Identification of Reader Specific Difficult Words by Analyzing Eye Gaze and Document Content

***Abstract*—This paper presents an approach for identifying reader specific difficult words while someone is reading a textual document. The work is motivated by the need of developing human-document interaction systems, in general and creating person-specific online educational content, in particular. Eye gaze information gives person specific behavior whereas textual content is analyzed to get general linguistic aspect of the document content. These two pieces of information are fused together through machine learning algorithms to identify the set of difficult words for a particular reader reading a particular document. An annotated dataset has been created where each word in a document is marked with its bounding box information and each reader identifies a set of difficult words while reading the document. The dataset consists of sixteen documents and each document is read by five subjects. The method is evaluated through recall-precision analysis. The impressive precision at high recall attests the feasibility of building a practical application based on this research. The experiment further brings out several interesting facts about human reading behavior.**

Keywords—Content personalisation, Reading behavior analysis, Difficult words, Eye tracking, Linguistic analysis, Machine learning, Dataset and evaluation.

I. Introduction

In the field of human-document interaction, content personalization is an important area of research. One task in this area is to create reader specific content because same content may not be always convenient for all readers even if they come from same group (age wise, education wise, or likes). For instance, for a particular document one reader may find certain set of words difficult to understand whereas another reader may find another set of words difficult. Content personalization would be easier if we could capture this reader specific reading difficulty.

This paper addresses this problem and presents an approach for identifying reader specific difficult words in a document. The method integrates two kinds of analysis namely, reading behavior analysis and document content analysis. The reading behavior analysis is done using the eye gaze data recorded while someone is reading a document and some linguistic aspects are used to do the content analysis.

In the field of applied psychology, there are several studies on reading behavior [1, 2]. With the advent of eye tracking technology, several researchers are now using eye gaze data and reporting their experiments. Eye tracking has been used for predicting reader's language comprehension ability [3, 4], identification of reader as a biometric task [5], or associating reading behavior with the readability of a document [6]. The readability of a given document is defined by a score that recommends the document for a particular group of readers. However, readers' ability, within a group, may vary and capturing of this variation is required for effective content personalization. This article presents a novel approach for capturing this reader specific reading ability at the word level.

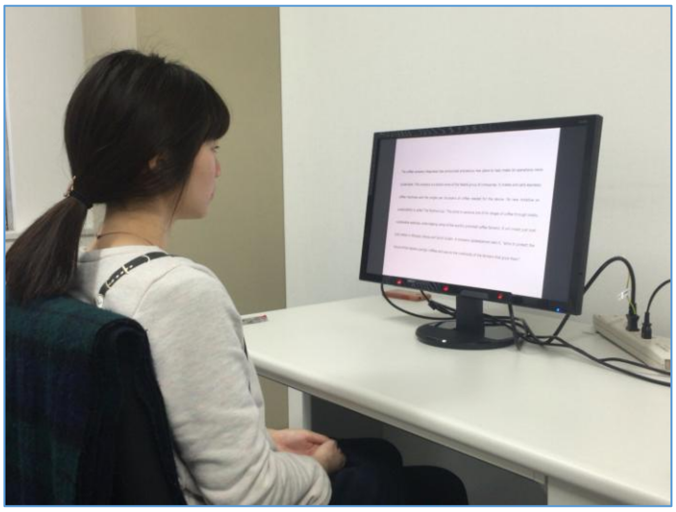
Fig. 1. Recording session*:* one subject reading one document. The eye tracker is fixed at the bottom of the screen. The document is displayed on the screen. Documents are consisting of seven to eight lines (7 lines for the document on the screen). Each reader was told to write a small summary after reading a document. This was told to ensure that the reader will try to understand the text well.

The method presented in this paper assumes a fixed layout of the document where position of each word (in form of upright rectangular bounding box) is known. Reading of a document gives us a series of eye gaze points. On receiving these points, the algorithm associates each eye gaze point to a word in the document. Forward and backward (regression) reading are labelled. Next, the number of forward and backward gaze points is calculated for each word and time periods spent for forward and backward reading are recorded for each word. Two linguistic features are considered for each word. They are namely, frequency of word (more specifically, frequency interval) and number of letters in a word. All these features are then used for identifying difficult words using off-the-shelf classifiers like neural networks, support vector machines, etc.

