From Stroke to Finite Automata: An Offline Recognition Approach

Abstract

A major challenge in making online education easier and more effective lies in developing automatic recognition, interpretation, and grading systems that can provide meaningful feedback to lecturers and students. Formal Languages and Automata Theory is a major module for many computer science and computing programmes worldwide. In such a module, students are taught how to design a finite state machine to recognise words in any given language. Despite the wide acceptance of this module by most universities across the globe, most students find this module difficult, boring and too abstract. Several research has been conducted on how to make this module interesting, but there still exist some gap. In this work, we propose and implement a system that assistant learners in learning this module. The system is in two units: unit one focus on offline recognition of hand-drawn finite automata diagram and the second unit focus on the tutor system. The system produced a 97% recognition rate. In future work, we intend to use formal grammars (second unit) to represent the recognised FA components, this will be used to automatically parse the output of the recognition system to determine if valid FA has been drawn.

1 Introduction and Problem Statement

Research in the area of handwritten document recognition is gradually shifting from normal text to the recognition of structured diagrams such as maths notation [1], music notation [2], engineering drawings [3], and UML[1] [4]. Structured diagram recognition systems can be distinguished by the systems taking online input in an interactive manner from the system user (e.g. the user draws using electronic input devices such as tablets, electronic pens), or the systems that perform image processing on scanned paper documents obtained offline. This paper focuses on how to perform offline recognition on finite automata (FA) diagrams, which will be done using scanned or photographed paper documents obtained from students. Few structured diagram recognition are in existence, but to the best of our knowledge, this work will be the first to implement a system that can automatically recognise an hand-drawn FA image and automatically generate its corresponding *transition table*.

This work attempts to solve a major problem faced by computer science (CS) students taking a course in formal language and automata theory (FLAT). We conducted a survey and realised that over 95% of Africa universities offer FLAT course in CS. Also, over 80% of the students find this course too difficult, abstract and boring. Numerous research has been conducted on how to make the course content of FLAT interesting and help learners on how to conceptualise the different topics [5–7]. The research conducted in [8] attempted to find out the difficulties experienced by FLAT learners; they identified problem-solving as the major challenge. FLAT tools used for visualization such as JFLAP[2] [9] helps learners get an idea of the given problem and also check their solutions, however, these tools do not help them to further develop their problem-solving skills to a certain acceptable stage. To further develop their skills, most learners work on extra problems with feedback at each stage. In this work, we propose a tool that can serve as a tutor and help overcome all this challenge.

[1] Unified modelling Language

[2] Java Formal Language and Automata Packages

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

The developed tool focuses on the offline approach to learning FAs (see Figure 1). In this, we present an image recogniser trained to recognise hand-drawn FA diagrams (using machine learning approaches). Learners are allowed to draw FA images using their pen and paper and then take a picture/ capture of their drawn images using either their phone camera or scanners. The captured FA image is uploaded into the tutor and the corresponding recognition button is pressed and the process begins. Once the process is done, the learner is given the option to save the recognised image as a .PNG or proceed to the next unit (within the tutoring system) where the corresponding transition table of the recognized image (output) can be generated by pressing the generate transition table button (using formal grammar technique). Once this button is pressed, the system checks if a valid FA diagram has been drawn and it automatically grades the student and give back suggestions and feedback on how to improve on her drawing in the future.

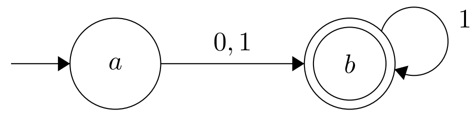

Figure 1: A Finite Automaton.

Due to the theme of this workshop, we focus on the recognition aspect (machine learning) of this work and leave the formal computing (grammar) aspect out. The recognition system is implemented and tested in phases: Data Acquisition and Annotation → Pre-processing and Thinning → Stroke Extraction → Text and Graphics Separation → Segmentation → Arrow Detection and Recognition → Shape Detection and Recognition → Text Recognition and Evaluation.

2 Methodology and Technical Contributions

To the best of our knowledge, this work will be the first to create a database[3] and recognise FA diagrams using an offline approach. We collected 500 FA images (BB_a) from students and split them into — 300 training images, 150 test images, and 50 validation images. We developed a tool to annotate these diagrams. During pre-processing, we removed noise and perform thresholding on the images. We improved on the existing Zhang-Suen thinning algorithm because of the nature of our drawing. The stroke extraction phase involves the extraction, merging and labelling of strokes. We used Markov Random Fields (energy function – unary and pairwise energy function) to perform this task. During the text and graphics separation, we used bidirectional long short-term memory recurrent neural network (BLSTM-RNN). We biased the result of this classifier during our experiment and got a stroke classification accuracy of 97.2 on our dataset i.e. 96.8 for the graphics/shape class and 97.5 for the class text. The classifier was tested on the IAM Online Database document, it achieved an accuracy of 96.5% on the database.

