Predicting Knowledge Gain during Web Search based on Eye-movement Patterns

Fakultät für Elektrotechnik und Informatik

Institut für Verteilte Systeme – Fachgebiet Visual Analytics

Leibniz Universität Hannover

Bachelorarbeit

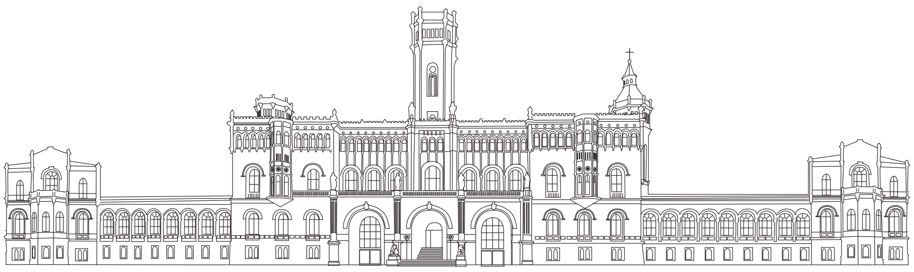
submitted for the degree of

Bachelor of Science (B. Sc.)

by

Ahmad Khawatmi

Matriculation Number : 10006556



First Examiner: Prof. Dr. Ralph Ewerth

Second Examiner: Prof. Dr. Sören Auer

Supervised By: M. Sc. Wolfgang Gritz

October 10, 2022

Erklärung der Selbständigkeit

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden, alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind, und die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt habe.

Hannover, October 10, 2022

Ahmad Khawatmi

Abstract

The content on the internet is expanding exponentially, and the virtual space has become a messy place. Therefore, acquiring information to fulfill the learning need is a difficult task.Search as Learning (SAL) is a new domain that investigates the importance of the learning process and supports individuals in acquiring information. Therefore, a solution to make obtaining information easier for knowledge seekers from a web search. Prior work in this field focused extensively on resource data (e.g., text and multimedia resources) and behavioral data (e.g., search interactions) to make a knowledge gain (KG) prediction during a web search. However, eye movement and reading pattern data are yet to be explored. Thereby, in this work, we introduce a set of features related to eye movements that would help us predict knowledge gain based on the reading pattern of the participants. For this purpose, we relied on data from a prior work-study, in which 114 participants had to acquire information about the foundation of lightning and thunder from a web search. We used a cutting-edge approach for the evaluation. Moreover, we extended with a word-level mapping to eye fixations of web pages, unlike prior work that attempted to rely on the eye's central vision to map the eye fixations. Experimental results demonstrate the ability to predict knowledge gain based on the reading pattern and eye movements.

Contents

List of Tables

List of Figures

Acronyms

Adaboost Adaptive Boosting. 13, 43

CSV Comma Separated Values. 12, 39

DT Decision Tree. 13, 43

FN False Negative. 42

FP False Positive. 42

HTML Hypertext Markup Language. 12, 36

I-DT Dispersion-Threshold Identification. 13, 37

IR Information Retrieval. 1

JSON JavaScript Object Notation. 12, 27

K-NN K-Nearest Neighbour. 13, 43

KG Knowledge Gain. 1, 3, 5, 6, 8, 42, 44

KS Knowledge State. 1, 5

ML Machine Learning. 13

MLP Multi-layer Perceptron. 13, 43

OCR Optical Character Recognition. 12

PCC Pearson Correlation Coefficient. 22, 40

RF Random Forest. 13, 43

SAL Search as Learning. 1, 45

SERP Search Engine Result Page. 5, 6, 8

SI Search Interaction. 6

SMI SensoMotoric Instruments. 36

SVM Support Vector Machines. 13, 43

TN True Negative. 42

TP True Positive. 42

TSV Tab Separated Values. 37

URL Uniform Resource Locators. 37

1 Introduction

With the quick technological advancement and the rapid growth of the worldwide web, our life has become dependent on this virtual space and the information that it stores. The internet enables us to find and access any information we need with only a few clicks. As a result of this fast growth, it has become a messy place with an overloaded amount of information [30]. Search engines, on the other hand, don't make the process easier because they rely on an Information Retrieval (IR) system with limited capabilities that only do simple tasks by looking for topic relevance in the documents, and ignore others that require complex search strategies, which are beyond simple lookups, such as learning [23]. Different intents can be distinguished from searchers' web search queries, and they are navigational, transactional, and informational [7, 64]. Searchers' goal in the navigational intent is finding a specific website, whereas looking for something to buy on the web describes the transactional intent of the searchers. Acquiring new information and learning a new skill represents the informational intent and it is a process in which the state of the searcher's knowledge gets changed due to the seeking and engagement with information. A modern IR system is required to fulfill the needs of web searchers with a learning purpose. The domain of Search as Learning (SAL) investigates the importance of the learning process on the internet and tries to detect the informational intent as it happens to facilitate acquiring information and support individuals by learning. It also focuses on predicting Knowledge State (KS) and Knowledge Gain (KG) of users after a web search and ranking web page results according to learning goals [31].

Prior work [2, 42, 64] has explored information acquisition of participants by looking into various aspects of learning to predict the knowledge gain during a web search. Gritz et al. [27] focused on exploring the relationship between textual complexity and knowledge gain. On the other hand, Otto et al. [42] investigated the influence of multimedia resources on knowledge gain prediction. Behavioral and search interaction data were used extensively to predict knowledge gain. Ran et al. [65] investigated the outcome of the search interaction on knowledge gain, and Yu et al. [64] attempted to look into the browsing behavior of the participants to measure knowledge gain prediction. Last but not least, Bhattacharya and Gwizdka [2] have explored different aspects of knowledge gain prediction by looking into the eye movements of the participants to see their influence on the prediction. However, this domain is not explored broadly. Therefore, in our work, we attempt to look deeper into the domain by trying to investigate not only the eye movements of the participants but also their reading patterns. We will analyze how knowledge is acquired based on the way the eyes move while reading the textual content. We also present an accurate word-level method to map the eye fixations on websites' textual content that aims to understand better the reading pattern of participants and their eye movements. Moreover, better prediction for knowledge gain.