The distinct contributions of this research are (i) formulating a novel method for capturing reader specific

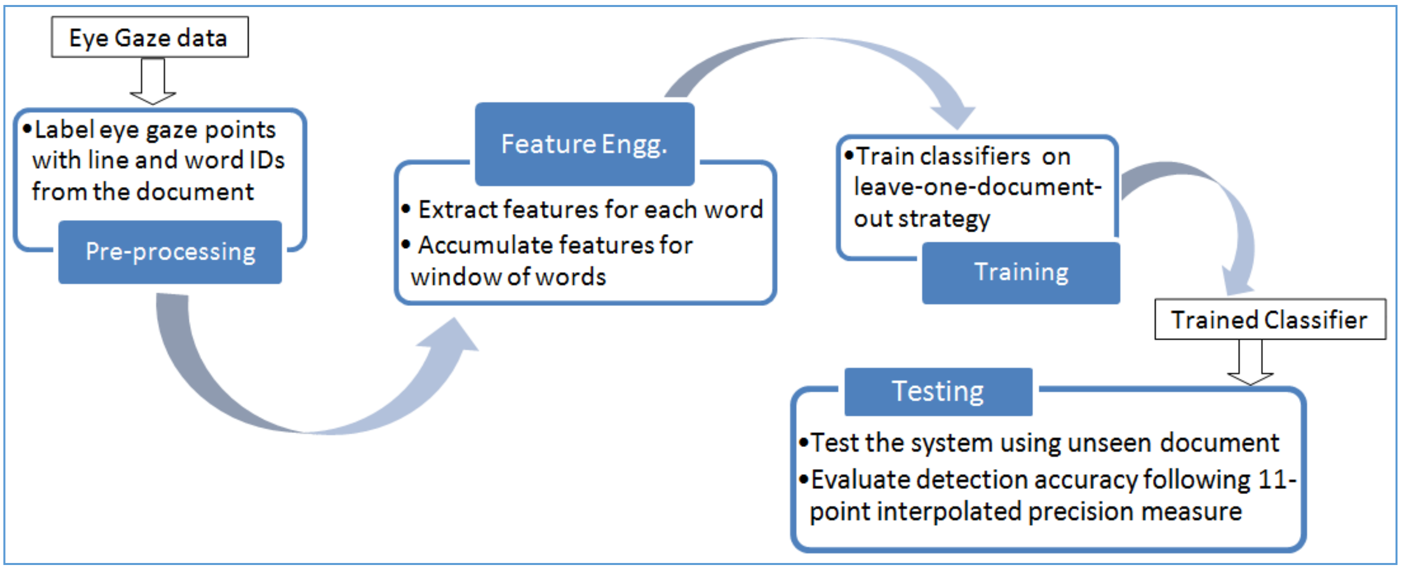
Eye Gaze data
•Label eye gaze points with line and word IDs from the document
Pre-processing
Feature Engg.
• Extract features for each word
• Accumulate features for window of words
•Train classifiers on leave-one-document-out strategy
Training
Trained Classifier
Testing
•Test the system using unseen document
•Evaluate detection accuracy following 11-point interpolated precision measure

Fig. 2. An upper level view of the system for detection of difficult words.

difficulty, (ii) extracting meaningful features from eye gaze data so that reading behavior can be properly analyzed for the given purpose, (iii) evaluation on a real and sharable dataset, and (iv) giving some meaningful insight into analysis of human reading behavior supported by experimental evidences.

The rest of the paper is organized as follows. Section-II explains the eye gaze recording process and the pre-processing step. The feature engineering part is explained in Section-III. Section-IV explains the classification process. The experimental protocol including dataset and evaluation metric is presented in Section-V. Results and related discussion are presented in Section-VI. Section-VII concludes the paper.

