm "ô" TV | bơh | thấy
21 | tơ | Phát âm như âm "ơ" TV | kơčŏ | chợ
2 | Pp | Phát âm như âm "pă" (p) TV | Tơm HướnE | Im
2 | ôr | Phát âm như âm "ra" () TV | TV: | bảo cáo
24 | Ss | Phát âm như âm "sa" (s) TV | I seng NGrl
25 | Tt | Phát âm như âm "tơ" (0 TV
26 | Uu | Phát âm như âm "u" TV | hơi x
Mới | âu | Phát âm như âm "ư" TV | các, ong ấy, anh ấy......
28 | Ww | Phát âm như âm "vở" (v) TV | nu bị, kiến
29 | ấy | Phát âm như âm "đe" (d) hay "g TV | Yuăn ngực kơ
Bơtho kơkăč pơre čư tơm mã mĭñ
Hướng dẫn cách phát âm các con chữ nguyên âm
TV (n) | Čư tơm (Chữ cái) | Pơre ƀôñ "Phiên âm) | Kơkăč pơre Pơtih gia
Nŏr Yuăn (Cách phát âm) mă gi: ng Hiện)
1 | a | a | Phátâm như chữ a trong TV =: L:
2 | ă | á | Phát âm như chữ č trong TV >2 đi"
3 | â | ầ | PhátâmnhưchữâtrongTV Pleiki ơtingi
4 | li | e | Phát âm như chữ ơdring TV sem 112 ng kết xông na 3C
5 | č | é |
6

Fig. 9. OCR result of page 5 and 6 after heuristic.

We evaluate the results in the post-processing step (between Figure 8 and Figure 9) by randomly choosing 5 pages from our dataset to create a sample for comparison. There are a total of 969 words from those pages. Prior to the heuristic, there were 706 correct words, which is fewer than the 768 correct words after applying the heuristic.

	Before heuristic	After heuristic
Validation accuracy	0.7286	0.7926

Tab. 1. Validation accuracy before and after applying heuristic.

By observing Table 1, we can see with clarity that heuristic post-processing produced excellent results, increasing the accuracy from roughly 72.86% to 79.26%.

Table 2 illustrates some of the common cases that were corrected after applying heuristic for post-processing.

Ground truth	Before heuristic	After heuristic
kơkăč	kơkăš	kơkăč
sŏk	sốk	sŏk
kơčŏ	kơšđ	kơčŏ
ƀôñ	bôñ	ƀôñ
phŏk	phõk	phŏk
tơxĭ	tơxï	tơxĭ
hơtŭt	hơtũt	hơtŭt
poñan	poñan	poñan
pơđôr	pơđØr	pơđôr
Nŏr	Nốr	Nŏr

Tab. 2. Common cases before and after applying heuristic.

6 Conclusion

In this work, we have come up with an approach to Bahnar text processing and recognition from photos or scans. Our research opens up great potential in the application of modern techniques to the processing of minority languages belonging to ethnic minorities. In particular, this research focuses on ethnic minorities in Vietnam. The aim is to maintain and protect their language, and at the same time open up research approaches to their culture as well as their national identity. However, in order to accomplish this, a lot of work, time, and dedication must be put in to increase accuracy and, more crucially, to add support for additional languages, such as Rade, Sedang, etc.

In the future, we will focus on improving our solution by extending the Bahnar vocabulary dictionary. Larger data sets will obviously lead to better heuristic results. Besides, we are also testing methods that use grammar rules instead of using probability. Furthermore, we will research and experiment modern language processing models to be able to process Bahnar at the syntactic level in the near future. Not only that, with the current promising results, the application of this solution to other languages of ethnic minorities is completely feasible and simple to implement. Therefore, we hope to be able to find a complete and highly applicable direction in practice in this field.

Acknowledgment

We acknowledge the support of time and facilities from HCMUT, VNUHCM[5] for this research. In addition, we would like to express our sincere gratitude and respect to Mr. Nguyen

[5] Vietnam National University Ho Chi Minh City (VNUHCM), Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam.

Quang Duc for his assistance throughout this study.

