Reading Progress Tracking: A Novel Autoencoder Model Approach

Almaz Shangareev[1] and Ivan Shanin[2]

[1] Lomonosov Moscow State University
[2] Institute of Informatics Problems, FRC CSC RAS

Abstract. Reading progress tracking is a challenging task in the eye-tracking field due to the overall measurement inaccuracy in eye-tracking systems. This study presents a two-step approach using the autoencoder model to solve the task of classification of text words into "read" and "non-read" classes using the eye-tracking data. The model utilizes fixation durations and text structure to produce a probability map representing the likelihood of each pixel being viewed and interprets this map using a pixel-wise average probability criterion to determine if the word was read. The proposed approach uses the dataset generated from the ZuCo 1.0 dataset and achieves an F-score of 0.9782, and mIoU of 0.9587 on a test set of 537 pages for word-wise "read" and "non-read" classification.

Keywords: Eye-tracking · Natural reading · CNN · Autoencoder · ZuCo

1 Introduction

1.1 The Role of Eye Movements in Reading

Reading is a complex cognitive process that involves the coordination of visual perception and cognitive processing. Eye movements play a critical role in reading, allowing readers to navigate the text and extract information from the printed page. However, reading isn't a continuous and uninterrupted process. While reading, eyes move in a series of fixations and saccades, pausing briefly to extract visual information during fixations and moving rapidly between fixations in saccades.

During fixations, eyes remain relatively still for periods ranging from 150 to 500 milliseconds, with the average fixation lasting between 200-250 milliseconds [1]. Saccades, on the other hand, are rapid, ballistic movements that propel the eyes forward by 7-9 letter spaces for readers of English. The duration of a saccade varies depending on the movement length, typically taking about 20-35 milliseconds for a typical reading saccade. As a result, it leads to the idea that most visual information should be extracted from the printed page during fixations, with readers processing the text in a sequence of snapshots, much like a slide show.

In addition, while most saccades move forward through the text, about 10-15% are regressive, moving backward in the text. Regressions occur about once

every 2 seconds, and most of them are short corrections of oculomotor error rather than comprehension difficulties. Another type of eye movement in reading is the "return sweep", which moves the eyes from the end of one line to the beginning of the next. For readers of English, return sweeps are right-to-left movements that do not count as regressions, as they move the reader forward through the text.

1.2 Problems in Reading Progress Tracking

In general, the main challenge of eye-tracking data analysis is a high level of noise in data [2], comprised by measurement errors in determining gaze position on any kind of flat surface i.e., screen or sheet of paper. Currently, most efforts to reduce measurement errors are usually applied on a registration step of the eye-tracking system, including signal filtering, smoothing and denoising, which does not utilize any high-level information about characteristics of gaze patterns. In the case of text reading a text-aware post-processing can significantly increase the quality of information that can be extracted from eye-tracking data, due to the fact that the reading process has special features and logic, some of which were described in Section 1.1.

1.3 Reading Progress Tracking Applications

There is still plenty of room for imagination regarding the applications of reading progress tracking. Some potential applications may include useful statistical information generation for reading systems; auto-scrolling mechanism; educational portal's supervising system; detection of disorders/diseases related to reading; automatic mapping of eye-tracking data into text in eye-tracking related studies.

Further exploration and research in this area can unlock additional innovative applications and contribute to advancements in reading technologies and methodologies.

2 Related Works

In the article [3], the authors propose an approach to track reading progression by utilizing discrete Hidden Markov Models (HMMs). The main concept involves considering eye movement events such as saccades and fixations as HMM observations, while the lines of text serve as HMM hidden states. Initially, eye-gaze fixation measurements obtained from the eye-tracking device are discretized. Each measurement, represented as $\{x, y\}$ coordinates of the eye fixation, is mapped to the corresponding text line. This mapping is based on the condition that the y-coordinate of the fixation falls within the upper and lower boundaries of the text line. Define them as y_{upper} and y_{lower}, respectively, then the condition will be:

$$y_{lower} < y < y_{upper} \tag{1}$$

The authors then use the Baum-Welch algorithm to train the discrete HMM, which provides the state transition matrix A and observation matrix B. These matrices allow us to predict the most probable set of read text lines. The authors demonstrate the proposed Line Detection System (LDS) using commercial eye-tracking devices shows an accuracy of 83.1%. However, the proposed algorithm has certain limitations. It assumes that individuals read the text line-by-line with no more than one line transition or with occasional one line returns. Additionally, the algorithm assumes a zero-mean Gaussian noise for measurement errors.

Another work [4] introduces the use of the Kalman Filter, which is equivalent to the Hidden Markov Model described in [3], with the difference that the hidden state variables have values in a continuous space as opposed to a discrete state space as for the Hidden Markov Model. The authors suggest using the Slip-Kalman Filter to smooth the signal, enabling the detection of when a reader finishes reading a line and moves to the next line by measuring the velocity. The proposed Slip-Kalman Filter approach, using a commercially available eye-tracking device, achieves line detection accuracies ranging from 97% to 98% on a practical dataset where all lines were read once without skipping or repeating. In comparison to the approach [3], the Slip-Kalman Filter relies on stronger assumptions. It assumes that text is read strictly line-by-line without line skipping or re-reading. Furthermore, it also assumes that the measurement error follows a zero-mean Gaussian noise distribution.

3 Dataset

This study utilizes the ZuCo 1.0 dataset [5], specifically the normal reading (sentiment) dataset, as the primary source for generating training data for the proposed model. Data preparation is described in Section 5.1.

The ZuCo 1.0 dataset is a collection of EEG and eye-tracking recordings from 12 healthy adult native English speakers, conducted at the University of Zurich, Switzerland. The participants, 5 female and 7 male, were aged between 22 and 54 years and were all right-handed. Their vocabulary and language proficiency were evaluated using the LexTALE test, which showed an average score of 94.69%. The dataset was designed for studying natural language processing and cognitive processes and included three reading tasks.

The first task involved normal reading of film reviews, where the participants analyzed the emotions and opinions conveyed in positive, negative, or neutral sentences. As a control condition, they were asked to rate the quality of the described movies in 47 of the 400 sentences. In the second task, the participants were presented with sentences containing semantic relations and were followed by multiple-choice questions about the content of the previous sentence in 68 cases. The third task required the participants to focus on specific relation types, such as award, education, employer, founder, job_title, nationality, political_affiliation, visited, and wife. As a control condition, the participants were asked to report whether a specific relation was present in each sentence.

All three tasks involved presenting the sentences one at a time in the same position on the screen. The participants completed the entire reading material over two sessions of 2-3 hours each, at the same time of day. In the first session, they completed Task 2 followed by the first half of Task 1, and in the second session, they completed Task 3 followed by the second half of Task 1. The sentences were presented in the same order to all participants, and a practice round of 3-5 sentences was displayed before each task to familiarize them with the task.

4 Proposed Approach

In comparison with the works [3] [4], where the task is defined as predicting the line number for each fixation, in this work we address the task of classification of individual words of the given text into categories "read" and "non-read" which gives us an idea of reading progress performed by the reader.

To address the task of tracking reading progress, we suggest a 2-step approach: first, we produce a probabilities map from the page's text structure and fixations data by an autoencoder model, described in Section 4.1, which represents the probabilities of each pixel being viewed; at the second step, we interpret the probabilities map from the first step picking the words that were read with high probability.

4.1 Autoencoder

To implement the proposed approach, an "encoder-decoder" model was adopted. The model consists of four convolutional and pooling layers each for both the encoder and decoder, with a bottleneck layer of size 37×26, as shown in Fig. 1.

The encoder layer extracts relevant features from the input data, which in our case are the fixation points and geometrical text composition. The decoder layer transforms these features into a probability map of the input pixels being viewed.