II. Recording and Pre-processing of Eye Gaze Data

The reading activity is recorded by a stationary eye tracker which has a variable frame rate with a minimum frequency of 30 Hz. Figure 1 shows the reading set up. The eye tracker returns a set of gaze points $\{g_1, g_2, g_3\ldots\}$ where each point represented by $<t_i, x_i, y_i>$ where t_i is the time stamp, x_i and y_i are x and y coordinate values of the gaze point g_i (top left corner of the screen is considered as the origin, x is increasing horizontally from left to right and y is increasing vertically from top to bottom).

Figure 2 presents an upper level view of the system presented here. The system consists of four modules: pre-processing, feature engineering, training and testing. While the other modules are explained in Section III and IV, the pre-processing techniques are described here.

The salient aspects of our reading behavior need to be discussed first for better understanding of the pre-processing and subsequent steps. Reading behavior is not a uniform continuous movement but consists of a sequence of brief stops (fixation) and jumps (saccade). Figure 3 shows a usual reading behavior. The area of a circle (blue or red) is proportional to the number of gaze points or alternatively, the amount time spent around the center of the circle (fixation). The circle is colored red if from that position the line of reading is changed. The lines joining circles show the order of reading (saccades).

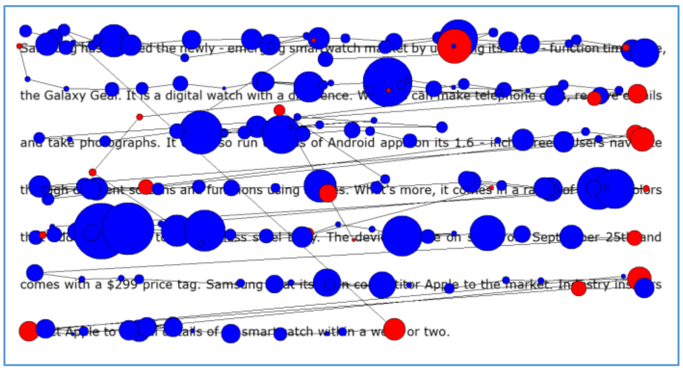

Fig. 3. A usual reading behavior: fixations are shown by circles. Fixations having more eye gaze points are larger in size. Fixations are marked red to indicate that the reader has changed line at that point.

The main task of the pre-processing deals with assigning each gaze point to its corresponding word. However, this is indeed a non-trivial task due to vertical error [7, 8]. The following sections describe the pre-processing steps through which the eye gaze data go through.

A. *Finding line boundaries in the gaze data*

The document lines are read from left to right therefore while a line is read, a steady increase in x-coordinate is observed. After finishing the current line, reading of the next line is signaled by a noticeable change in x-value because of the line width. Therefore, if a change in two successive x-coordinates exceeds some threshold (τ, which depends on the line width), a new line is assumed. At this step, each gaze point is tagged with a line number (l_i), i.e. $< t_i, x_i, y_i, l_i >$. The line number counter starts with 1 (reading starts from the first line) and is incremented when $|g_i.x - g_{i+1}.x| > \tau$. Note that because of vertical errors, y-values of individual gaze points are not very reliable in identifying lines.

Line numbers as identified above based on x-values are often affected by errors due to backward reading. When a reader goes backward and rereads a part of the current line or another line, then the change in x-coordinates may exceed the threshold. Backward reading for a line is often observed. After finishing reading of the whole document, many readers re-read some parts of the document. This is natural when someone attempts to understand the content. Therefore, the number of lines (say, K) obtained at this stage is normally higher than the actual number of lines (say, L) in a document. For fixing these errors y-coordinates are considered as follows.

The y-median value is calculated for each identified line. Let $y_{\min}$ and $y_{\max}$ are the lowest and the highest y-median values. Therefore, y-gap between two successive lines can be roughly understood by $y\text{-gap}=\frac{y_{\max}-y_{\min}}{L}$. Now, two lines are given same line number if their y-medians lie within a fraction (0.75) of y-gap. Following these process, the line numbers are renumbered. Since the documents used in this experiment are prepared with large (more than triple-line spacing) inter-line gap, vertical errors are nicely corrected by the above algorithm. It is also to be noted that just before setting eye on the first word to start reading there are some inconsistent eye gaze points due to random eye movements. These points are characterized by random x- and y-coordinates and are removed.

