iOCR: Informed Optical Character Recognition, Election Ballot Tallies Implementation in Javascript

Mansur Yassin , Darian Jennings , Dr. Juan Gilbert
University of Florida , Computer and Information Science and Engineering , Social Computing for Good

Abstract. This report presents the development of the Informed Optical Character Recognition (iOCR) system and its implementation within JavaScript which aims to improve the accuracy of ballot verification such that the program will run with 100 percent accuracy. OCR engines often fail when processing structured documents like election ballots due to variations in results. Initially, the iOCR system used the object data structure by implementing a key-value pair for mapping ballot entries, but this method proved to be incorrect since it would resolve the candidate names but random characters and lines appeared throughout the scanned paper. The implementation shifted to account for a line variable that was hardcoded to be 10 lines so that the scanned ballot matches the original ballot definition This variable helped remove unnecessary characters and addressed inconsistencies in line length. The system applied the Levenshtein distance algorithm for post processing corrections with the final output formatted into HTML with tabulation in order to account for voters with visual impairments.

Introduction. Optical Character Recognition (OCR) technology is widely used to automate text digitization from printed documents; Tesseract is an open source OCR that is best for ballot verification because the software is capable of converting documents into digitized text. However, in structured documents such as election ballots, OCR engines often encounter significant challenges, especially when minor variations in formatting, spacing, and layout lead to errors in text recognition. This iOCR system was developed to address these limitations. Initially, the Javascript Implementation used key-value pairs to map ballot entries; After testing this approach with the Brother OCR scanner we found that this approach could not handle real world variations in scanned ballots due to randomness in character generation above the ballot definition title and below the final candidate selection. Since this approach was found to be incorrect, hardcoding the line as a variable made this approach consistent with accurate formatting, alongside regex based text cleaning to remove unwanted characters. Post processing corrections were applied using the Levenshtein distance algorithm to correct minor OCR errors, and the text- to - html converter outputted as the corrected scanned ballot.

Related Work. OCR has long been used in document digitization, but it is known to face difficulties in structured documents. Tesseract, is prone to errors when dealing with documents that have complex layouts like ballots (Smith, 2007). Nagy (2000) emphasized the need for post processing techniques to improve OCR accuracy in structured environments such as forms or ballots. This report is built upon the research findings by Oyibo, Louis, and Gilbert (2022) introduced in Informed Optical Character Recognition (iOCR), which integrates OCR systems with post processing algorithms like the Levenshtein distance to correct text recognition errors. The Levenshtein distance measures the minimum number of character edits (insertions, deletions, substitutions) needed to transform one string into another. This project extends the iOCR framework by implementing it in JavaScript and highlighting the importance of ballot verification accuracy through regex validation and structured data handling.

Character Validation using Regular Expressions. The iOCR system integrated regular expressions (regex) to clean and validate the OCR output such that unnecessary characters were removed and each line adhered to the ballot definition formatting. The regular expression begins by anchoring the match to the start of the line using the caret (^) which shows if the pattern applies from the beginning of the text. It then looks for any leading whitespace characters (such as spaces or tabs) with the *s pattern, which allows for zero or more occurrences of such characters. This is followed by an optional underscore*, matched by the [ ] ?* segment, where the question mark indicates that the underscore may or may not be present. After the optional underscore, the regex accounts for any additional spaces using another *s pattern. Next, the pattern \d? matches an optional single digit, allowing for lines that may or may not start with a number. After the digit, the pattern [.,_|\/]? allows for an optional punctuation mark, such as a period, comma, underscore, vertical bar, or forward slash. This guarantees if any common characters that might precede the main text in noisy OCR output are handled appropriately. The question mark again makes this optional, meaning the regex can still match lines that lack these characters. Following this, the regex looks for the delimiter ⇒ , surrounded by possible spaces on either side, using \s*==>\s*. This allows flexibility in the spacing around the delimiter, ensuring that the pattern matches even if extra spaces are present. Finally, the capturing group (.)* extracts the remaining text after the delimiter by matching any characters that follow. The candidate's name or proposition should now be captured correctly. The pattern concludes with a dollar sign, which anchors the match to the end of the line so that no extra characters are present beyond the captured text.

Handling Line Length Edge Cases. A significant challenge in processing scanned ballots is the variability in line length, where some lines may be too short or too long due to OCR errors. The iOCR system implemented a length check for lines shorter than 20 characters and it will be ignored. These lines were often incomplete or contained noise, and the length check helps remove randomness from the scanned output. Similarly, lines that exceeded the expected length were flagged for review or trimmed to avoid processing errors caused by excessive characters or unnecessary formatting.

Post Processing with Levenshtein Distance. After cleaning the text, the iOCR system applied the Levenshtein distance algorithm to correct minor errors in the scanned ballot lines. By calculating the minimum number of edits required to transform the scanned text into the expected ballot entry, corrections were stored in a new list so discrepancies such as misspellings or minor formatting errors were automatically corrected.

HTML Tabulation Process. The final stage of the iOCR system was converting the verified ballot data into HTML format for easy review. This process, known as HTML tabulation, involved rendering each cleaned and corrected ballot line into a separate HTML div element. These elements were tab indexed to allow for visually impaired users to activate character recognition via the audio software Jaws by easily navigating through the ballot in a browser and further verifying the accuracy of the OCR output. HTML tabulation provided an accessible and structured format for reviewing the corrected ballot data, with each ballot entry clearly separated for ease of validation.

Conclusion. This report outlined the development and evaluation of the iOCR system for ballot verification and its implementation in JavaScript. The initial approach using key-value pairs was insufficient for handling real world variations in scanned ballots. By adopting a formatted structure from the ballot definition and integrating regex for text validation, unnecessary characters were removed and each line met the necessary format. The Levenshtein distance algorithm further improved accuracy by correcting minor errors in the OCR output, and the HTML tabulation process provided a user-friendly format for reviewing the final ballot data. By handling each edge case evaluated from the data gathered through Brother OCR, patterns of inaccuracies proved to be reliable in refining the algorithm for ballot verification.

References :

Nagy, G. (2000). Optical Character Recognition: Theory and Practice. *Proceedings of the IEEE*, 88(8), 1400-1412.

Oyibo, K. U., Louis, J. D., & Gilbert, J. E. (2022). *iOCR: Informed Optical Character Recognition for Election Ballot Tallies*. University of Florida, Gainesville FL.

Smith, R. (2007). An Overview of the Tesseract OCR Engine. *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, 629-633.