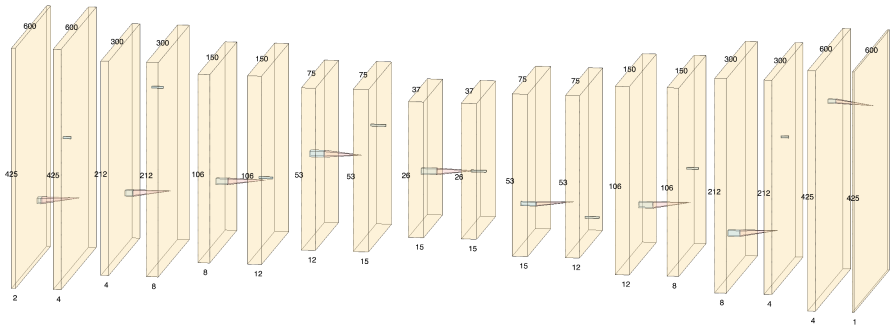

Fig. 1. Autoencoder architecture

4.2 Input

The model takes a 2-channel tensor as input: the fixation points in the first channel and the text structure in the second channel. The text structure is a 600×425 matrix of 0s and 1s. Each element in the matrix corresponds to a pixel in the page image. If the pixel belongs to any word's bounding box, the element is set to 1. Otherwise, it is set to 0. An example of the text and its structure is shown in Fig. 2.

On the other hand, the fixation points are also a 600×425 matrix, where the element at row x and column y represents the fixation point's duration with coordinates x and y in the original page image. Fig. 3 provides a visualization of the text structure and fixation points input.

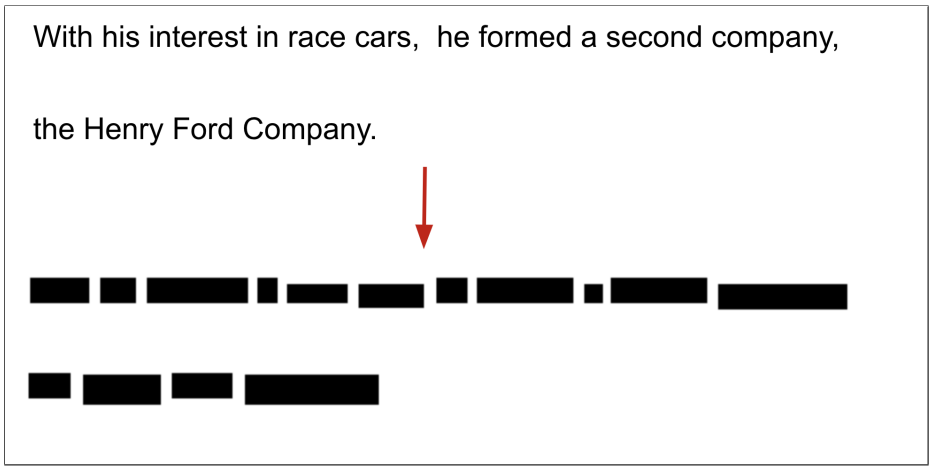

Fig. 2. Example of text and its structure

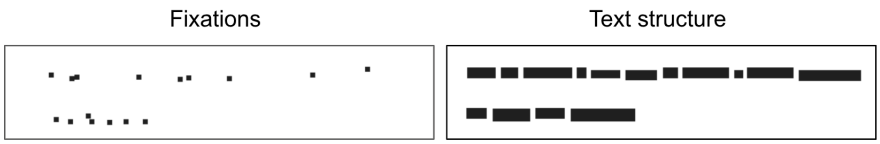

Fig. 3. Fixations and text structure

4.3 Output

The model produces an output in the form of a 600×425 tensor with float values ranging from 0 to 1, representing the probability of each pixel being viewed. We will use this probability map in the future to determine if the word was read. An example of the model's output can be seen in Fig. 4.

4.4 Model Output Interpretation

The model output, in the form of a probability map, can be used to determine whether a word was read or not. To do so, we define a criterion that takes into account the average value of the probability map inside the corresponding bounding box of the word.

