Towards Automatic Diagram Description for the Blind

Preprint · July 2017

DOI: 10.13140/RG.2.2.11969.04961

4 authors, including:

Seehait Chockthanyawat
Refinitiv Software (Thailand) Limited
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Ekapol Chuangsuwanich
Massachusetts Institute of Technology
49 PUBLICATIONS **453** CITATIONS
SEE PROFILE

Proadpran Punyabukkana
Chulalongkorn University
69 PUBLICATIONS **207** CITATIONS
SEE PROFILE

Towards Automatic Diagram Description for the Blind

Seehait Chockthanyawat
Asst Tech Research Group
Dept of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
Seehait.C@gmail.com

Ekapol Chuangsuwanich
Asst Tech Research Group
Dept of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
EkapolC@gmail.com

Atiwong Suchato
Asst Tech Research Group
Dept of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
Atiwong.S@chula.ac.th

Proadpran Punyabukkana
Asst Tech Research Group
Dept of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
Proadpran.P@chula.ac.th

ABSTRACT

Conventional methods for describing complex diagrams to blind people are either ineffective or inefficient. We describe our preliminary work on how one can describe a diagram to a blind person with minimal supervision. We explain our text localization method which beats commercially available off-the-shelf state-of-the-art systems. We also provide a prototype user interface which can effectively describe diagrams based on our user study.

Categories and Subject Descriptors

[**Information Systems Applications**]: Information retrieval—*Document representation, content analysis and feature selection*; [**Human-centered computing**]: Accessibility—*Accessibility systems and tools*

General Terms

Theory, Experimentation, Human factor, Design

Keywords

Diagram description, Text localization, Assistive Technology

1. INTRODUCTION

There have been many developments in assistive technologies for the visually impaired recently. Screen readers such as the one in [1] or camera devices that can read printed text [9] help the blind access information that are otherwise unavailable to them. Despite these advances, the blind still have many difficulties in understanding printed materials, especially photographs and illustrations, where current solutions usually only identify the predominating object [7] rather than describing the picture or the scene as a whole.

Describing images using natural language has garnered much interest in the computer vision community recently. [4] can describe the relationships between objects or the actions being taken within a particular image. Not only this can assist the blind in understanding images or the world around them, it also has implications for image retrieval [8].

In the field of education for the blind, textbooks are usually converted to braille or audio books. However, figures, graphs, and diagrams provide difficulties for conventional methods. They can still be described in words or embossed. However, complicated illustrations such as diagrams with many nodes and edges does not lend themselves well to such methods due to the amount of content. Moreover, automatic methods for describing typical pictures such as ones mentioned previously would fail to capture the relationships between contents that are in both textual and visual format. For such illustrations, educators typically go through the diagram together with the blind, verifying their understanding as they go along. This process can sometimes take up to six hours per diagram. Thus, an automatic system that can describe any diagrams to the blind effectively and efficiently would be very beneficial for the education of blind people.

This system, which we call the Diagator, would help the user navigate through the diagrams, explaining the relationship between each entity. The system should be able to first locate all the texts, arrows, and boxes, and identify what they are, and their relationship within the diagram. It also needs a user interface that can explain the diagram to the visually impaired in an effective manner. This paper presents some preliminary work on the Diagator system. We focus on two of the main components, namely text localization, and the user interface.

The rest of the paper is organized as follows. In Section

0*I-CREATE* '17 Kobe, Japan
0

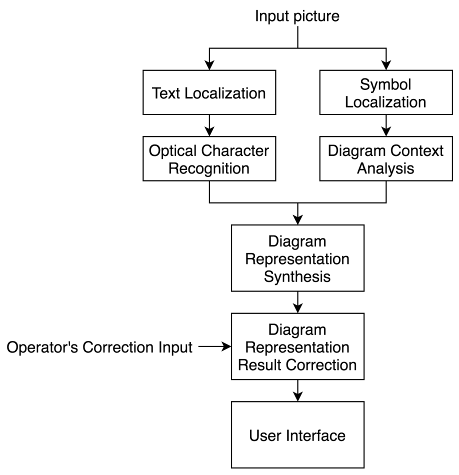

Input picture
Text Localization
Symbol Localization
Optical Character Recognition
Diagram Context Analysis
Diagram Representation Synthesis
Operator's Correction Input
Diagram Representation Result Correction
User Interface

