An Android-Based Useful Text Extraction Framework Using Image and Natural Language Processing

Rafsanjany Kushol, Imamul Ahsan, and Md. Nishat Raihan

***Abstract*—With the rapid advancement of the mobile technologies, we now use mobile applications in every aspect of our life. At present when we find some useful information from business cards, newspapers, books, flyers and so on we do not usually note them rather we capture the image using mobile devices. However, the texts obtained from these images need to be processed further to make our task unequivocal. In this paper, we present an android application framework that can process the texts obtained from an image to get contact information from business cards as well as event information from magazines, posters or flyers. Google Cloud Vision API is used to retrieve the text from the captured image and OpenNLP to extract useful information from the obtained text. The experimental results show that our application is effective and efficient in terms of accuracy as well as processing time.**

***Index Terms*—Android, optical character recognition, natural language processing, Google Cloud Vision API, OpenNLP.**

I. Introduction

Nowadays people use the camera of their mobile phones every now and then to capture important text from various types of documents such as business card, posters, flyer etc. Furthermore, people want to keep a record of contact detail from business cards and important event information in their calendar. As the information is in image format so it becomes very troublesome for the users to manually input the data. Our main objective is to enhance the camera capability of mobile phones so that it can detect the useful text information from the image as well as store necessary data for future use. In this context, we have implemented an android application framework which can process these images to retrieve useful information of our day to day lives. Examples of the captured image taken from a different source are shown in Fig. 1. Our application can analyze the text from the captured image and automatically sync the obtained information with contacts or calendar of the android device.

The iplementation of our application has been quite challenging because all of the computational tasks are done independently by the Android mobile device. Although at present the processing capability of the mobile device has increased to some extent still it is challenging if the application is very complex. Our application mainly consists of two processing steps which are image processing and natural language processing. The image processing requires Optical Character Recognition (OCR), performed using Google Cloud Vision API [1] whereas the Natural Language Processing (NLP) requires a toolkit called Named Entity Recognition (NER), which is performed using Apache OpenNLP API [2]. We also have to consider the performance measure and accuracy while developing the application. It is challenging to perform OCR in real time images because the text in these images was variant and unstructured. While performing the OCR on business cards, posters, magazines or flyers we have found that the text fonts are not regular as in the formal documents. Different style fonts and artistic backgrounds are used in these images and for this reason, OCR accuracy is very difficult to obtain. After performing the OCR, our next challenge is to find useful information like name, email, location etc. from the obtained text using NLP. This is a further challenge to us because the obtained text from OCR is totally unstructured data. It is very tedious to extract information from the text which does not follow any grammatical pattern and not organized. To extract the information accurately we have to create a very large training dataset. Finally, the overall performance and accuracy of our application are utterly satisfactory.

Manuscript received February 5, 2018; revised April 20, 2018.

The authors are with the Computer Science and Engineering Department, Islamic University of Technology (IUT), Bangladesh (e-mail: kushol@iut-dhaka.edu, imamulahsan@iut-dhaka.edu, nishatraihan@iut-dhaka.edu).

II. Literature Review

In the past few years, there has been some research to scan and digitize business cards after capturing its image by the camera of the device. This has led to few research works [3]-[5] in which only OCR was emphasized. Vuong and Do [3] implemented a multilingual name card reader that can detect names from business cards. The authors used Tesseract OCR engine to recognize characters and simple pattern matching techniques were applied to detect the required information. Bhaskar *et al.* [4] emphasized on the MATLAB implementation of OCR and compared the outcome with Tesseract OCR output. On the other hand, Sharma and Fujii [5] performed OCR on a preprocessed image to detect text and used regular expressions to extract the useful information. These works were fully focused on extracting information from business cards. Most of the works relevant to this topic implement the image processing part by uploading the image to a cloud server then process it and send the result back to the device such as Prisma App. The technique of offloading the data saves processing and battery power but also reduces the usability as it requires a continuous online network connection. However, the computational capability has improved dynamically in recent years and so the device can itself perform the computation in feasible time which is required for the image processing part. A detail experiment and condition is analyzed by Kumar and Lu [6] about

DOI: 10.7763/IJCTE.2018.V10.1203

offloading data to a cloud server. After considering all the parameters and experimental result, we have implemented all the calculation of our application on the android device without offloading any data to the cloud server.

(a) (b) (c) (d)

Fig. 1. Captured image from different sources (a) business card, (b) poster, (c) magazine, and (d) flyer.