2 Related Work

Research has shown [2, 27, 42] that it is feasible to model and predict a user's KG by combining various features recorded during a search on the web. It has investigated the relationship between Knowledge Gain prediction, and a set of features based on text, multimedia, behavioral, search interaction, eye-tracking, and reading patterns. Our research focuses on two domains of prior work: (1) The influence of various features on KG and (2) Modeling and predicting KG based on eye-tracking data and reading patterns.

2.1 Knowledge Gain

Acquiring new information from web searches has been one of the most crucial tasks on the internet. Researchers [2, 24, 64] have extensively explored different approaches to measure the amount of gained information by using various means and analyzing multiple characteristics during the search, and it is called Knowledge Gain. Every researcher has a slightly different definition of Knowledge Gain. However, all of them share the same goal: evaluating the amount of new information the participant acquired during their web search while learning about their task.

Gadiraju et al. [24], Ran et al. [65], and Yu et al. [64] defined Knowledge Gain as the difference between the score of the post-session test and the calibration knowledge score using Behavioral and Search characteristics.

Moraes et al. [40] followed the same measurement criteria for evaluating the participants' vocabulary change by comparing the Knowledge Gain before and after the study experiment.

On the other hand, Gritz et al. [27] defined a range of values from -10 up to 10 that indicates the learning outcome's efficiency. The gain is calculated by subtracting the post-session knowledge test score from the pre-session test score. They counted mainly on Resource data to compute the prediction of the Knowledge Gain.

Kalyani and Gadiraju [36] measured Knowledge Gain change by giving participants a pre and post-test, which consisted of the same questions that test the level of participants' cognitive abilities during the web search task.

Furthermore, Bhattacharya and Gwizdka [2] decided on an assessment test without a time limit to measure Knowledge Gain, in which participants could write in a free-recall manner to estimate their knowledge. Participants took two tests, one before and one after the session, and the difference between the two scores was taken as the final value. The KG prediction was primarily based on the participants' reading patterns and gaze data.

2.2 Resource Data and Knowledge Gain

A wide range of resource data has been investigated to study its impact on KG, especially Textual data.

Gritz et al. [27] investigated the influence of the textual features on measuring Knowledge Gain during a web search and analyzed its impact based on the complexity of the text on predicting KG. He extracted eight different types of features to assess the complexity of the text. Textual complexity features types varied between Syntactical, Readability scores, Part of speech (POS), Lexical richness, Group of Lexical variation, Group of Lexical sophistication, Group of Syntactic constituents, and Group of Connectives. Multiple classifiers were implemented to analyze the large set of text-based features. Results obtained from his analysis of the textual complexity data of the visited website pages during the study showed the importance of text complexity features on Knowledge Gain prediction. Moreover he included that web pages with the lowest textual complexity are the most important for the classification result.

Similarly, Eickhoff et al. [17] tried to predict Knowledge Gain by investigating the features extracted from the visited pages that may lead to knowledge acquisition on the internet. He extracted text length, sentence length, and term length features, coverage of query terms in the title, distribution of the query terms, POS Distribution, page complexity, and page complexity vs. query complexity features. a machine learning model was built based on the mentioned features and it showed promising results by predicting precisely the gain in knowledge from the pages on the internet.

Tang et al. [60] focused on the extraction of the textual information from the web pages and entered queries by users, and introduced a classifier that aims to predict the KG of the users during a web search. Resource features and User behavior features were the categories based on the data extracted. Resource features included the complexity of the documents, such as the average word number per sentence and Gunning Fog Grade, HTML structure, and linguistic characteristics. On the other hand, User behavior features included session-related features (e.g. duration of the session), queries (e.g. the average length of queries), Search Engine Result Pages (SERP) (e.g. click ranks), browsing behavior features (e.g. visited pages ratio), and lastly, mouse movements (e.g. scroll distance amount). The classifier proved the ability to predict KG using the extracted set of features and outperformed the state-of-the-art baseline by 13.6

The influence of multimedia resource consumption on the knowledge gain of users was studied by Otto et al. [42]. He extracted numerous multimedia features on document layout by analyzing them and separating them into regions that represent page structure. As a result, six area classes were discriminated between (Heading, Menu bar, Content list, Text, Images/Frames, Background). In addition to that, image regions were examined regarding their content in order to define their classification type, resulting in several image types (infographics, indoor photo, naps, outdoor photo, technical drawings, and information visualization). Furthermore, the study included a set of 110 features that represent the textual information taking into consideration the complexity of the document, HTML structure, and linguistic aspects. The results of the analysis show that none of the multimedia features are listed at the bottom of the most crucial features, and emphasized the role multimedia resources (images/videos) play to improve KG prediction.

Moraes et al. [40] had a different approach to measuring learning gain. He attempted to measure Knowledge Gain by evaluating the change in the participants' vocabulary knowledge. Every participant was assigned a random specific condition for the learning task. Among these conditions is users can either watch only lecture videos about the topic, look in a single web search about the topic, or have both options to watch videos and use the web search page. The experiment results showed that the learning gain increased by 24 percent when watching only videos and up to 41 percent when participants used videos to learn alongside the search functionality.

2.3 Behavioral Data, Search Data and Knowledge Gain

Researchers did not stop exploring the influence of resource data on KG, but extended the area to include participants' behavioral and search data and study its impact on KG.

Various search session metrics were extracted for the studies. Ran et al.[65] extracted features related to the queries, SERP activity, Browsing behavior, mouse movements, and session. Query features contained its count, length (min, avg, total), complexity, and the number of total/unique terms. Additionally, SERP activity features and mouse movement features included a click count on SERP and logs of the clicking time between each click. Documents count, visited pages' count, SERP browsing time, and navigation source (pages navigated through/non-through SERP) were listed under browsing features. Last but not least, session duration was calculated in session features. The outcome of the study indicate that the time the user spent during the session is critical for Knowledge Gain prediction, and it affects it strongly. Moreover, features that are related to the content of the queries and visited documents are important indicators for a better KS and thereby a better KG.