Figure 1: Diagator's system overview

2 we describe the overall framework for our Diagator system. In Section 3 we go over our proposed text localization algorithm and evaluate it against another state-of-the-art system. In Section 4 we describe and evaluate how we can describe diagrams automatically to the blind. Finally, we conclude the paper in Section 5.

2. SYSTEM OVERVIEW

The Diagator has several components as outlined in Figure 1. A typical diagram consists of text and different kinds of symbols, such as arrows and bounding boxes which can have different shapes. To mark the location of these text and symbols, the input image is segmented into different regions using the text localization module and the symbol localization module. Then, the text image segments from the previous step are converted to plain text using the Optical Character Recognition (OCR) module. Simultaneously, the symbol segments are recognized in order to analyze the diagram context using the diagram context analysis module. The results from the two aforementioned modules are combined and used to build a machine-readable representation, in this case a graph in json format, by the diagram representation synthesis module. This step can also have a human operator rectify any mistakes made by the system. Finally, the diagram representation is used by the user interface in order to describe the diagram to user.

For this paper, we focus on two main parts, namely the text localization module and the user interface.

3. THE TEXT LOCALIZATION MODULE

In this section, we describe the work done for our text localization module. We start by describing the corpus used in our work, and then proceed to describe our proposed text localization algorithm.

3.1 The diagram corpus

A diagram is an illustration that describes the relationships between different entities. For our Diagator system, since we plan to represent the image as a graph, we focus on images that have a clear representation for nodes and edges connecting the nodes. The nodes and also the edges can have texts describing the entity or the relationship. The diagrams can contain text in either English or Thai, and also numerals. To ensure the quality of the images, all images should originally be in digital format, as in we do not process photographs of printed materials. We also want to avoid vector graphic formats, which will trivialize the text localization. For the purpose of our text localization work, the text regions were manually labeled as positive examples, while the unmarked regions were considered negative examples.

We collected 320 diagrams from various sources, such as software engineering textbooks, database management textbooks, and from the Internet. These 320 diagrams were separated into a 199-image training and a 121-image test set.

3.2 Text Localization methodology

Text localization, the task of finding the location of any text in an input image, is a standard task in computer vision. Recently, methods based on deep learning have yielded state-of-the-art results [5, 10]. However, these works were done on natural images which could have many kinds of variation in the text, such as occlusion or distortion effects, which do not occur in our target images. Thus, we followed a more traditional and simpler framework proposed in [6] which was also optimized for natural images, but required a lot less computation. We made some modifications to make it more suitable for diagrams.

3.2.1 Preprocessing

For preprocessing, each picture is segmented into 16x16 regions with 1 pixel overlap. Features are then extracted from each region. Each image is converted to a gray-scale image, A. According to [6], two-dimensional Discrete Cosine Transform (DCT) was applied to filter unwanted information such as background textures. However, we do not find it beneficial for our task.

3.2.2 Feature Extraction

From every 16x16 block of gray-scale image, we extract two main features, namely variance and contrast features. These features are often used in many computer vision tasks.

Variance features are computed for each 16x16 block using Eq. 1.

$$Variance = \frac{\sum_{i=1}^{r}\sum_{j=1}^{c}(A(i,j)-\mu)^2}{(r \times c)-1} \tag{1}$$

where r and c are the number of rows and columns of the image. For the contrast features, we can compute contrast features in the direction θ according to Eq. 2

$$Contrast_\theta = \sum_{i,j} |i-j|^2 p_\theta(i,j) \tag{2}$$

where P_θ is the Grey Level Co-occurrence Matrix (GLCM) computed in the direction θ [3].