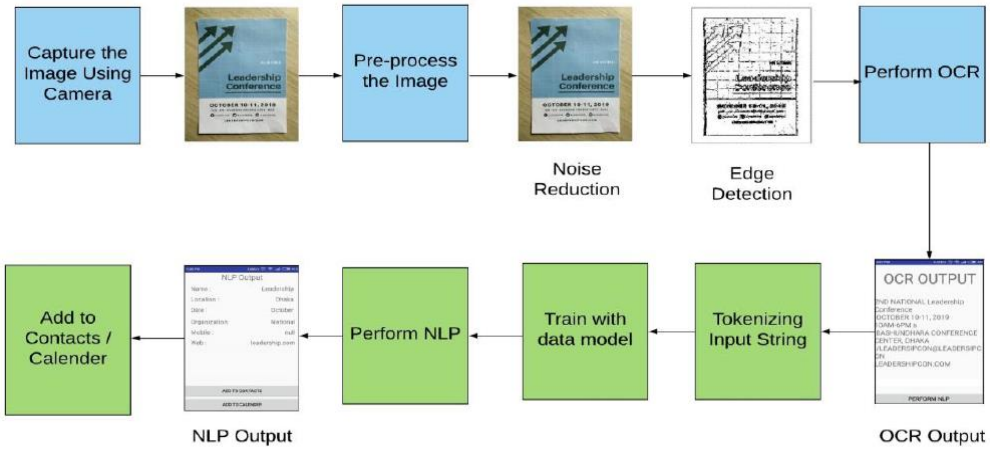


Fig. 2. Overall flowchart showing the system framework of our application.

In this project, OCR and NER are implemented on separate modules and then integrating them together to complete our application. OCR is implemented using the Text Recognition API which is an integral part of Google Cloud Vision API Engine. It is an open source Application Programming Interface (API) to detect the text from an image. We have tried to improve the image processing capability by considering several environmental conditions (lighting, reflection, rotation, and scaling). Therefore, we have used several preprocessing techniques to make the image suitable for performing OCR. The preprocessed images performed very accurately to determine the texts from the images. When performing the preprocessing we have found that OCR accuracy is maximum when the image is converted into some bi-level images. We have performed locally adaptive binarization of the image followed by the algorithm of N. Otsu [7]. The input images can contain some noises which are filtered properly and it is discussed thoroughly in section III. The output of the OCR needs further processing in order to extract the useful information such as name, location, time, date, email, web address etc.

In order to find the named entities from the given text, we perform NER. But the recognized text from the OCR usually contain some unstructured data which also require some preprocessing. At first, we have removed some characters from the text that could increase the complexity in performing OCR such as comma, semicolon, and brackets. Next, the given string is tokenize into separate tokens to perform NER. As the input text is unstructured we have faced a lot of difficulty in extracting the accurate information. Several research works have been done to extract information from well-formatted document [8] but it is always difficult to extract information from unstructured data. Since we have to use unstructured data in order to find the named entities, our work in similar to performing Named Entity Recognition in twitter data [9]. The reason we chose NER to extract the named entities from data is for its learning capability from given training data. Therefore, an appropriate training dataset is developed in order to extract the named entities accurately. Moreover, we employ regular expressions to find the email and website information from the given text as it involves less complexity.

III. System Overview

This section discusses the overall structure of our application and can be easily understood by Fig. 2. The major stages of the pipeline of our application are noted below:

1) Image acquisition: The built-in camera in Android device is used in our application. The user can select the input image from the storage device or capture the image after running the application. So the image resolution and quality may vary from device to device.
2) Preprocess the image: To find the most accurate text we must preprocess the captured image. The step includes resizing the image, converting the image into grayscale, and rectification of the image.
3) Text detection criteria: Firstly, we detect the block which is a continuous set of text lines i.e. paragraph. Secondly, we try to determine the individual lines of the text body which is the contiguous set of words on the same vertical axis. Finally, a word from the text body which is a

continuous set of alphanumeric characters on the same vertical axis is identified.

4) Performing OCR: Each detected block of the text is passed to the Text Recognition API developed by Google and the result is concatenated into a string.
5) Determining useful text and ignoring the unnecessary ones: In this stage, we use a Natural Language processing tool called OpenNLP to determine the required set of words (person name, location, date, time, website, email) for our application and ignore the rest of items.
6) Importing data to Contacts or Calendar: The useful data will be stored in the default Contacts or Calendar application of the Android device depending on the user choice.