B. *Assigning words to the gaze points*

Each word in a document is tagged with its position in terms of up-right rectangular bounding box and the line number. Each eye gaze point is also tagged with line number. Therefore, if the line number of a gaze point matches with that of a word and the x-coordinate of the gaze point falls within x-extend of the word's bounding box, then an immediate association between the gaze point and the word is found. For cases where the x-value of a gaze point does not fall with bounding box of any word of the corresponding line, x-coordinates of the centroids of word bounding boxes are considered to tag the gaze point with its nearest word. At this stage, each gaze point is represented by $< t_i, x_i, y_i, l_i, w_i >$.

C. *Tracing of forward or backward reading*

Backward reading is a predominant trend. This is associated with our effort for understanding the content. Whenever, we have difficulty in understanding a word or a phrase we go back and re-read a part. We may back read a few words in the current line, we may read few parts of any of the previous lines. Even after finishing a paragraph, many readers do back reading of some portion. We put special emphasis on back reading as this has strong association with words or phrases one reader find difficult.

A word normally associates many gaze points. When the word is read for the first time, all its gaze points are labelled as forward (F). In subsequent reading is the same word reappears then all the corresponding gaze points are labelled as backward (B). This classification helps to computer the number of forward and backward gaze points incident to a particular word. At this stage, each gaze point is represented by $<t_i, x_i, y_i, l_i, w_i,$ F/B>. This representation is passed to the subsequent module where feature engineering is done.

III. Feature Engineering

Two types of features are computed for each word of a document namely, (i) reading behavior and (ii) linguistic aspects. The features are explained below.

A. *Features from reading behavior*

Four features are computed from the reading log. Previous section labelled each gaze point with its corresponding word number and this representation helps word wise feature computation. Both forward and backward reading were considered for feature engineering. Four features as described below are computed for each word:

Number of forward gaze points ($\mathbf{P_f}$): Number of gaze points labelled with 'F' incident ~~on~~ to a word when the word is read for the first time (forward reading).

Number of backward gaze points ($\mathbf{P_b}$): Number of gaze points labeled with 'B' incident on a word when the word is re-read (regression or backward reading).

Duration of forward reading ($\mathbf{D_f}$): Duration of i-th gaze is calculated as difference between time stamps of i+1st gaze and i-th gaze points. Total duration for all p_f number of gaze points is computed and labeled as feature D_f.

Duration of backward reading ($\mathbf{D_b}$): Total amount of duration during re-reading calculated from the gaze points labeled as 'B'.

B. *Linguistic Features*

Two simple linguistics features are computed for each word: (i) the number of letters in the word ($\mathbf{F_c}$) and (ii) the relative occurrence frequency of the word are noted ($\mathbf{F_f}$). The frequency list (of project Gutenberg) of Wiktionary[1] is used to compute F_f. An eleven-point (0, 0.1, 0.2, …, 1.0) measure is used to indicate the frequency of a word. If a word belongs to the rank list [1-1000], its F_f is assigned to 0. The words belonging to the rank list [1001-2000] get F_f = 0.1 and finally, words having ranks more than 10,000 get 1.0 value for F_f.

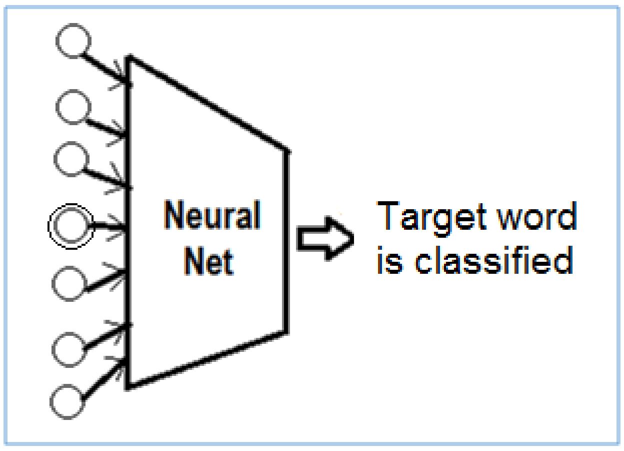

Fig. 4. A neural net for classifying words as difficult or easy (target word is marked with double circles).