3.2.3 Classification

We used Support Vector Machine (SVM) in order to classify each 16x16 block into text or non-text [2]. We used 3971 positive and 3372 negative patches to train the SVM. The negative examples were randomly selected from the non-text regions of the training images.

We used the radial basis function kernel and optimized for the gamma and cost parameters using 10-fold cross valida-

method	training error	held out error
pre-processing with DCT	9.94%	11.98%
pre-processing without DCT	1.81%	5.68%

Table 1: Text localization results on the heldout training data with and without DCT pre-processing.

number of contrast features	training error	held out error
2	54.78%	46.85%
4	34.04%	38.86%
12	1.81%	5.68%

Table 2: Text localization results on the heldout training data with different number of contrast features used.

tion. This configuration achieved 1.81% training error and 5.68% held out error.

3.2.4 Post-processing

The classification results are smoothed using morphological opening and closing, and finally a Gaussian blur filter. Then, bounding boxes can be generated using the minimum bounding box algorithm provided in MATLAB.

3.3 Experimental results and discussion

We start our experiments by evaluating the effectiveness of the DCT pre-processing mentioned in Section 3.2.1. Using 10-fold cross validation, we compared the classification accuracy between pre-procssing with and without DCT. Table 1 summarizes the results. As shown, the performance greatly improves by omitting the DCT because diagrams usually have less background textures than natural images.

We then compare the difference in performance between using different numbers of directional contrast features. Using 12 directions yields the best classification results as shown in table 2. We believe this is because there are several lines and arrowheads in our application which is very similar to text. Additional directions help distinguish between text and other contents.

Finally, we compare our fully tuned method with another state-of-the-art text localization service, namely the Google Vision API[1]. The two systems were applied on the test set which contains 1159 text regions. We used precision, recall, and f score as metrics to measure the performance of both system. A candidate location is considered correct if it encapsulates all the text in the ground truth location.

The results are shown in Table 3. Our system outperforms Google API in all 3 metrics. We believe that this is because Google's training data heavily contains natural images which are quite different from our target images. This signifies the importance of having a algorithm designed and tuned specifically for diagrams. Since our goal is to use the text localization results to describe diagrams, identifying as many possible text locations as possible, i.e. a higher recall, is desirable. Moreover, from the point of view of a human operator who can correct the mistakes generated by the system, removing false alarms is relatively simpler than manually transcribing texts that were missed by the system. Our system is able to achieve a high recall rate of 0.98, while the Google API only gets 0.92 recall, meaning that the human operator would have to add new text locations 4 times as often if the Google's API is used instead of our text localization method. We would say that our system greatly outperforms the Google API for this task.

[1]https://cloud.google.com/vision/

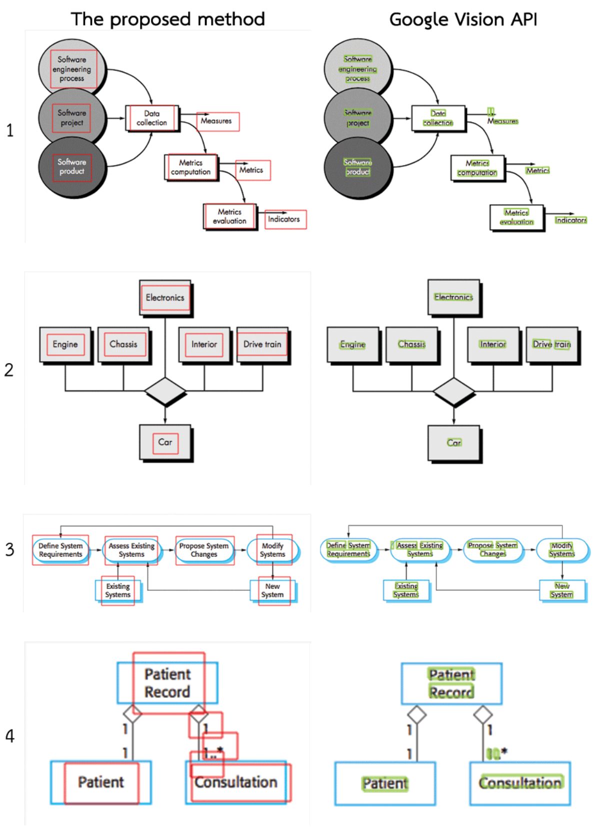


Figure 2: Localization results (1 - 4)