IV. Image Processing

The image captured from the device camera might have some perspective distortions which ultimately makes the text recognition much more difficult. Therefore, we have given much more emphasis on preprocessing the image. The following subsections describe how the text recognition is performed from the captured image as well as can be understood by the flow chart given in Fig. 3.

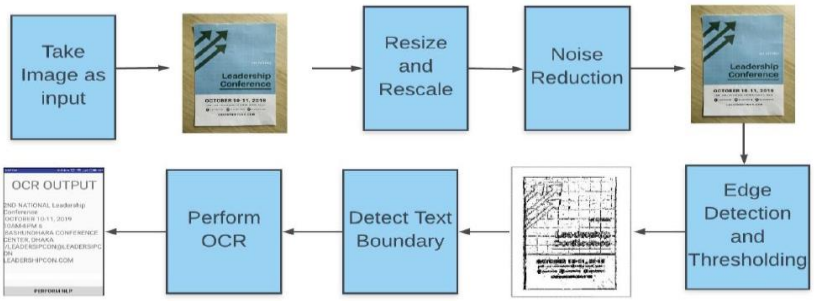


Fig. 3. Pipeline of image processing stage.

A. Scaling the Image

The quality and size of a captured image vary from device to device. In case of recognizing the text, a good quality image is a prerequisite as we perform the Optical Character Recognition (OCR) on the same image. To speed up the computation by maintaining proper accuracy we reduce the size of the image if it crosses a certain size threshold. Moreover, the best result that we have observed after our experiments are the image size of 1024x768px.

B. Noise Reduction

The performance of OCR could be reduced significantly if there is any noise in the input image. To diminish the noise in the image we at first convert the input image into grayscale then apply fuzzy filter noise reduction method [10]. Another image enhancement and noise reduction method introduced in Kushol *et al*. [11] for medical images is also tested but the result obtained after performing fuzzy filter proves better and one sample output is shown in Fig. 4.

C. Edge Detection and Thresholding

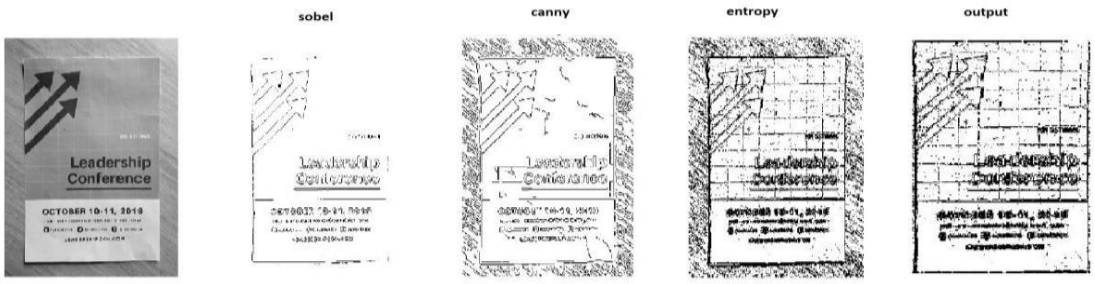
The input images of our application are taken from business cards, posters, and magazines which can be assumed as a flat plane on a 3D surface. If we can find the outer boundary of a business card, poster, or magazine which is generally a quadrilateral shape then unnecessary area of the image can be removed. So the main objective here is to find the largest quadrilateral within the given image. For detecting the edges, we first applied Sobel edge detection and Canny edge detector [12] technique but the edge obtained were not very convincing. In order to find even better result, we have applied Entropy based edge detection technique [13]. Fig. 5 shows the output of detected edge after using different edge detection techniques. Comparing all the edge detection methods we have found that Entropy based edge detection technique provides the most accurate result.

Fig. 4. Using fuzzy noise reduction method, (a) image after adding noise and (b) image after removing noise.

However, we have determined the outline of the card, magazine or poster which is the largest quadrilateral found after applying the aforementioned edge detection technique. The Hough transformation [14] is used to find the largest quadrilateral within the edges. We find the four nodes of the quadrilateral if the selected edges are along the corresponding axis and more the 1/4 th of the input image and give priority to the ones those are furthest from the image center. After finding the quadrilateral the portion that lies outside the boundary can be ignored and finally adaptive thresholding is performed to convert the grayscale image into binarized form.