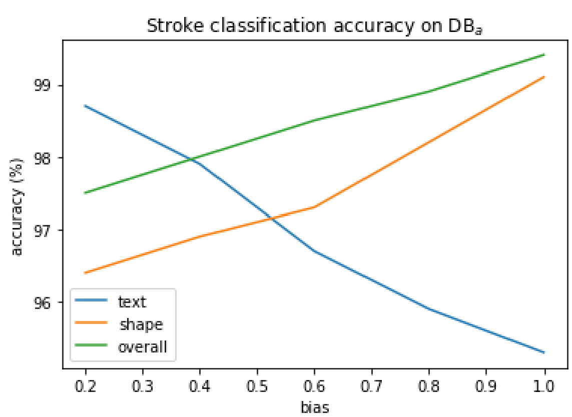

Figure 2: Text and stroke classification accuracy on DB_a.

During the segmentation phase, the characteristics of an FA diagram components (various components are made up of strokes that are somehow connected to each other, hence forming a closed structure

[3] will be made publicly available soon

of strokes) was utilised. We focus on finding the distance and convexity between the endpoints of consecutive strokes that form the FA structure. The values from the distance and convexity are used to reject improper strokes in the structure and to process and compute the confidence of promising strokes. To detect arrows or transitions, we developed an algorithm that searches and classifies an arrowhead and the shaft, the classifier accuracy is 98%. To detect and recognise circles or states, we used multiclass SVM classifier. We base the classifier on hybrid features capturing dynamic information and the visual appearance of symbols. Text recognition was done last. The input at this phase is the bounding box of the text layer extracted from the whole diagram during the text and graphics separation phase. We used Tesseract OCR [10] to process the content of the bounding boxes. The accuracy of the text recogniser was tested by comparing its output with the values of the input text and we got 92%.

Table 1: FA datasets

Database	**Writers**	**Patterns**	**Diagrams**	**Strokes**	**Symbols**
DB_a	50	10	500	40,855	19,568
DB_b	100	2	200	11,102	3,801

Table 2 shows a detailed overview of the datasets in Table 1 after separating them into training, validation and test dataset.

Table 2: FA database overview

		Writer	**Patterns**	**Diagrams**	**Strokes**	**Symbols**
DB_a	Training set	50	10	300	20,426	11,562
	Validation set	25	5	50	6,808	1,986
	Test set	50	10	150	13,621	6,020
DB_b	Training set	100	2	150	8,320	2,852
	Test set	50	2	50	2,782	949

After the whole recognition phase, to test how easy and quick our algorithm can learn to recognise more difficult diagrams, we went further to collect 200 images from students examination papers. We created a database for this called DB_b. The result of the whole recognition process is presented in Table 3.

The recognition system achieved high precision (see Table 3). We test the online FA dataset (we call it DB_{on}) of [11] using our algorithm. We went further to compare the performance of DB_{on} with DB_a and DB_b. The results are presented in Table 3.

Table 3: Stroke and symbol comparison between our algorithm and DB_{on}.

Symbol Class	Stroke Recognition			Symbol Recognition		
	DB_{on}	DB_a	DB_b	DB_{on}	DB_a	DB_b
State	96.1	96.2	90.4	95.0	94.5	87.9
Final State	94.4	96.8	91.0	95.2	95.7	87.6
Text	99	90	90	99	90	90
Start Arrow	96.2	97.4	89.8	94.1	96.0	87.6
Arrow	92.2	96.6	87.6	93.1	94.0	93.1

3 Future Work

We intend to extend the developed system for flowcharts and UML diagrams.

References

[1] J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, and L. Dai, “Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition,” *Pattern Recognition*, vol. 71, pp. 196–206, 2017.

[2] J. Adamska, M. Piecuch, M. Podgórski, P. Walkiewicz, and E. Lukasik, "Mobile system for optical music recognition and music sound generation," in *IFIP International Conference on Computer Information Systems and Industrial Management*. Springer, 2015, pp. 571–582.

[3] T. Hammond and R. Davis, "Ladder, a sketching language for user interface developers," in *ACM SIGGRAPH 2007 courses*. ACM, 2007, p. 35.

[4] B. Di Martino and A. Esposito, "A rule-based procedure for automatic recognition of design patterns in uml diagrams," *Software: Practice and Experience*, vol. 46, no. 7, pp. 983–1007, 2016.

[5] D. Berque, D. K. Johnson, and L. Jovanovic, "Teaching theory of computation using pen-based computers and an electronic whiteboard," in *ACM SIGCSE Bulletin*, vol. 33, no. 3. ACM, 2001, pp. 169–172.

[6] C. I. Chesñevar, M. L. Cobo, and W. Yurcik, "Using theoretical computer simulators for formal languages and automata theory," *ACM SIGCSE Bulletin*, vol. 35, no. 2, pp. 33–37, 2003.

[7] D. Zingaro, "Another approach for resisting student resistance to formal methods," *ACM SIGCSE Bulletin*, vol. 40, no. 4, pp. 56–57, 2008.

[8] N. Pillay, "Learning difficulties experienced by students in a course on formal languages and automata theory," *ACM SIGCSE Bulletin*, vol. 41, no. 4, pp. 48–52, 2010.

[9] S. H. Rodger and T. W. Finley, *JFLAP: an interactive formal languages and automata package*. Jones & Bartlett Learning, 2006.

[10] R. Smith, "An overview of the tesseract OCR engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. IEEE, 2007, pp. 629–633.

[11] M. Bresler, D. Prusa, and V. Hlavác, "Detection of arrows in on-line sketched diagrams using relative stroke positioning," in *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, 2015, pp. 610–617.