OA-CHOCR: Online Adaptive Crowd-Based Handwriting OCR

Muthu Chandrasekaran[1]*, Jeetendra Patil[1], Chaitali Patel[1], Meenaz Devani[1]

Abstract
The goal of this project is to develop a note-taking software which will take as input, scanned handwritten notes shared from a "smart board" installed in educational institutions (and the like) including schools, universities, conference rooms etc. The software will potentially allow students or colleagues to search, categorize, and tag these notes shared in real-time in order to allow for easy navigation. For this, the software needs to be able to understand the content of these notes. The first task will be to identify/recognize the handwriting of a teacher/presenter who used the smart board to share his/her notes. Obviously, utility of such software will be contingent on the existence of a compatible "smart board" with real-time note-sharing abilities.

Keywords
Crowd Sourcing — Handwriting OCR — Software Development

[1] *Department of Computer Science, University of Georgia, Athens, GA - 30605*
***Corresponding author**: mkran@uga.edu

Contents

Introduction

Due to resource constraints, we simplified the scope of this project wherein our software will have a simple UI to upload a screenshot of a whiteboard with handwritten notes. The goal is to transcribe the note using an offline handwriting Optical Character Recognition (OCR) algorithm and display it to the recipient (end-user). In order to facilitate a feedback system to improve the accuracy of our offline OCR algorithm over time, the recipient will have the ability to accept or reject the resulting transcription. If accepted, the transcription is assumed to be satisfactory and reasonably accurate. If rejected, the software will seek an accurate transcription from the crowd using Amazon's Mechanical Turk (AMT) platform[1] which will then be used to re-train our OCR, online.

[1] www.mturk.com

1. Background

1.1 Handwriting OCR

Optical character recognition (OCR) is the conversion of typewritten text into machine-encoded text. OCR is a widely researched domain in the field of pattern recognition, artificial intelligence and computer vision. Early versions of OCR algorithms needed to be trained with images of each character, and worked on one font at a time. These versions typically have low accuracy but are easier to implement. However, advanced versions that have a higher degree of recognition accuracy for most fonts are now common. The goal is to reproduce a formatted output that closely approximates the original content including images, columns, and other non-textual components. Handwritten text recognition adds a level of complexity to typical typewritten text OCR algorithms. Handwriting OCR may require additional preprocessing operations [1] including handwriting movement analysis and highly sophisticated image processing and pattern recognition techniques to achieve desired levels of accuracy.

For our project, we implemented an algorithm called the **Mean Square Error (MSE) Algorithm** using the Java OCR toolkit[2] whose core concept, at the character level, is image matching using a least-square-error formula to score each training image's resemblance to the character being decoded. This algorithm is font-specific. A font is another name for a handwriting pattern/template. For each font, we will provide training images for each character that needs to be recognized for that font. The MSE algorithm is one of the early implementations of OCR which is simple yet reasonably effective.

[2] http://sourceforge.net/projects/javaocr/

1.2 AMT: Amazon Mechanical Turk

Amazon's Mechanical Turk [2] is a popular online marketplace for getting work done by others (the crowd) where individuals register as either *requesters* (task creators) or *workers* (paid task completers). Requesters can create and post tasks (or questions) called Human Intelligence Tasks (HITs). *Workers* can browse available tasks and are paid upon successful completion of each task. *Requesters* can review completed HITs and refuse payment for (intentional) subpar work. Being refused payment has negative consequences for the *workers*' reputation because *requesters* can limit their tasks to *workers* with high reputation. AMT also provides an **API** using which the *requestors* have the ability to programatically create and manage (modify the incentivization structure, approve/refuse payment etc.) HITs.