D. Optical Character Recognition

The Text Recognition API [15] that we use in our project is a part of Google Cloud Vision API which is used for performing image content analysis. Previously Google used Tesseract OCR which is also an open source library. The reason we chose this Text Recognition API is that of its accuracy and ability to integrate with android device easily. Here, the preprocessed image acts as the input of our OCR. For accurate text recognition, the OCR determines individual words comparing with their given data sets.



(a) (b) (c) (d) (e)

Fig. 5. Edge detection and unnecessary area subtraction in (a) Original image (b) using Sobel, (c) using Canny, (d) applying Entropy-based method, and (e) final output after unnecessary area removal.

V. Natural Language Processing

The detected text after performing OCR is converted into a string. This string is needed to be further processed to extract useful information. We have used an NLP tool called NER for faster and accurate result. The computation of NER is done in the Android device but we also performed NER on a server to compare the performance and accuracy of Android platform. The pipeline is shown in Fig. 6. This section describes how useful information like person name, location, time, date, website, email etc. are determined using NER. To find the email, website and mobile number we have used Regular Expression using two Java classes named Pattern and Matcher.

A. Importing Predefined Models

Apache OpenNLP library is utilized which is a machine learning based Natural Language Processing tool. It helps to extract useful information from a given text. A sub-tool of OpenNLP called NER is used for finding the named entities. NER detects the named entities using a pre-defined model which consists of pre-trained datasets provided by the Apache OpenNLP. These training models are dependent on the language used and type of entity by which it was trained. We have used the English language in our input text and the model used also is in English. The pre-trained models that used in our application [16] are shown in Table I.

TABLE I: Pre Trained Apache OpenNLP Models

Purpose of Model	Model Used
Tokenizer	en-token.bin
Sentence Detector	en-sent.bin
Date Parser	en-date.bin
Name Finder	en-person.bin
Location Finder	en-location.bin
Organization Finder	en-organisation.bin

B. Preprocessing of Raw Text and Tokenization

The raw text obtained from OCR contains some characters like punctuations which increase the complexity when performing NER. Therefore we need to pre-process the raw text so that performing NER would be easier. Again in order to detect named entities from the raw text, it is also required that the text must be tokenized into some tokens. OpenNLP provides three ways to perform tokenization i.e Simple Tokenizer, Whitespace Tokenizer and Tokenization using pre-defined models. The most suitable tokenization process in our case is to tokenize using pre-defined models. The models used for tokenization is en-token.bin.

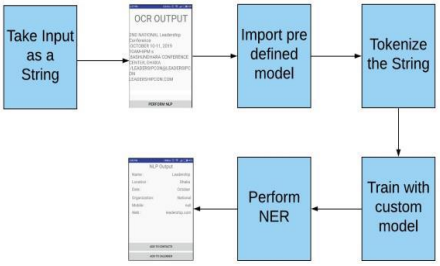


Fig. 6. Pipeline of Natural Language Processing (NLP).

C. Training the Datasets

During our experiment, it is observed that NER is not able to find all the named entities as it was trained for finding only American and British names. Therefore to fulfill our purpose we have to develop our own model using the models provided by Apache OpenNLP. We have created training datasets of nearly 1500 sentences to make our own model. Using these models we are able to determine the named entities correctly. It is observed that NER tool must learn to recognize previously unknown entities. While it is clear that NER follows the supervised learning but it is also evident that it trains the data sets following the results obtained from the test sets. Apache OpenNLP provides a separate Java API to perform the NER. This Java API is known as Name Finder API. It is implemented in our application by loading necessary libraries and training models.

D. Post Processing

The post-processing includes sorting the data obtained after performing NLP and importing them to contacts or calendar depending on the user activity. The sorting of the data is done according to name, location, organization name, date, email, and phone number. If all the related data can be extracted, then the corresponding value is assigned to that field otherwise it is kept empty. Once the text extraction is complete the user will be prompted to save the data to contacts or calendar based on the image criteria and the result achieved.

VI. Experimental Results

To develop our application, the IDE used is Android Studio. External libraries from Apache OpenNLP and Google

Cloud Vision are imported into the project.

A. Dataset

We have created a dataset that contained 200 images for testing purpose. In order to attain a better result, all the images are scaled in our dataset to 1024*768px. The information of database images is summarized in Table II.