Likewise, Gadiraju et al. [24] studied the Knowledge Gain analysis of the recruited participants after their search session on the internet to acquire information. The search sessions were recorded, and it included all interactions of participants and their various needs with their logs. User query and user clicks features were extracted. Other features such as search duration and browsing behavior features were extracted and arranged into 5 categories. the first category is the length of the session, second is the count of web pages the participant was forwarded to. Third, Domain and search engine result pages, fourth is the pay-level domains across topics (PLD), the fifth is the formulation of the query (e.g. query complexity, topic description overlapping, and finally, the evaluation of queries within search sessions. The results showed participants with higher Knowledge Gain spent more active time on pages, and it was also found that there is a correlation between the average complexity of queries and their knowledge gain.

Similarly, Yu et al. [64] investigated ways to detect the learning process of acquiring information when it occurs on the web and attempted to understand it to support the individuals' informational goal. In addition to that, predicting the outcome of the informational session by building a supervised classifier that takes 22 features and categorized them into three types of features, which are query, session, and browsing features. Query features include the count of query terms and the similarity between entered queries, on the other hand, issued query count, duration of the session, and session break features were involved in the session category of structured features. Finally, the features extracted out of the browsing behavior category were the count of clicks, the similarity between clicked URL and query, and revisited pages. In terms of Knowledge Gain, Yu's work states that Browsing features are among the most important categorized features. In the contrast, Session features come last. Query features played also an effective role in predicting the KG.

Bhattacharya and Gwizdka [2] studied the differences in users' Search Interaction (SI) and their impact on KG. The behavior was captured by collecting many SI metrics such as the number of visited Search Engine Result Pages (SERPs) and CONTENT pages. Dwell time was also included among the metrics, alongside the type of query reformulations, and commonality of used words in the queries. Results showed that the participants with low KG typed fewer new queries than those with high KG. However, there was no difference in the entered queries count, nor in the count of relevant pages that were found during the search.

Kalyani and Gadiraju [36] explored predicting knowledge gain by testing the participant's cognitive abilities level during web search sessions based on their search behavior. The task of the study consisted of test questions that would test his cognitive level by asking questions that would require the participants to remember, understand, analyze, evaluate, and create. Results of the study show that better cognitive abilities greatly affect the search behavior of the participant and the knowledge gain.

With the focus on search behavior only, Ghosh et al. [25] explored the relationship between information seeking, searching, and learning and the outcome of the process. They extracted multiple features to trach the search behavior of the participant of the experiment, such as the number of visited pages, the number of saved pages, query count, length, diversity, and reformulations. Last but not least, they extracted the average dwell time for every task. The reported findings show patterns for high-knowledge learners in their search, such as differences in query length.

In a different approach, Zhang and Liu [67] attempted to measure Knowledge Gain by looking into the change in the usage of participants' vocabulary during their search activities. They used multiple variables to model their strategy. Amongst these variables are the prior knowledge vocabulary counts (Active PKV), the ratio between prior knowledge vocabulary used in search activities and the total vocabulary of the prior knowledge (Active PKV ratio), prior knowledge vocabulary ratio used in queries to total query terms (PKV dependency), and last but not least, count of queries' used vocabulary of prior knowledge that is on the i-level in the pre-search mind map (Active PVK from level i). As anticipated, the findings ensure the importance of prior knowledge vocabulary in shaping the queries and looking for information in the first stage of the search. However, the reliance on prior knowledge vocabulary reduces as the participants continue their query search due to attaining new knowledge.

In this work, we attempt to study the participants' behavior by calculating the time the participants spend while reading and fixating on the words. On the other side, we want to calculate the time they spent looking at regions unrelated to the textual contents. Therefore, exploring if these behavioral characteristics can make an impact on Knowledge Gain and the model ability to predict it.

2.4 Eye-tracking Data, Reading Pattern, and Knowledge Gain

The research area in reading patterns and knowledge gain has not been studied extensively. However, prior work [2, 10] shows the correlation between eye movement and knowledge.

J. Cole et al. [10], in their research, looked into eye movement patterns and the level of user's knowledge by distinguishing the types of Reading patterns the participants follow when performing an informational task that requires knowledge acquisition. He found that there is only one way to acquire the sought information by gazing at it repeatedly. Additionally, the person's attention is needed to perform and complete the acquisition. Eye-tracking data enabled them to discover patterns in observing documents as well, such as the "F-shape" reading pattern in which the user looks at the information in an F-shape way to read search engine result page (SERP), and the E-Z Reader model, which describes the movement pattern of the eye to identify the word, do the visual processing, the attention and control the oculomotor system during the reading function. The results show a connection between eye movements and the user's domain knowledge in deriving the cognitive nature of information interaction.

Syed et al. [59] explored the signifying indications of low and high-learning participants by examining their reading and gazing patterns and trying to uncover a connection with processing information to support their learning and evaluate knowledge. They computed participants' total reading time, total fixation count/duration of regression, reading, and scanning. In addition to that, they investigated the relationship between the time spent on reading and post-test grades from one side and another side between the result of learning and reading fixation behavior. Their study showed that the participants with low learning knowledge had higher numbers of fixations from scanning and regression types, indicating that they did not pay much attention to reading the text.

On the other hand, Bhattacharya and Gwizdka [2] researched the effect of eye-tracking data on predicting Knowledge Gain by analyzing the eye tracker data of the participants during their web search on the internet. This process was intended to look into the reading pattern that the participants follow to acquire knowledge. Several features were extracted to measure the influence, such as total reading duration and count of active fixations. Additionally, the total length of the reading sequence, number of eye regressions, and the length of regressions were calculated on both SERPs and CONTENT pages that are relevant to the task. The results of the paper showed that the group of participants who had spent more reading fixations time and fixation duration per CONTENT page belonged to the LO group, which represents participants with low Knowledge Gain. Similarly, the LO group committed more eye regressions in reading. Therefore, longer overall reading sequences than the participants in HI group with high KG.