Let $p_{i,j}$ be a probability of the pixel with coordinates (i, j) being viewed and $W = \{(x, y) \in \mathbb{Z}^2 : x \in [x_1 : x_2], y \in [y_1 : y_2]\}$ be a set of coordinates of probability map's part inside the bounding box with corners (x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2). Then, we say that the word associated with the set of coordinates W is read if the average value p_W inside the bounding box of the word in the model output is greater than a threshold value of 0.3:

$$p_W = \frac{1}{|W|} \sum_{(i,j) \in W} p_{i,j} > 0.3, \tag{2}$$

Threshold value 0.3 was chosen as the threshold giving the highest metrics. The relationship between the threshold value and mean Intersection-Over-Union (mIoU) and F-score is presented in Fig. 5.

5 Performed work

5.1 Data Preparation

The model's training data was prepared from ZuCo's normal reading task. Specifically, we concatenated three original pages of word boxplots from ZuCo and augmented them with "fake"-words and "fake"-rows, as shown in Fig. 6 and Fig. 7, to create new pages with more information. This approach allowed us to obtain instances of the "non-read" class, which were not present in the original dataset. The result was a dataset consisting of 297 text pages, with a total of 41,107 words per person for each of the 12 individuals.

5.2 Model training

Data preparation, described in Section 5.1, led to the dataset with the size of 3576 elements (pages). In order to train the model, the dataset was split into train, validation and test sets in the ratio of 0.7, 0.15 and 0.15.