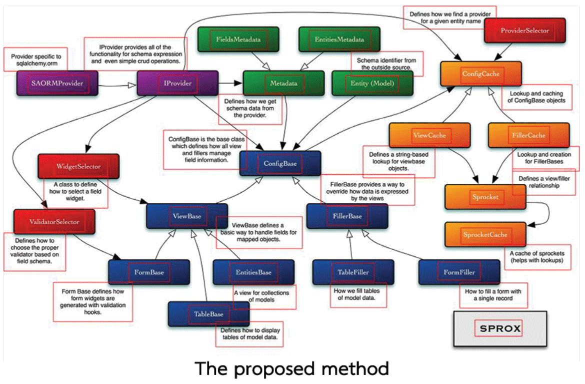
method	precision	recall	f
proposed method	0.94	0.98	0.96
Google Vision API	0.88	0.92	0.89

Table 3: Comparison between our text localization method and Google Vision API on the test set.

Fig. 2 and 3 show some examples of the localization results from our system and Google Vision API. As shown, the outputs from our system and Google's are quite similar except for the harder cases. For diagram number 1-3, ours and Google's are equally good, except for the one false alarm at the arrow head in Google API's output. Diagram number 4 shows mistakes that are common in both methods because some portions of the diagram symbols can be similar to text. For diagram number 5, there are a lot of misses in the Google API's output (see the text in the blue and purple boxes).

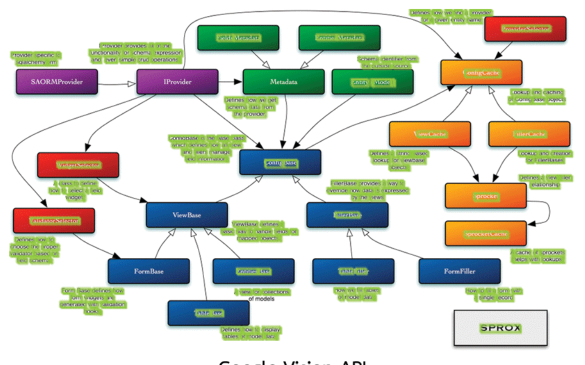
4. DIAGATOR USER INTERFACE

In this section, we discuss our prototype user interface and the user studies we conducted to evaluate the effectiveness of such interface. We made the interface simple so that it can be easily controlled by a blind person. Our application can be controlled by only 7 keyboard buttons: the four arrows keys, the space bar, the Enter key, and the Esc key. The


Defines how we find a provider for a given entity name
ProviderSelector
FieldsMetadata
EntitiesMetadata
Provider specific to sqlalchemy.orm
IProvider provides all of the functionality for schema expression and even simple crud operations.
Schema identifier from the outside source.
SAORMProvider
IProvider
Metadata
Entity (Model)
ConfigCache
Lookup and caching of ConfigBase objects
Defines how we get schema data from the provider.
ViewCache
FillerCache
ConfigBase is the base class which defines how all view and fillers manage field information.
ConfigBase
Defines a string-based lookup for viewbase objects.
Lookup and creation for FillerBases
WidgetSelector
Defines a view/filler relationship
A class to define how to select a field widget.
FillerBase provides a way to override how data is expressed by the views
Sprocket
ValidatorSelector
ViewBase
FillerBase
ViewBase defines a basic way to handle fields for mapped objects.
SprocketCache
Defines how to choose the proper validator based on field schema.
A cache of sprockets (helps with lookups)
FormBase
EntitiesBase
TableFiller
FormFiller
Form Base defines how form widgets are generated with validation hooks.
A view for collections of models
How we fill tables of model data.
How to fill a form with a single record
TableBase
SPROX
Defines how to display tables of model data.