[1] https://en.wiktionary.org/wiki/Wiktionary:Frequency_lists

IV. Formulating Difficult word detection as a classification problem

After feature engineering, each word (W_i) is represented by a feature vector $<P_f^i, P_b^i, D_f^i, D_b^i, F_c^i, F_f^i,>$. As context plays a great role in reading, initially we tried with a recurrent model but because of small amount of data, recurrent network could not learn properly. Next, a simpler feedforward network as shown in Fig. 4 is designed to classify each word as difficult or easy (2-class problem). The net has two hidden layers of 8 nodes in each layer. A context of ±3 words around the word (W_i) to be classified is considered.

The classification accuracy achieved by the above network is quite poor as shown in the next section. Our reading behavior is responsible behind this poor result. The assumption that the features will have high values for difficult words is often wrong as the neighboring words (even many of them are highly frequent including some stop words too) also get high feature values (some times more than the difficult word). This explains one aspect of our reading behavior that we often try to understand the meaning of a difficult word from its context and therefore, we go back or forward to read neighboring words and subsequently, the features related to reading behavior (as described in section III-A) get high values for the neighboring words irrespective of their difficulty. Considering this aspect we modify our classification model as described next.

A. Classification of Phrases

Instead of trying to classify each word, our next model considers a window of n words and classifies whether that window corresponds to reading difficulty (i.e. consists of one or more difficult words). WIN_i considers n words (W_i, W_{i+1}, …, W_{i+n-1} and feature vector for WIN_i is computed by summing the values for a particular feature for all n words. For instance, P_f^i of WIN_i is computed as $\sum_{j=i}^{i+n-1} P_f^j$. Overlapping windows are considered by sliding each window by one word towards right. So, if there are N words in a document, N-n+1 windows result in. WIN_i is labeled difficult if one or more words in it are labeled as difficult.

By classifying each window as difficult or easy we actually capture difficulty at the phrase level (phrase length depends on the value of n. This is closer to our reading behavior as mentioned before to explain why a word level classification model does not give good result. We designed three types of classifiers namely, feed-forward neural network, SVM-based and Random Forest for classification of each window of words as difficult or easy (2-class problem).

Experiment shows that higher values of n do not give good classification accuracy. As shown next the best result is obtained for n = 2, i.e. when window of 2 words is considered. This again corresponds to our reading behavior that while reading we read small number of words together.

B. Identification of difficult words in each difficult phrase

As the task is to identify difficult words, we investigate the difficult phrases further to identify difficult words inside the phrases. If WIN_i is classified as difficult, we check each word in WIN_i and identify a word as difficult by considering their occurrence frequency. If a word does not belong to the list of 4000 mostly used words, then we label it as a difficult word. Similar concept has been used before in largely used readability score following Dale-Chall formula [9]. A window of words labeled may have all words used frequently (i.e. within top 4000). In such cases, an evidence based approach is followed. The word W_j is labeled as difficult if at least one more evidence is found, i.e. W_j is found in another WIN_k labeled as difficult.

V. Experimental protocol

This section describes the dataset, evaluation metric, and the baseline for comparison.

A. Dataset

A dataset has been developed for evaluating the task of identifying reader specific difficult words in documents. A stationary eye gaze recorder working at minimum 30 Hz is used. Five subjects participate in data collection process. The participants are university students (4th year bachelor to 2nd year master's). For all five subjects English is the second language.