2. OA-CHOCR

We implement a hybrid software called *Online Adaptive Crowd-Based Handwriting OCR* (OA-CHOCR). The architecture of the software is shown in Fig. 1. As the name suggests, our software performs OCR of handwritten texts using the MSE algorithm and further probes the crowd if the end-user is unsatisfied with the results returned. It then uses these newly crowd-tagged handwriting samples to re-train the underlying OCR algorithm in an online fashion, making it an adaptive learner. This software is classified as *hybrid* because it takes advantage of the computational efficiency of computers as well as the intelligence of humans to make computers smarter over time. There are 2 main motivations behind using such a hybrid system:

1. Humans are inherently good at OCR-like tasks but computers are not!
2. There are NO guarantees on the crowd's response time. We may end up waiting for ever!

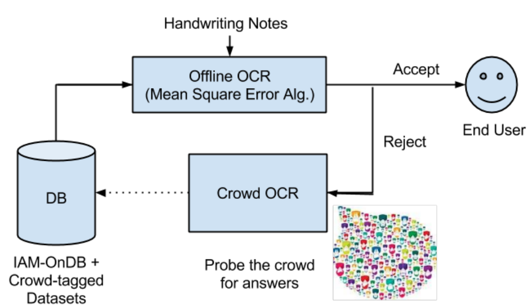

Figure 1. OA-CHOCR Architecture

A hybrid system such as ours will ensure instantaneous transcription of handwritten text. Our end-users wouldn't have to wait on the crowd for a simple transcription (or search) task. However, the underlying OCR algorithm may not have the desired level of accuracy. This inaccuracy can be corrected by using the help of humans, who are inherently good at OCR-like tasks, via crowdsourcing.

2.1 Dataset

For training and testing our OCR, we used an online english sentence database acquired from handwritten text on a whiteboard - **IAM-OnDB** [3]. The eBeam[3] interface was used to record the handwriting of users. The resulting database (IAM-OnDB) consists of more than 1,700 handwritten texts from 221 writers. It contains 86,272 word instances from a 11,059 word dictionary written down in 13,049 text lines. In addition to the recorded handwritten texts and their transcriptions, some information about the writers which could be useful for future work, were also stored in the database. We added some of our own training and testing data to IAM-OnDB to test our algorithm's generality.

2.2 Algorithm

Algorithm 1 OA-CHOCR Algorithm (*IAMonDB, targetImg*)

```
1:  DB ← IAMonDB
2:  trained ← meanSquareTrainer(DB)
3:  result ← meanSquareAnalyzer(trained, targetImg)
4:  if getUserFeedback (result) = accept then
5:      return result
6:  else
7:      hit ← createHIT(result)
8:      answer ← getHITAnswer(hit)
9:      if reviewAnswer (answer) = approved then
10:         approvedAnswer ← answer
11:     else
12:         go to 8
13:     DB.add (approvedAnswer)
14:     go to 2
15: return result
```

Our OA-CHOCR algorithm (shown above) takes as input the IAM-OnDB and the target handwritten image as input and attempts to return the transcription of the uploaded image. There are 5 major steps involved in this algorithm:

1. **Data Collection Step** - Obtain a tagged data set from the IAM-OnDB handwriting database (line 1)
2. **Prediction Step** - Train on this data set and perform handwriting OCR to transcribe the contents of the target image (lines 2 to 3)
3. **User Feedback Step** - Users can then accept/reject the transcription results returned after the **Prediction Step** (line 4). If rejected, then go to the **Probing Step**
4. **Probing Step** - Using the AMT Crowdsourcing Platform, use the crowds' help to transcribe the target image until at least one answer is approved (lines 7 to 12)
5. **Online Adaptation Step** - Add the approved answer back into our database to re-train our OCR for better accuracy (lines 13 to 14)

[3] eBeam System by Luidia, Inc. - www.e-Beam.com

3. Results and Discussion

Note that in its present state, our OCR software can only recognize alphabetic characters. Keeping that in mind, we evaluate the performance of the software on several test cases and share some preliminary findings below:

Case 1: When a particular handwriting font was already used during training, we obtained near 100% accuracy in the prediction (See Fig. 2).