Bhattacharya and Gwizdka's paper is the foundation of our work for multiple reasons: (A) They studied the effect and influence of eye-tracking features on KG. (B) They considered behavioral features on the KG prediction. (C) They built a machine learning model to predict KG. Furthermore, we extend the features area by looking extensively at Eye-tracking features and Reading pattern features to measure and predict KG.

This work expands the work prior researchers did in two manners. First, as shown in this section most studies focused on textual, multimedia, search interactions, and behavioral features. However, eye-tracking and reading patterns features weren't explored appropriately. Therefore, we want to measure the outcome of their influence on KG. Second, several studies introduced machine learning models to predict KG based on resource (text, multimedia) and behavioral features. In this work. We want to build and train a supervised model that predicts Knowledge Gain based on eye-tracking features and the reading patterns of the users.

3 Foundations

3.1 Eye Movements during Reading

To understand the various reading models, we need to briefly understand the terms that describe the movement of the eyes.

3.1.1 Fixations and Saccades

During the reading process, the reader doesn't move their eyes simply from left to right in a straight line across the page. Instead, however, their eyes move back and forth. Fixations are the stops during reading, in which the eye stop on a word to identify. Many factors define the duration of the fixation but it can be less than 100 ms and more than 400 ms depending on the complexity of text [51]. On the other hand, saccades are the jerky movement of the eyes to shift the attention to the next word to identify. Saccades can skip, repeat or fixate on words as well.

3.1.2 Visual Acuity

It is the amount of visual information the eye can process during reading. It is limited, which is why the eyes continuously move during reading because we need to identify the words by fixating on them. The visual acuity is minimal in the periphery and maximal in the retina center [51].

3.1.3 Saccade Latency

The eyes read about 300 words per minute approximately as Reichle et al. [51] stated. For the eyes to achieve reading that amount of words, there is a race between the process of recognizing the words and planning for the next saccade, and this is called saccade latency [51].

3.1.4 Perceptual Span

Ryner [50] defines it as, in a given fixation, how much visual information is processed. In other words, how much visual information can be encoded in one area, and usually, the readers proceed at a normal rate when the window contains between 14-15 character spaces to the right and 3-4 characters to the left. Language can be a factor for the area of the span to increase or decrease, such as in Hebrew, the perceptual span area decreases as we can see in figure 3.1. Moreover, the difficulty of a word can also cause the perceptual span to decrease. On the other hand, when the coming word can be predictable, the perceptual span increases, and obtain more information to the right.

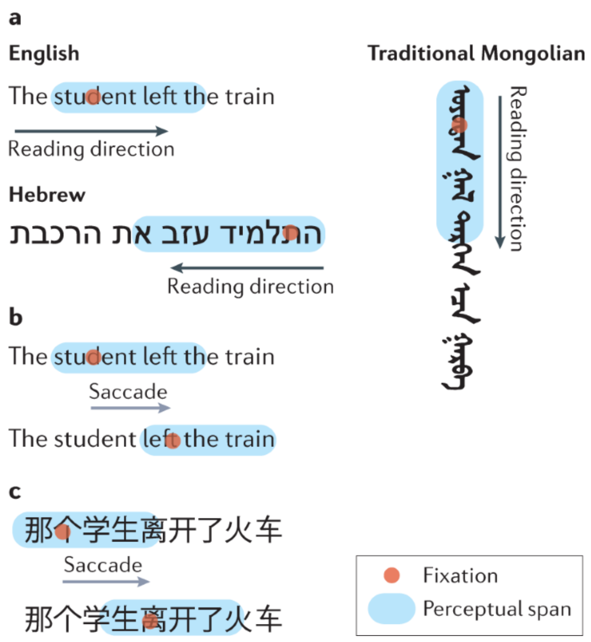


Figure 3.1: Visualization of the perceptual span and its change based on different languages as Li et al. depicted [38]

3.1.5 Regressions

Regression is the eyes' movements backward when reading, such as to the left when reading English or German content. It can be a measure of either the complexity of reading a word which is a linguistic processing issue, or oculomotor error [51]. The regression of the eyes can be seen in the figure 3.2 in the steps 7 and 8.

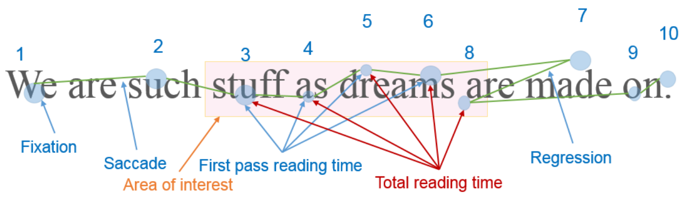


Figure 3.2: Visualization of the regression movement in the eyes in step 7 and 8 as Eckstein et al. [16] displayed

3.1.6 Where to Fixate Next

Low-level visual cues in the text can determine where the next fixation will be. The gap between the words or the length of the words. Studies [51] defined several types of conditions to determine where to fixate next. First, the length of the fixated word can influence the saccade length. Second, Saccade length drops when reading does not have information about where gaps are between coming words. Third, the first fixation is usually made between the first word and the middle between the start and middle of the word.

3.2 E-Z Reading Model

E-Z Reader Model is an eye movement method and a process for human reading that explains the chain sequence of reading steps, in which the reader looks at a word (fixation) at a time and transitions to the next word (saccade). This method describes the relationship between the low-level side of language processing and vision. It includes four crucial steps that tell where and when the eyes should jump while reading, and they are word identification (also called familiarity check), visual processing, attention, and oculomotor control [51].

3.2.1 Word Identification

It is also called a familiarity check and consists of two phases. The identification starts once the attention is allocated to a word to read. The first phase is word form identification. In this stage, only the lexical aspect is essential. The other aspects, such as the semantic or the phonological form of the words, are not. In the second phase, the semantic and phonological forms of the word are identified to enable linguistic processing. The mean time for word identification can vary depending on the word and the language itself, and some languages take more time and some less. Moreover, it depends on the predictability of the word as well [51].