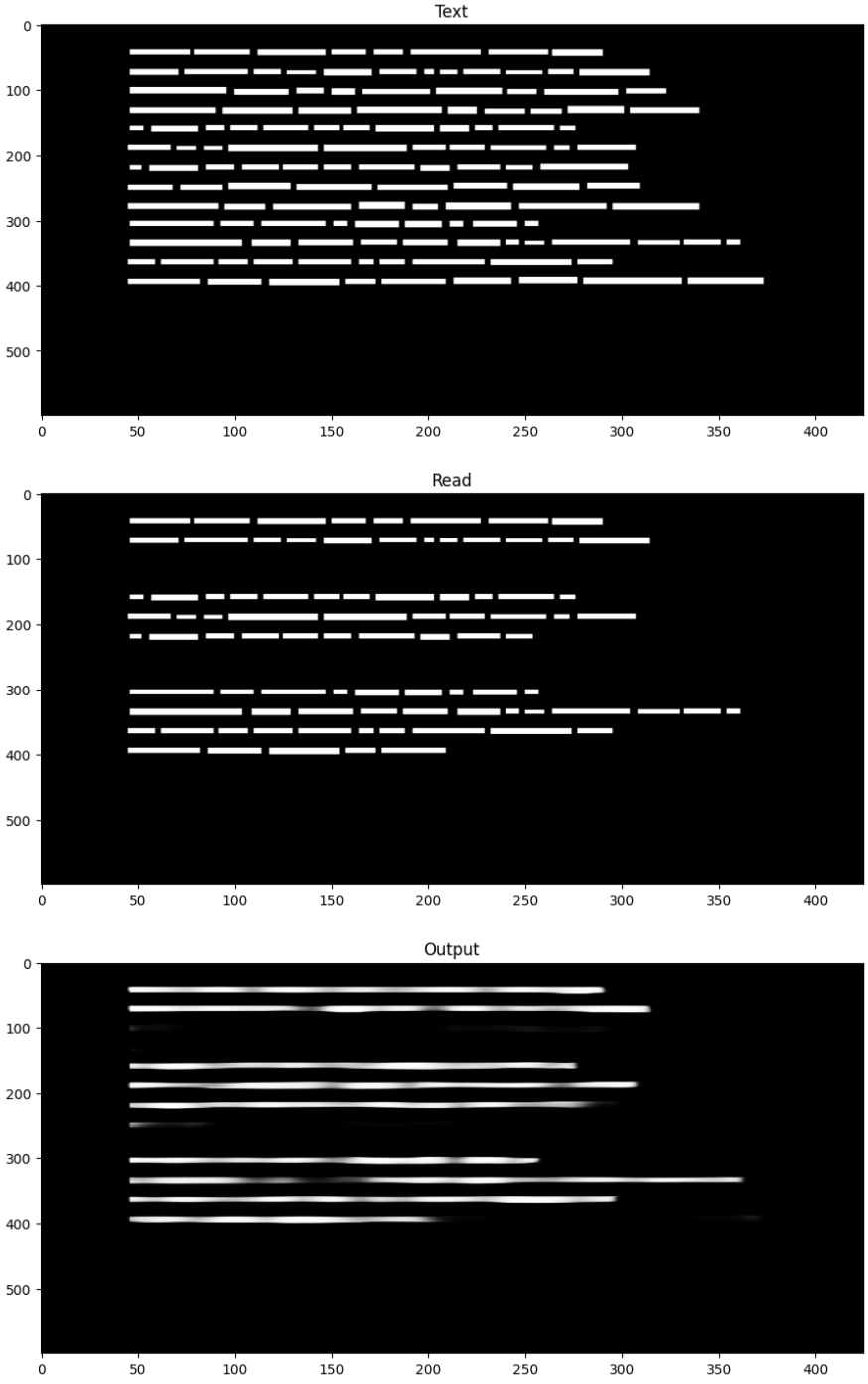

Fig. 4. Model output example

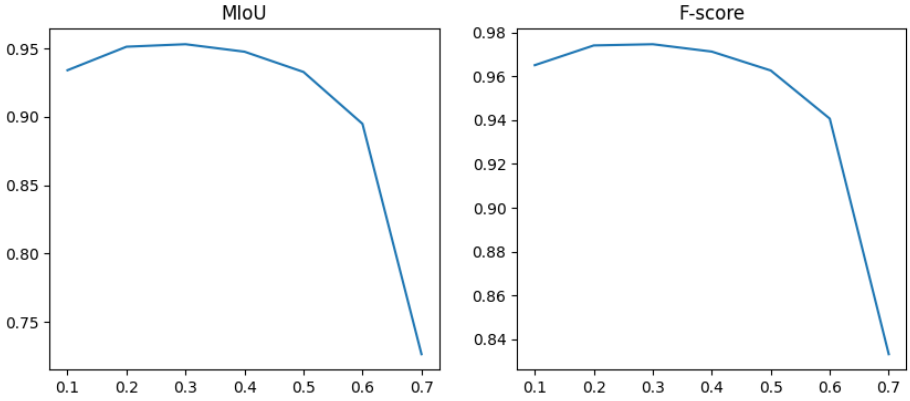

Fig. 5. Dependence of mIoU and F-score on the threshold parameter

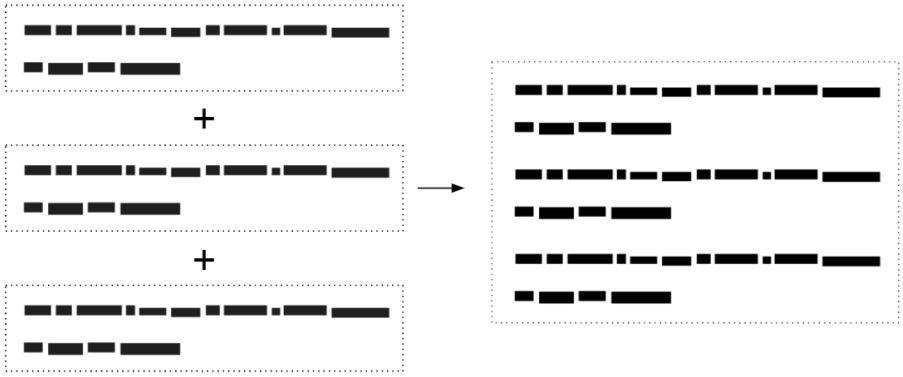

Fig. 6. Pages concatenation

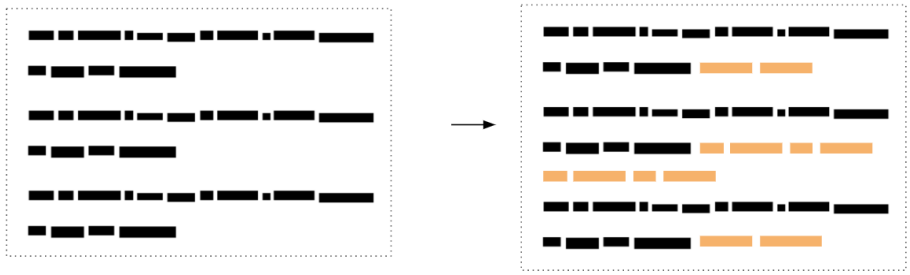

Fig. 7. Pages augmentation

The model was trained using binary cross-entropy loss (BCELoss) and the Adam optimizer. The BCELoss function measures the binary cross-entropy between the predicted output, which is a pixel-wise probability map, and the target output, which is a binary mask indicating the pixels of truly read words.

During training, a batch size of 4 was used and the model was trained for 15 epochs. The learning rate was set to 0.003.

6 Results

To evaluate the proposed approach several metrics were calculated on the test set of size 537 text pages. Those metrics were calculated for each page on the model's prediction of "read" and "non-read" words and were averaged. Formulas used for metrics calculation in the context of the task:

$$\begin{gathered} Precision = \frac{|Predicted\,read\,words \cap True\,read\,words|}{|Predicted\,read\,words|}, \\ Recall = \frac{|Predicted\,read\,words \cap True\,read\,words|}{|True\,read\,words|}, \\ F = \frac{2 * Precision * Recall}{Precision + Recall}, \\ mIoU = \frac{|Predicted\,read\,words \cap True\,read\,words|}{|Predicted\,read\,words \cup True\,read\,words|} \end{gathered} \tag{3}$$

The resulting metrics are presented in Table 1.

Metric	Value
Average Precision	0.9728
Average Recall	0.9848
Average F-score	0.9782
mIoU	0.9587

Table 1. Evaluation metrics for the proposed autoencoder model.