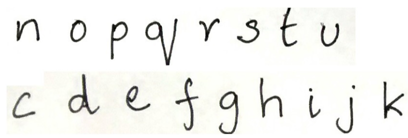

Figure 2. Eg. target image. Transcription Accuracy = 100%

Case 2: When the handwriting font was used in the training and when target image, contained multiple words spanning several lines of text, our offline OCR was able to predict with near 100% accuracy. (See Fig. 3).

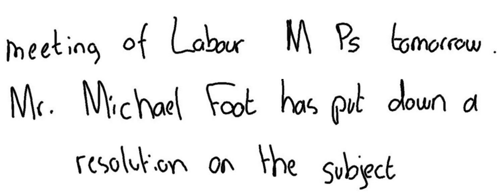

Figure 3. Target image with multiple lines. Transcription Accuracy = 100%

Case 3: When the handwriting font was **NOT** used in the training and when target image was captured in bad lighting, our offline OCR was able to predict with reasonable accuracy of about 60%. That is, on an average 60% of the characters were correctly recognized. (See Fig. 4).

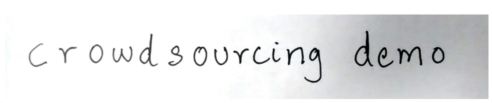

Figure 4. Target image under bad lighting. Transcription Accuracy = 60%

Case 4: When the handwriting font was **NOT** used in the training and when the handwriting in the target image was illegible or the characters were too close to each other or if the image was captured in poor lighting, our offline OCR was able to predict with only poor accuracy. We owe this problem to a simplistic implementation of the OCR algorithm. State-of-the-art sophisticated algorithms which consider handwriting movement analysis to detect cursive writing patterns, and with better pre-processing techniques will definitely improve our OCR's accuracy (See Fig. 5).

We requested the crowd to provide a better transcription if the transcription accuracy was less than 80% and automatically approved the answer provided. Alternatively, we could

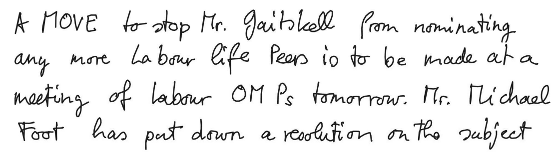

Figure 5. Illegible target image with multiple lines of text including characters that are too close to each other. Poor Transcription Accuracy

choose to accept an answer if the answer gets a fixed minimum number of votes of confidence. We then merged target image along with its accepted transription with the existing IAM-OnDB and re-trained our OCR. Although we were not able to directly observe improvements in the performance of the online adaptive OCR, we believe that with more data and more runs, we will begin to notice significant improvements.

4. Conclusions and Future Work

To conclude, we developed a hybrid software that leverages human intelligence to achieve a task difficult for computers (like handwriting OCR) and in the process becoming smarter over time. As a part of future work, we would like to implement a more sophisticated offline handwriting OCR that can be used to additionally predict numbers and special characters with greater accuracy. We think that using some kind of probabilistic model to predict which characters/words are more likely to occur after certain characters/words, will help improving the accuracy. We would also like to be able to upload several notes at once, and implement a keyword-search functionality to find notes containing those keywords, in addition to transcribing its textual content. Additionally, we could try different mechanisms to automate HIT management. It would also be interesting to investigate ways to improve the crowd's response time by using popular social media outlets like facebook as a platform to post and answer questions.

Acknowledgments

We would like to thank Dr Arpinar and the rest of the CSCI8380 class for their valuable suggestions.

References

[1] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in online handwriting recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.

[2] Michael Buhrmeste, Tracy Kwang, and Samuel D. Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? In *Perspectives on Psychological Science*, 2011.

[3] Marcus Liwicki and Horst Bunke. Iam-ondb – an on-line english sentence database acquired from handwritten text on a whiteboard. In *Proc. 8th Int. Conf. on Document Analysis and Recognition*, pages 956–961, 2005.