TABLE II: MANUALLY CREATED DATASET TO DETERMINE THE PERFORMANCE AND ACCURACY OF OUR APPLICATION

Image Type	Number of Images	Image Format
Business Card	100	JPG
Poster	50	JPG
Magazine	25	JPG
Flyer	25	JPG

TABLE III: PERFORMANCE MEASURE FOR OCR

Image Type	Preprocessing	Accuracy in Percentage
Business Card	No	79
Business Card	Yes	96
Poster	No	70
Poster	Yes	90
Magazine	No	80
Magazine	Yes	92
Flyer	No	72
Flyer	Yes	92

B. Performance Measure

Firstly, we have performed OCR by capturing the images from business cards, posters, magazines and flyers using a mobile device. The accuracy of OCR output is measured after comparing with real data. When all the characters of an image are correctly identified by our proposed method that image is considered as a correct output. Table III shows the performance result of OCR in our application. For all types of image source, accuracy is obtained around 90 percent whereas business card achieves 96 percent accurate outcome.

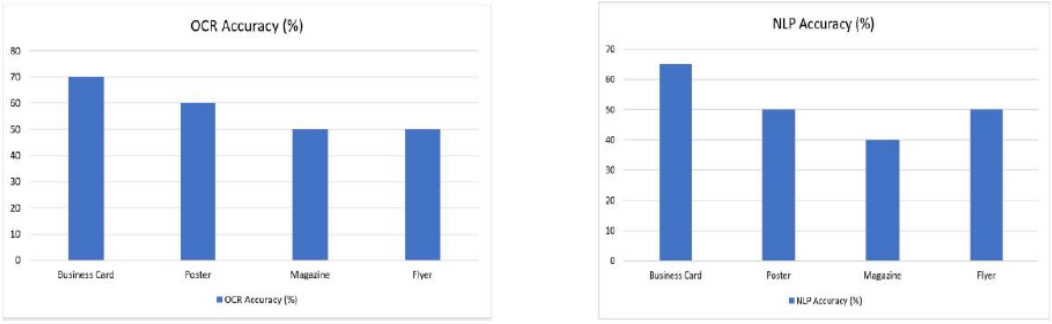
Table IV shows the performance of NLP in our application. The input of NLP is actually the output of OCR. As the input data is unstructured, so pre-processing improves the overall performance. It performs very well in extracting information from business cards. But in case of magazine, poster or flyer the performance decreases due to a lot of unstructured and ambiguous data. Fig. 7 shows two bar chart which represent the overall performance measure of our application.

TABLE IV: PERFORMANCE MEASURE IN NLP

Image Type	Preprocessing	Model Used	Accuracy in Percentage
Business Card	No	Custom	90
Business Card	Yes	Custom	92
Business Card	Yes	Pre-defined	62
Poster	No	Custom	88
Poster	Yes	Custom	90
Poster	Yes	Pre-defined	64
Magazine	No	Custom	56
Magazine	Yes	Custom	68
Magazine	Yes	Pre-defined	40
Flyer	No	Custom	64
Flyer	Yes	Custom	76
Flyer	Yes	Pre-defined	52

C. Performance Evaluation

The output of the obtained text from each type of images after performing OCR and NLP is shown in Fig. 8, Fig. 9, Fig. 10, and Fig. 11. We have evaluated the performance accuracy after comparing the output with corresponding real data.



(a) (b)

Fig. 7. Two bar charts showing (a) OCR accuracy and (b) NLP accuracy of our application.

(a) (b) (c)

Fig. 8. Output of (a) a business card input image, (b) OCR result and (c) NLP result.

(a) (b) (c)

Fig. 9. Output of (a) a poster input image, (b) OCR result and (c) NLP result.

(a) (b) (c)

Fig. 10. Output of (a) Magazine input image, (b) OCR result and (c) NLP result.

(a) (b) (c)

Fig. 11. Output of (a) a flyer input image, (b) OCR result and (c) NLP result.

After the completion of OCR and NLP all the useful information are stored in some String which can add these values to contacts or calendar. Based on the user choice and relevant data the user will be able to store the information either in their default contact application or calendar application. The outcome is shown in Fig. 12.