Sixteen news article documents are considered. Documents' lengths vary from 121 to 197 words (average is 147). Documents have a fixed layout and each word is tagged with their positional information in terms of up-right minimum sized rectangular bounding boxes. Numbers of lines in the document vary from to 6 to 9 (average is 7.6 lines). Lines are spaced by about 3 lines as to minimize vertical error of the eye tracker.

Each subject was told to write a summary after reading each document as to ensure that subjects would try to read each document carefully to understand the content. After completing the reading of a document a subject is asked to do the following (i) mark difficult words on the print out of the document, and (ii) mark difficult words in a list of infrequent words taken from the document. The step (ii) is introduced not to miss any difficult word which might be (mis-) understood by using its context. We deal with these mark words as difficult words and generate reader specific word labels (difficult/easy) for each of the sixteen documents.

B. Evaluation

Given a document and a reader, the task is to detect which words of the document are difficult for the reader. We followed a leave-one out strategy in evaluating the detection performance. Out of sixteen documents, fifteen are used to train the system and its performance is noted on the sixteenth one. For a reader this evaluation is done sixteen times so that each document is treated once as a test document. Average over these sixteen runs gives the performance for the reader.

The words detected as difficult are sorted based on their difficulty. The classifiers classify a word as difficult with a probability which is used for ranking of words. The highest probable word is ranked 1. The detection accuracies are measured by 11-point interpolated average precision [10]. The interpolated precision p_{interp} at a certain recall level r is defined as the highest precision found for any recall level $r' \geq r$: $p_{interp}(r) = \max_{r' \geq r} p(r')$. The result is presented by recall-precision curve. This evaluation shows two things: (i) Precision at 1.0 recall shows how the system performs to detect all the difficult words. A low precision at 1.0 recall shows that the system falsely detects many words as difficult, and (ii) Precisions at lower recall values show that whether high ranked "difficult" words detected by the system are really difficult. Sometimes it is advantageous if the system detects small number of words as difficult with very high accuracy.

C. *Baseline*

The baseline is formed by considering the relative occurrence frequencies of the words in a document. Words are sorted in ascending order of their relative frequencies and ranked. The top ranked word is the word having lowest relative frequency. The words (some proper names or transliterated words) which are not found in the frequency list are treated as words of having very low frequencies. They are sorted following their appearances in the document. Such a word appeared first in the document is ranked 1.

VI. RESULT AND DISCUSSION

At first the performance of the baseline method (explained in Section V-C) is evaluated. It is found that such a method can achieve an average precision about 0.28 at 1.0 recall. Precision can go up to 0.32 for recall equals to 0.6 or less. Next, we evaluated the neural network presented in Figure 4. The net classifies each word as difficult or easy. Context of six words (three previous words and three subsequent words) is used. Precision at 1.0 recall was computed and found to be only 0.2. The reason behind such poor result (less than the baseline) was investigated and found that often the previous or subsequent words of a true difficult word are wrongly labeled as difficult. For instance, one reader labels "insolvency" as a difficult word while reading "to declare insolvency". In such a case, the net wrongly classifies words like "to" and "declare" as difficult thereby reducing the precision. Note that these words do not appear in the top-ranked words when words are ranked based on their relative frequencies in the baseline method. Although the net considers context but still it fails to identify difficult words with high precision.

Next we evaluated our method described in Section IV-A. Here window of words is formed and classified. As mentioned before we used three types of classifiers for classifying each window of words. We used Keras implementation of feed-forward neural network and Scikit-learn implementations for SVM and Radom forest based classifiers. Default parameter values are used for the classifiers. The value of n (i.e. window size) is varied from 2 to 11 and we got best result for $n = 2$. We noted that all the three classifiers perform similarly (the SVM-based classifier being slightly better) achieving an average precision of about 0.40 at 1.0 recall. Consideration of 3-word window size shows average precision of about 0.39 at 1.0 recall. Results for different values of n (from $n = 2$ up to 9) are shown in Figure 5. We noted that this trend is dominant in the subsequent experiments too where we explore hyper parameter optimization as presented next. Observation at this stage corresponds to our reading habit that while reading our gaze covers (i.e. we read) two or three words together.