3.2.2 Visual Processing

The visual features of the displayed text on the screen are transmitted to the visual cortex through the retina so words can be recognized. The transmission takes time to be processed, up to 90 milliseconds. Visual processing helps to acquire the frame of the word, which helps to plan saccades to the following words during reading. Furthermore, it helps allocate attention to words in the higher-level visual regions to recognize them [51].

3.2.3 Attention

One of the most crucial steps of the E-Z Reader model. Attention allocation allows the word to be identified. Therefore, proceeding to the next word. The process happens serially because the eye moves from one word to the other in a straight order. Attention allocation moves to the next word after finishing the second phase of word identification [51].

3.2.4 Oculomotor Control

The coordination of eye movements is one of the tasks of oculomotor control. That includes the eyes' quick and steady movement and fixations on targets such as words for reading [32].

3.3 Reading Protocol Tool

The Reading Protocol Tool [29] is an eye movement tracking Software that can map the data points of gaze information down to the word level. Therefore, giving more flexibility to interpret the eye gaze data, insights, and many possibilities to analyze those interpretations. Unlike other eye-tracking software that relies on generating a plot or heat-map of the gaze points, which makes the process of analyzing those data costly because of the need for a manual observation to analyze the image or video-based data. The Reading Protocol Software is capable of mapping the subject's eye gaze points to their corresponding stimuli pages down to the word level. Moreover, It gives the power to manipulate the data quickly and filter them instantly based on some filtering options such as filtering by a specific subject, stimulus, and areas of interest. The Reading Protocol Software also allows the export of fixations time for the entire eye data to continue processing using other tools to get more information about the users and to understand their behavior from their data information.

We summarize Hienert et al. [29] work in the following two subsections which describe the software inputs and the computation methods.

3.3.1 Data input

For the software to start processing and mapping the words, it must be fed with two dataset types of files. First, the software needs a file of the raw gaze data of users from the eye-tracking software. The file can be exported as a CSV file. Second, it needs the HTML content of the visited stimulus. The website HTML content can be acquired in various ways, either by plugins that record the website in new experiments or can be used from another experiment if recorded or from existing images of the stimuli by using Optical Character Recognition (OCR). After providing the software with the needed data, it starts mapping the eye-gaze data points down to the text-level of the words of every stimulus page the users looked at while searching. As a result, a JSON structure is built that contains the mapped information. Every word in the JSON structure contains the word position on the x and y axis on the page. In addition, the word's width and height alongside the fixation time and its timestamp for the first and last time the user looked at the word.

3.3.2 Word-Eye-Fixation Computation

The algorithm of the Reading Protocol software looks for the original website layout and opens it up in a browser. After that, the coordinates of the eye gaze data points are fetched from the database. Then the algorithm uses the Dispersion-Threshold Identification (I-DT) [53] algorithm to divide the data points between fixations and saccades. A particular method is called in the browser to find the word on which the eye data point is located, and this process is done for every fixation. The algorithm returns timestamps, text position, word position, fixation times, and stimulus if a word is found. The returned word is saved alongside all the other found words and stored in the database.

3.4 Machine Learning Classifiers

Machine Learning is becoming more involved in nearly every scientific domain. It facilitates research because it enables us to unlock and observe patterns that are not easy for humans to discover without its help. To understand how machine learning algorithms work, we need to look into the diverse classifiers to comprehend how they perform the calculations. We relied on multiple classifiers in this work, which led to our results. Amongst these classifiers are Adaptive Boosting (Adaboost), K-Nearest Neighbour (K-NN), Decision Tree (DT), Random Forest (RF), Multi-layer Perceptron (MLP), and last but not least, Support Vector Machines (SVM). In this section, we will look closely at these classifiers, their definition, and the way their algorithms do the computations.

3.4.1 AdaBoost

AdaBoost, an abbreviation for Adaptive Boosting, was proposed by [22] Yoav Freund and Robert Schapire in 1996 and is one of the iterative Ensemble boosting classifiers. In AdaBoost, multiple classifiers are combined in order to increase the accuracy of the classifier, and in each iteration, the weights of the trained sample of data and classifiers are set [21]. Thereby, the prediction accuracy is guaranteed in case of uncommon observations. In order for AdaBoost to work correctly, some conditions have to be met, such that the classifier has to be on multiple weighed training instances trained in an interactive way. Moreover, it attempts to look for a perfect fit by making training error reduction. [15]

3.4.1.1 Terminology

- Bagging: reducing the variance of estimates by fusing multiple models.
- Boosting: fusing multiple low-performing classifiers to build one with better accuracy
- Stacking: also called blending, a combination of many base prediction models into a new data-set the output of the fused base prediction models is treated as an input for another classifier.

3.4.1.2 Ensemble Machine Learning Approach

It is the process of harnessing multiple low-performing machine learning algorithms in order to fuse them and produce one with a better performance [15]. Therefore, it delivers better accuracy than the base or individual classifiers. Bagging is used in the Ensemble approach to reduce the variance. Boosting is also used to decrease the bias, and Stacking to increase the prediction.

3.4.1.3 How does AdaBoost Algorithm Work?

In the beginning, AdaBoost Algorithm randomly picks a subset from the training data. After that, the model gets trained iteratively, and in every iteration [28], the training set with the highest prediction accuracy gets chosen for the next iteration. In the next step, The classifications with low observations will be set with a higher weight. Thereby, in the following iteration, these observations will get an increased probability for observations. Additionally, the weight of the trained classifier is set according to the accuracy of the classifier, and the classifier with the highest accuracy will be considered by voting as depicted in the figure 3.3. The iteration cycles will keep happening till the max number of estimators is reached, or all training data is fit without errors.