The proposed method

5



Google Vision API

Figure 3: Localization results (5)

arrow keys are used to traverse between nodes within the diagram. The down arrow moves from the current node to the first child node. The up arrow is used to move to the first parent node. The left and right arrows are used to switch between sibling nodes on the same level. The space-bar can be used to replay speech describing the current node. Pressing the Enter key replays the speech describing the information about parents, siblings, children, and loops. Finally, the user can hit Esc to exit the application.

To choose the starting node, we choose the first node we found from the top left of a diagram which has no parent. If we can't find any node that satisfied the prior condition, we just choose the first node from the top left.

To deal with loops, we provide the user all of the loop information, such as the total number of loops in the diagram, and the number of nodes in each loop.

When the user arrives at a node, we provide the user, by a synthesized speech, all information about the location with respect to the graph, i.e. the node label, and the number of parent nodes, child nodes, and sibling nodes. If the user navigates to an unavailable node, for example, attempting to go to a child of a childless node, we tell the user that the required node is not available.

4.1 Usage experiments

We conducted experiments to evaluate the effectiveness of our user interface. Ten physically normal users are asked to use the interface and reconstruct diagrams number 1 and 3 in Fig. 2. They can continue to use the interface during the reconstruction. Since we are interested only in the usability of the interface, we used human generated ground truths as the diagram representations for the system. All users were able to completely reconstruct the diagrams. The average time for completion is 4 minutes per diagram and the maximum time is 9 minutes. Thus, our proposed user interface seems to be effective in describing diagrams.

Several users mention that a bookmark function which lets the user traverse back to a bookmarked node would be helpful. We plan to add such a function in future iterations.

5. CONCLUSIONS

We described our preliminary work on the Diagator system, an automatic system for describing diagram illustrations to the blind. We explained our text localization method which outperformed conventional methods for diagram text localization. We also proposed an effective interface to describe diagrams to the blind. For future work, we would like to continue our work on other pieces of the Diagator and make it able to handle scanned or photographed inputs. We also would like to add a functionality for the user to verify the user's understanding and provide the appropriate feedback if needed.

6. REFERENCES

[1] J. P. Bigham, C. M. Prince, and R. E. Ladner. Webanywhere: a screen reader on-the-go. In *Proc. 2008 Int. cross-disciplinary conf. on Web accessibility (W4A)*, pages 73–82. ACM, 2008.

[2] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[3] R. M. Haralick, K. Shanmugam, et al. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, 3(6):610–621, 1973.

[4] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[5] T. Kobchaisawat and T. H. Chalidabhongse. Thai text localization in natural scene images using convolutional neural network. In *Asia-Pacific Signal and Information Processing Association, 2014 Annu. Summit and Conf. (APSIPA)*, pages 1–7. IEEE, 2014.

[6] T. Kumuda and L. Basavaraj. Detection and localization of text from natural scene images using texture features. In *Computational Intelligence and Computing Research (ICCIC), 2015 IEEE Int. Conf. on*, pages 1–4. IEEE, 2015.

[7] V. Mohane and C. Gode. Object recognition for blind people using portable camera. In *Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave), World Conf. on*, pages 1–4. IEEE, 2016.

[8] V. Ordonez, X. Han, P. Kuznetsova, et al. Large scale retrieval and generation of image descriptions. *Int. Journal of Computer Vision*, 119(1):46–59, 2016.

[9] L. Stearns, R. Du, U. Oh, Y. Wang, et al. The design and preliminary evaluation of a finger-mounted camera and feedback system to enable reading of printed text for the blind. In *European Conf. on Computer Vision*, pages 615–631. Springer, 2014.

[10] R. Zhu, X.-J. Mao, Q.-H. Zhu, et al. Text detection based on convolutional neural networks with spatial pyramid pooling. In *Image Processing (ICIP), 2016 IEEE Int. Conf. on*, pages 1032–1036. IEEE, 2016.