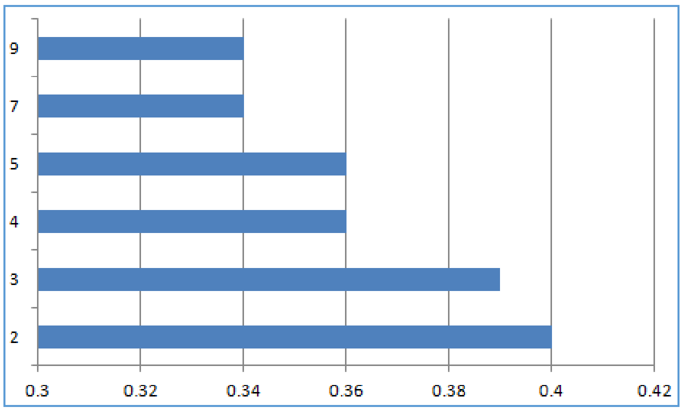

Fig. 5. Average precision at 1.0 recall for different window sizes. Consideration of 2-word window gives the best result.

A. *Hyper-parameter Optimization*

The above results were obtained by using the default values provided by the off-the-shelf implementations as mentioned before. We next explore hyper-parameter optimization to check whether precision can be further improved. This effort gives us noteworthy success in case of SVM-based classification. SVM implementation of scikit-learn makes use of fourteen hyper-parameters and we found that three of these fourteen parameters play important role in changing results. These three parameters are: (i) c, which is the cost function regularization factor, (ii) gamma, which is the parameter of a Gaussian Kernel to handle non-linear classification (a small gamma means a Gaussian with a large variance, which increases support vector spread), and (iii) kernel (rbf, polynomial or linear). The default values for these parameters are: c = 1.0, gamma = 0.1, kernel = rbf. We explore with various values for these three parameters ranging values for c taken from [0.001, 0.01, 0.1, 1.0, 10, 100}, gamma from {0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0, 10, 100} and kernel from {rbf, poly, linear}. The following set of values gave the best result (average precision is 0.66 at 1.0 recall): c = 0.01, gamma = 0.1 and kernel = rbf.

The recall-precision curve is shown in Figure 6. The highest precision 0.75 (at 1.0 recall) is obtained for subject no. 3. The average case precision is 0.66 at 1.0 recall. When this result is compared with the baseline result we note that use of eye gaze data can improve the accuracy for detection of difficult words significantly (improvement is 136%, from 0.28 to 0.66).

Analysis of errors reveals that sometimes words in phrase are relatively easy if considered in isolation but in phrase they might appear as difficult for some readers. For example, one subject labeled all three words in "*behind the scenes*" as difficult but the system fails to capture them as all the three words are very frequent in use. So though reading behavior shows higher feature values for this phrase, word frequency-related feature dominate and wrong classification results in. The same analysis holds when another subject labeled all words in "*would be thief* " as difficult but the system fails to detect them. It is understandable that in such cases instead of the individual words, the phrases might be unknown to some subjects. A phase-level annotation might be helpful for such cases.

In some cases, reading behavior and linguistic features both show potential for a word to be difficult for a reader but as the reader actually did not mark it difficult, a classification error is reported. For instance, while reading the phrase "*has filed for bankruptcy*" one subject spends substantial time to read the phrase and the relative frequency of "bankruptcy" is also low. Hence, the system marked "bankruptcy" as difficult but the subject did not label it so. It may happen that the subject, with an effort, finally understood its meaning and decided not to label the word as difficult. However, the system has no clue to capture this aspect.