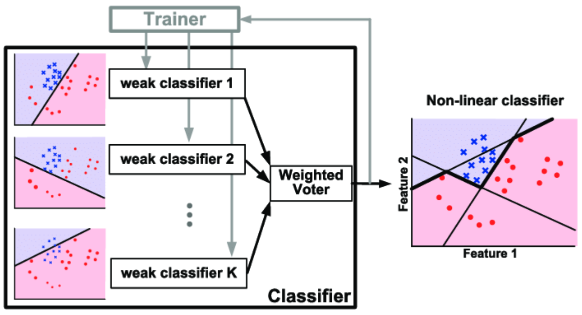


Figure 3.3: Wang et al. [62] steps visualization of Adaboost algorithm

3.4.2 K-Nearest Neighbour

K-Nearest Neighbor, also known as K-NN, created by Joseph Hodges and Evelyn Fix in 1951 [45], is a supervised non-parametric learning classifier. It uses data points grouping to predict the outcome. K-NN's algorithm is also applicable to Classification and Regression tasks but is mainly used for solving Classification problems.

3.4.2.1 How does K-NN Work?

In order to make the class prediction for the test data, K-Nearest Neighbor computes the distance between the available training points and the test data. After that, The K value of points is chosen to make the model consider the exact amount of points most proximate to the test data. Finally, It selects the class with the highest probability by computing the probability of the test data, which belongs to the classes of K training data. On the other hand, in the case of a Regression task, the mean of the chosen K training points is the value. Nevertheless, there are multiple methods of computing the distance between the data points depending on the type of the problem. Some of these methods [9] are the Straight line distance, also called Euclidean distance:.

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}.$$

In addition to that, there are Manhattan Distance (also called City Block Distance):

$$d_{st} = (\sum_{j=1}^{n} |x_{sj} - y_{tj}|) \tag{3.1}$$

and Hamming Distance as well:

$$d_{st} = (\frac{\#(x_{sj} \neq y_{tj})}{n}) \tag{3.2}$$

3.4.2.2 How to Choose the K Value

In order to get the best K value for our data, we need to execute the K-NN algorithm several times, but with different values for K. The K value with the lowest error count and good predictability ratio for the test data that have not been seen will be used. Moreover, we must be careful about choosing the K value because the lower the value, the less stable the algorithm becomes. In contrast, the higher the K value, the more stable, but up to a certain limit because the number of errors will start to increase once the limit is passed. Figure 3.4 shows how the K-value affect the sample as depicted by [66]. The stability of the algorithm is due to the Averaging or Majority voting that gives the algorithm the ability to predict accurately. In the case of majority voting, it is best to choose K as an odd number to eliminate a tie if it occurs.

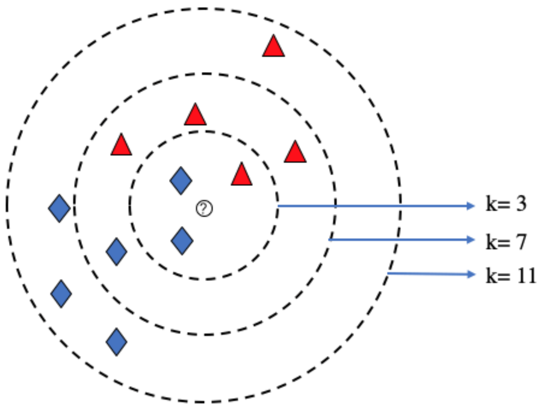


Figure 3.4: Figure shows the change in the K-value leads to a different classification for the sample [66].

3.4.3 Decision Tree

Decision Tree classifier is also one of the supervised learning algorithms [61]. It makes decisions based on a set of rules that mimic how humans make decisions, and it is a tree-like structure that models the potential outcomes and displays it as a path of decisions with conditional statements [8]. Branches get created along the way during the decision-making steps, which lead to the final result.

3.4.3.1 Types of Decision Trees

There are two main kinds of Decision Trees depending on the target. First, Categorical Variable Decision Tree (Target divided into categories, for instance, the categories can be either yes or no), and Second, Continuous Variable Decision Tree (Continuous target with unknown variable limit, such as score points) [14].

3.4.3.2 Terminology in Decision Trees

There are a group of concepts related to Decision Trees to understand how they function.

- Splitting means dividing the node into more or multiple sub-nodes.
- Pruning: The process of removing sub-nodes from the decision tree. It is the opposite of Splitting.
- Root Node: It is the first root in the tree and represents the whole sample. In later steps, it gets divided into two or more sets.

- Decision Node: The sub-nodes are called decision nodes when they are split into more sub-nodes.
- Terminal Nodes: They are the nodes that can not be further split into sub-nodes. They are also called Leaf.
- Branch: It is a tree sub-section. It is also called a sub-tree.
- Parent Node: The Node, which is split into two sub-nodes, is called Parent Node concerning its Child Nodes.
- Child Nodes are the sub-nodes of the split Parent Node.

3.4.3.3 How Do Decision Trees Work?

The idea behind the Decision Tree algorithm is to continually split the data-set features based on YES/NO questions until all data points are separated and belong to a specific class [35]. With each question asked, we create a new node from the parent node, and the data-set is split based on the question's answer depending on the feature's value, resulting in new sub-nodes. The first node in the structure is called the root node, and the last nodes in this process are called leaf nodes or child nodes. As a result, the outcome of this procedure is a tree-based structure. As long splitting keeps occurring, new branches get created to separate feature regions. One region would hold all yes-answered data points to questions while the other would hold the remaining data points. The feature space gets narrowed down with every split making the data points belong to one region only. The purpose of this process is to keep splitting and applying the set of rules till we do not have more data points to use or no more rules to apply. Once the splitting ends, a new process starts, in which all data points in leaf nodes get assigned to a single class, and these nodes are called pure leaf nodes [33]. Sometimes we get mixed leaf nodes because not all data points get assigned with the same class. Therefore, the algorithm assigns the most common class among the node's data points.