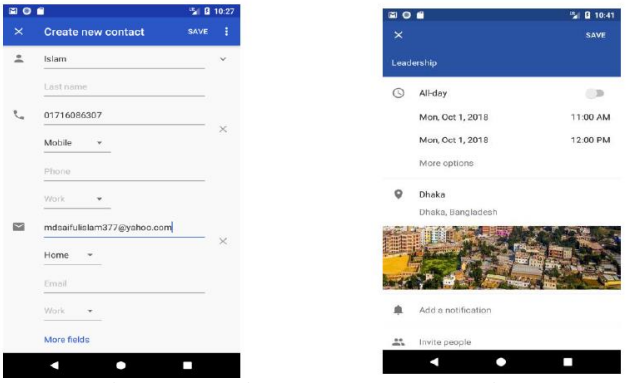


Fig. 12. The final outcome of the experiment: (a) Output for business card and (b) output for a poster image.

VII. Conclusion and Future Work

Extracting meaningful information from real-world images through the mobile camera is a quite challenging task. Our OCR accuracy is quite satisfying even though we have faced difficulty in processing the image due to uneven illumination, perspective distortion and misplace text. Again during performing NER the problem raised by the presence of noisy data inside the text is solved with the help of fuzzy noise reduction filter. Overall our application framework is very user-friendly and robust. However, further improvement can be done in the User Interface. It has all the functionality to be a stand-alone app for extracting useful information of certain type of images. In the future, we are hoping to extract more categorized information from any type of images.

References

[1] Google Cloud Vision API. [Online]. Available: https://cloud.google.com/vision/docs/ocr

[2] Apache OpenNLP 1.8.4 documentation. [Online]. Available: https://opennlp.apache.org/docs/
[3] B. Q. Vuong and H. N. Do, "Design and implementation of multilanguage name card reader on android platform," in *Proc. 2014 International Conference on Advanced Technologies for Communications (ATC)*, 2014.
[4] S. Bhaskar, N. Lavassar, and S. Green, "Implementing optical character recognition on the android operating system for business cards," *EE 368 Digital Image Processing*, 2010.
[5] P. Sharma and K. Fujii, *Automatic Contact Importer from Business Cards for Android*.
[6] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51-56, 2010.
[7] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62-66, 1979.
[8] E. F. T. K. Sang and F. D. Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proc. of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 2003, vol. 4, pp. 142-147.
[9] A. Ritter, S. Clark, and O. Etzioni, "Named entity recognition in tweets: An experimental study," in *Proc. the Conference on Empirical Methods in Natural Language Processing*, 2011.
[10] D. V. D. Ville, M. Nachtegael, D. V. D. Weken, E. E. Kerre, W. Philips, and I. Lemahieu, "Noise reduction by fuzzy image filtering," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 4, pp. 429-436, 2003.
[11] R. Kushol, M. H. Kabir, M. S. Salekin, and A. A. Rahman, "Contrast enhancement by top-hat and bottom-hat transform with optimal structuring element: Application to retinal vessel segmentation," in *Proc. International Conference Image Analysis and Recognition*, 2017, pp. 533-540.
[12] J. Canny, "A computational approach to edge detection," *Readings in Computer Vision*, pp. 184-203, 1987.
[13] J. Chen, J. He, and G. Su, "Combining image entropy with the pulse coupled neural network in edge detection," in *Proc. 2010 17th IEEE International Conference on Image Processing (ICIP)*, IEEE, 2010.
[14] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Readings in Computer Vision*, pp. 714-725, 1987.
[15] Google text recognition API. [Online]. Available: https://developers.google.com/vision/text-overview
[16] Apache OpenNLP models for 1.5 series. [Online]. Available: http://opennlp.sourceforge.net/models-1.5/

Rafsanjany Kushol received his B.Sc. Engg. in computer science and engineering degree from Islamic University of Technology (IUT), Bangladesh in 2013. Currently he is working as a lecturer in the Department of Computer Science and Engineering, Islamic University of Technology (IUT), Bangladesh and finished his M.Sc. Engg. in computer science and engineering at the same university in 2018. His research interests include image processing, graph theory, biomedical image analysis, smartphone applications, and computer vision.

Imamul Ahsan is currently pursuing his B.Sc. Engg. in computer science and engineering degree from Islamic University of Technology (IUT), Bangladesh. His research interests include software defined networking, internet of things, smartphone applications, mobile computing and human computer interaction.

Md. Nishat Raihan is currently pursuing his B.Sc. Engg. in computer science and engineering degree from Islamic University of Technology (IUT), Bangladesh. His research interests include image processing, data mining, android applications and human computer interaction.