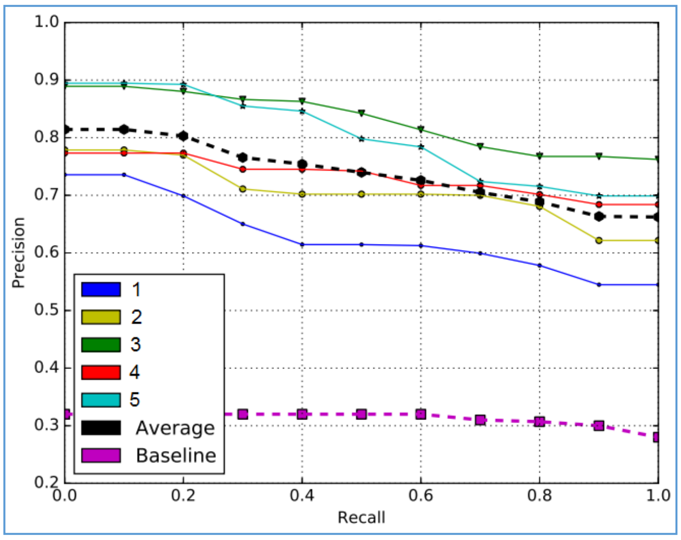

Fig. 6. Recall-precision curves: Reader specific curves, overall average result and the baseline result are shown in different colors.

VII. CONCLUSION

This paper presents an in-depth study to detect reader specific difficult words in a document. The experiment makes use of both reading behavior and linguistic features. A relative frequency-based baseline shows that use of reading behavior is significantly helpful alone with linguistic feature for achieving high detection accuracy. Hyper-parameter optimization is another area that gives meaningful improvement [11].

The future research on this problem would surely address two aspects: (i) exploring deep learning approaches and (ii) replacement of difficult words/phrases for content personalization. Our initial experience behind involving deep learning techniques shows that we need more data to have meaningful results but such a frame work would be more attractive than what we have reported here as it could avoid the feature engineering step.

For content personalization the present work can be used along with a paraphrase generation module [12]. Either the detected difficult words can be replaced by (in-context) synonyms having higher relative frequencies or a phrase can be replaced by a suitable paraphrase to make the content more convenient for the reader.

REFERENCES

[1] K. Rayner, "Eye movements in reading and information processing: 20 years of research," Psychological bulletin, 124(3): 372–422, 1998.

[2] T. Malsburg, R. Kliegl, S. Vasishth, "Determinants of Scanpath Regularity in Reading," Cognitive Science 39(7): 1675-1703, 2015.

[3] O. Augereau, H. Fujiyoshi and K. Kise, "Towards an Automated Estimation of English Skill via TOEIC Score Based on Reading Analysis," Int. Conf. Pattern Recognition (ICPR), 2016.

[4] H. Kang, "Understanding online reading through the eyes of first and second language readers: An exploratory study," Computers & Education, 73:1–8, 2014.

[5] A. AbdelWahab, R. Kliegl, and N. Landwehr, "A Semiparametric Model for Bayesian Reader Identification," EMNLP: 585-594, 2016.

[6] A. Mishra, D. Kanojia, S. Nagar, K. Dey, and P. Bhattacharyya, "Scanpath Complexity: Modeling Reading Effort Using Gaze Information," AAAI: 4429-4436, 2017.

[7] A. Yamaya, G. Topić, P. Martínez-Gómez, and A. Aizawa, "Dynamic-Programming–Based Method for Fixation-to-Word Mapping," In: Neves-Silva R., Jain L., Howlett R. (eds) Intelligent Decision Technologies. Smart Innovation, Systems and Technologies, vol 39. Springer, Cham, 2015.

[8] C.L. Sanches, O. Augereau, and K. Kise, "Vertical error correction of eye trackers in nonrestrictive reading condition," IPSJ Transactions on Computer Vision and Applications, 8:7, 2016.

[9] E. Dale and J. Chall, "A Formula for Predicting Readability," Educational Research Bulletin. 27: 11–20, 1948.

[10] C.D. Manning, P. Raghavan and H. Schütze, "Introduction to Information Retrieval," Chap. 8 on *Evaluation in Information Retrieval*, Cambridge University Press, 2008.

[11] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," in Journal of Machine Learning Research 13: 281-305, 2012.

[12] S. Zhao, X. Lan, T. Liu and S. Li, "Application driven statistical paraphrase generation," in Proc. 47th Annual Meeting of the ACL and the 4th IJCNLP, pages 834–842, Singapore, 2009.