References

[1] General Statistics Office of Vietnam, "Completed Results of the 2019 Viet Nam Population and Housing Census," in Vietnam, 2019, pp. 44.

[2] N. Anh. (2022, Dec 13). Preserving ethnic minority scripts: Need long-term and synchronous solutions. [Online]. Available: https://www.bienphong.com.vn/bao-ton-chu-viet-dan-toc-thieu-so-can-nhung-giai-phap-dong-bo-dai-hoi-post456888.html

[3] A. Gilani, S. R. Qasim, I. Malik and F. Shafait, "Table Detection Using Deep Learning," in 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 2017, pp. 771-776, DOI. 10.1109/ICDAR.2017.131.

[4] R. Dong and D. Smith, "Multi-Input Attention for Unsupervised OCR Correction," in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 2363–2372.

[5] V. Nastase and J. Hitschler, "Correction of OCR Word Segmentation Errors in Articles from the ACL Collection through Neural Machine Translation Methods," in Proceedings of the Eleventh International Conference on LREC, Miyazaki, Japan, 2018, pp. 706–711.

[6] L. Lyu, M. Koutraki, M. Krickl, B. Fetahu. (2021, Jan 01). Neural OCR Post-Hoc Correction of Historical Corpora. Presented at Transactions of the Association for Computational Linguistics. [Online]. Available: https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00379/100788/Neural-OCR-Post-Hoc-Correction-of-Historical

[7] K. Mokhtar, S. Bukhari, A. Dengel, "OCR Error Correction: State-of-the-art vs An NMT Based Approach," in Proceedings of the 13th IAPR International Workshop on Document Analysis Systems, Vienna, Austria, 2018, pp. 429-434.

[8] X. Tong and D. A. Evans, "A Statistical Approach to Automatic OCR Error Correction in Context," in 4th Workshop on Very Large Corpora, Copenhagen, Denmark, 1996, pp. 88-100.

[9] Y. Bassil and M. Alwani, "OCR Post-Processing Error Correction Algorithm Using Google's Online Spelling Correction," in Journal of Emerging Trends in Computing and Information Sciences, vol. 3, no. 1, Jan. 2012.

[10] P. Thompson, J. McNaught, S. Ananiadou, "Customised OCR Correction for Historical Medical Text," in Digital Heritage International Congress, Sep. 2015, DOI. 10.1109/DigitalHeritage.2015.7413829.

[11] P. Pyreddy and W. B. Croft, "Tintin: A system for retrieval in text tables," in Proceedings of the second ACM international conference on Digital libraries, Philadelphia, USA, 1997, pp. 193–200.

[12] T. Kasar, P. Barlas, S. Adam, C. Chatelain, and T. Paquet, "Learning to detect tables in scanned document images using line information," in 12th ICDAR, Washington, DC, USA, 2013, pp. 1185–1189.

[13] Y Jil, H'Mer, D. V. Hai, D. V. Khoa, "Bahnar Dialect Dictionary," 1st ed. Gia Lai, Vietnam: Gia Lai Department of Education and Training, 2018.

[14] J. Yousefi, "Image Binarization using Otsu Thresholding Algorithm," University of Guelph, Ontario, Canada, 2011, pp. 1-4, DOI. 10.13140/RG.2.1.4758.9284.

[15] C. Cherun, N. Funabiki, Y. Huo, M. Mentari, K. Suga, T. Toshida, "A Proposal of Printed Table Digitization Algorithm with Image Processing," in Algorithms, vol. 15, Dec. 2022, DOI. 10.3390/a15120471.

[16] R. Smith, "An Overview of the Tesseract OCR Engine," in Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, 2007, pp. 629-633, DOI. 10.1109/ICDAR.2007.4376991.

[17] D. S. Vargas, L. L. de Oliveira, V. P. Moreira, G. T. Bazzo, G. A. Lorentz, "sOCRates – a post-OCR text correction method," in Simpósio Brasileiro de Banco de Dados, Rio de Janeiro, Brazil, 2021.

[18] N. D. Linh, "Building language model for automatically correcting Bahnar language," M.S. Thesis, Thu Dau Mot Univ., Binh Duong, Vietnam, 2021.