7 Conclusions and Future Work

This study presented a two-step approach for tracking reading progress using an autoencoder model, where a probability map representing the likelihood of each pixel being viewed is produced at the first step; and the read words are determined with the use of an average-probability criterion at the second step.

Additionally, a special data preparation was applied to include both "read" and "non-read" classes.

In the future, the proposed autoencoder model is planned to use in a new Convolutional Recurrent Neural Network (CRNN) model as the producer of input-embeddings for Recurrent Neural Network (RNN), which is conceived to utilize the natural reading's time-dependant features that can be caused by reader's head position, which is usually changes over time. This is believed to significantly improve the reading progress tracking performance to a level suitable for practical applications.

References

1. Pollatsek, Alexander, and Rebecca Treiman, eds. The Oxford handbook of reading. Oxford University Press, 2015.
2. Hyrskykari, A. (2006). Utilizing eye movements: Overcoming inaccuracy while tracking the focus of attention during reading. Computers in human behavior, 22(4), 657-671.
3. Bottos, S., Balasingam, B. (2019). An Approach to Track Reading Progression Using Eye-Gaze Fixation Points. ArXiv, abs/1902.03322.
4. Bottos, Stephen. (2019). A Novel Slip-Kalman Filter to Track the Progression of Reading Through Eye-Gaze Measurements.
5. Hollenstein, N., Rotsztejn, J., Troendle, M., Pedroni, A., Zhang, C., Langer, N. ZuCo, a simultaneous EEG and eye-tracking resource for natural sentence reading. Scientific data, 5(1), 1-13. (2018)