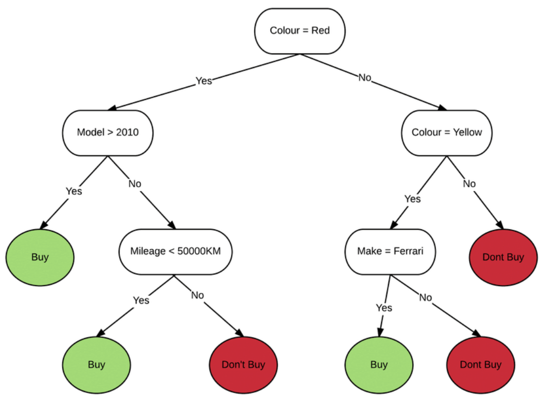


Figure 3.5: Figure shows the decisions path of DT [63].

3.4.3.4 Attribute Selection Measures

Picking the root attribute might not be a trivial task because choosing a random attribute as the root node for the splitting might lead to bad results and low accuracy. Charbuty et al. [8] explained multiple criteria to select the attribute to solve this complicated task by calculating the values of each attribute, such as:

1. Entropy: It is a randomness measure, and the higher the entropy is, the more difficult it is to make any decisions from the information. The maximum value of entropy that indicates perfect randomness in the data is 0.5.
2. Information Gain (IG): It is a measure to decide how efficiently the data-set was split based on a specific attribute to return the best Information Gain and least Entropy. It is calculated by subtracting the entropy before the split from the average entropy after the split, depending on a given attribute.
3. Gini Index: It evaluates splits based on the cost of function and is computed by subtracting the squared probabilities of each class's sum from one. Gini Index serves binary splits.
4. Gain Ratio: It favors an attribute over another for the splits due to its extensive number of different values.

3.4.4 Random Forest

Random Forest classifier, proposed by L. Breiman in 2001, is a supervised machine learning algorithm that can efficiently solve Classification and Regression problems [3].

3.4.4.1 How does RF Work?

Biau et al. [3] describe how the classifier works as randomized splits of multiple built decision trees with averaged predictions to deliver better performance.

Sruthi [57] adds that the algorithm of Random Forest consists of two main steps for the classification: Data bagging and Feature Randomness when creating every tree to avoid correlation between them.

Data bagging: also called bootstrap aggregation, is an Ensemble technique that combines multiple models and therefore utilizes the models to train the data. In this technique, different base models are built and provided with a random sample dataset. This step is called Row Sampling with Replacement because the row of data chosen for a model is put back into the data. When the training ends, the generated prediction outputs get based on either majority voting or averaging, and all the outcomes of the models get aggregated. This step guarantees that we are not using the exact data for every tree.

On the other hand, Feature Randomness picks a random subset of features for each tree and uses them exclusively for training. This step helps to reduce the correlation between the trees and avoids having an identical decision node for every tree. Therefore, they will act similarly. Moreover, it is a good indicator of the importance of features, especially when some trees get trained on less essential features. As a result, the prediction score will be low.

After having the data and feature subset, the trees are ready to be built. As a result, all trees look different from each other.

Random Forest Hyper-parameters The classifier has the ability to improve its performance even further by using its hyper-parameters. Some of these hyper-parameters increase the computational speed of the classifier, and some increase the predictive ability. The following hyper-parameters n-estimators (trees number that is built before averaging the prediction), mini-sample-leaf (the leaves' minimum number that is needed to split a node), and max-features (the maximum number of features that are regarded for the split) can be adjusted to make the model predict better. On the other hand, adjusting n-jobs (number of processors that are allowed to be used), oob-score (out of the bag, one-third of the sample is not used to train the data, and this method is called the cross-validation method, and it is used to evaluate its performance), and random-state (it defines the randomness of the sample)

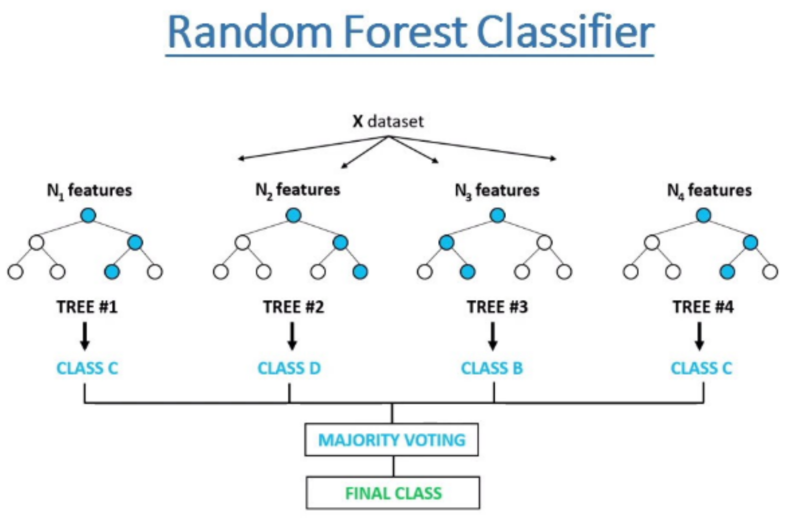


Figure 3.6: Figure shows that RF classifier is made of multiple decision tree [49]

3.4.5 A Multilayer Perceptron

A Multilayer Perceptron (MLP) is one of the deep learning algorithms proposed by Frank Rosenblatt in 1950 [48]. It is made of an input layer, an output layer, and one or more hidden layers in between. The input layer takes the information to process, and the output layer throws the result according to the information entered in the input layer. The hidden layers consist of neurons that are piled together.

3.4.5.1 Terms in MLP

Several terms need to be understood in MLP, and they are described below by Murtagh [41].

- Backpropagation: updating weights in multilayer by a learning algorithm
- Epoch: also called a training cycle, and it is training the MLP by the whole data set, in other words, when each training instance has been seen once by the neural network.
- FeedForward: updating weights from lower to higher layers, from the input layer to the output.
- Hidden units: The layers between the input and output layers.
- Multilayer Perceptron: a fully connected neural network trained by a backpropagation algorithm.

3.4.5.2 How MLP Works?

The inputs in Multilayer Perceptron have initial weights in a weighted sum, which is why the algorithm is considered a Feedforward algorithm and is subjected to an activation function, which means how much weight to give to the incoming input and has a threshold that must be reached to have a specific effect. Every layer's input is the output from the previous layer's computations, and the data flow from the input layers, through the hidden layers, and finally to the output layer. It is described as feeding the successive layers with information. This whole process is categorized among the Feedforward algorithms [47], the structure can be seen in figure 3.7, and it is an iterative process that computes multiple times to reduce the cost function by learning the weights through Backpropagation. There would be no learning if this process were only executed once. The mean squared error is calculated in all output and input pairs in every iteration when the weighted sums are forwarded through all layers. After that, the first hidden layer's weight is updated with the new value of the gradients, which is how it is propagated back.

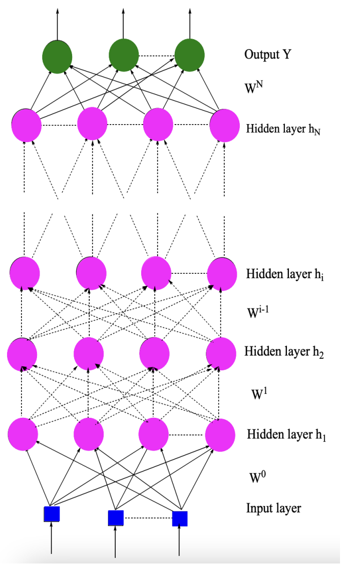


Figure 3.7: The figure shows FeedForward structure [47]

In contrast, as a disadvantage, adding more layers above the limits for a task will make our convergence slow due to the high variance.

3.4.6 Support-vector Machine

SVM (Support Vector Machine) is an ML algorithm proposed by Boser, Guyon, and Vapnik in 1992. It consists of a set of supervised machine learning algorithms, and it is applied for both Regression and Classification tasks. The algorithms type is from generalized linear classifiers and prevents overfitting while increasing its prediction accuracy. SVM can also be used in different domains, such as face and handwriting analysis and pattern classification [34].

3.4.6.1 How does SVM Algorithm Work?

The algorithm can be applied for many kinds of classification, such as binary and non-binary classification.

- Binary classification: The algorithm tries to draw a hyperplane, a line that separates the data points plotted in a 2-dimensional space [37]. The hyperplane is a decision edge itself, meaning any data that fall on one side of the hyperplane will be classified based on the class of the side. Finally, the algorithm chooses the best hyperplane, which is the one that increases the margin of data points on both sides of the divided plot. The hyperplane can be seen in the figure 3.7.
- Multi-class classification: In real-world applications, most data sets can not be separated by a linear hyperplane. Therefore we need a method that can divide classes in

a non-linear way. The Kernel Trick method in SVM allows us to separate the data sets without worrying about the number of dimensions by mapping the data input to a high-dimensional space, resulting in linear dividable mapping [34].

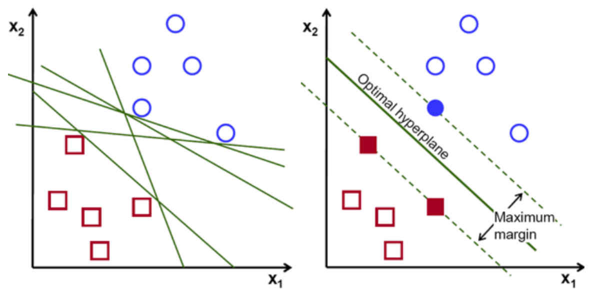


Figure 3.8: Support-vector Machine binary classification [58]

3.5 Correlation Coefficient

The correlation Coefficient is a measure that finds the statistical relationship strength between two continuous variables. The relationship is linear and falls in the interval between -1 and 1, meaning two variables can have a positive or negative correlation. Correlation Coefficients rely on multiple approaches to estimate the statistical connection, such as Rank Correlation, Re-sampling, and Pearson Correlation Coefficient [46]. In our work, we apply Pearson Correlation Coefficient to measure the relationship between our eye-movements features and pre, post-test, and knowledge gain scores. Therefore, we will focus on it in this section.

3.5.1 Pearson Correlation Coefficient

Pearson Correlation Coefficient (PCC), also called Pearson Product-moment Correlation Coefficient, belongs to the linear relationship measurement strategies to find the connection strength between two continuous variables [46] by applying the formula 3.3.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.3}$$

Where:

- x_i: x-variable values in a sample
- y_i: y-variable values in a sample
- $\bar{x}$: sample mean of x_i
- $\bar{y}$: sample mean of y_i

PCC Assumptions: There are several assumptions that have to be followed in order to get a proper indicator of the strength of the relationship between the variables as defined by Schober et al. [56]. Among these assumptions are the following:

- There has to be an assumption that the variables have a linear relationship and form a straight line on a scatter diagram.
- Variables have to be continuous. Ordinal variables have to be calculated in a different correlation method.
- Outliers in the variables are to be avoided because they will affect the correlation results by dragging the correlation line in one's direction.
- Variable pairs are to be measured individually from the other pairs.

3.5.2 Correlation Coefficient Interpretation

There are multiple levels of connections between the variables. Therefore different degrees to classify, such as Perfect, strong, moderate, weak, and no correlation. If the class is perfect (close to $\pm$ 1), it means the increase of one variable will increase the other variable as well, and in both directions, negative and positive. a Strong degree means the value is between ± 1 and ± 0.50. A moderate degree is between ± 0.30 and ± 0.49, whereas a weak degree falls between ± 0.29 and below. Finally, when the value is zero, then it is classified as no correlation [56].

4 Methodology

In this chapter, we will look into the implementation part of Bhattacharya and Gwizdka's [2] replication and our extended word-level implementation in detail. We will start by describing the pre-processing steps for both implementations in section (4.1), as depicted in the figures (4.1 and 4.2). The pre-processing section consists of cleaning data (4.1.1) and grouping fixations (4.1.2). After the pre-processing phase, we will present features' extraction methods in section (4.2). In section (4.2.1), we will talk about extracting the eye movement and reading pattern features. Finally, in section (4.2.2), we will talk about the extraction of the behavioral features of the participants.

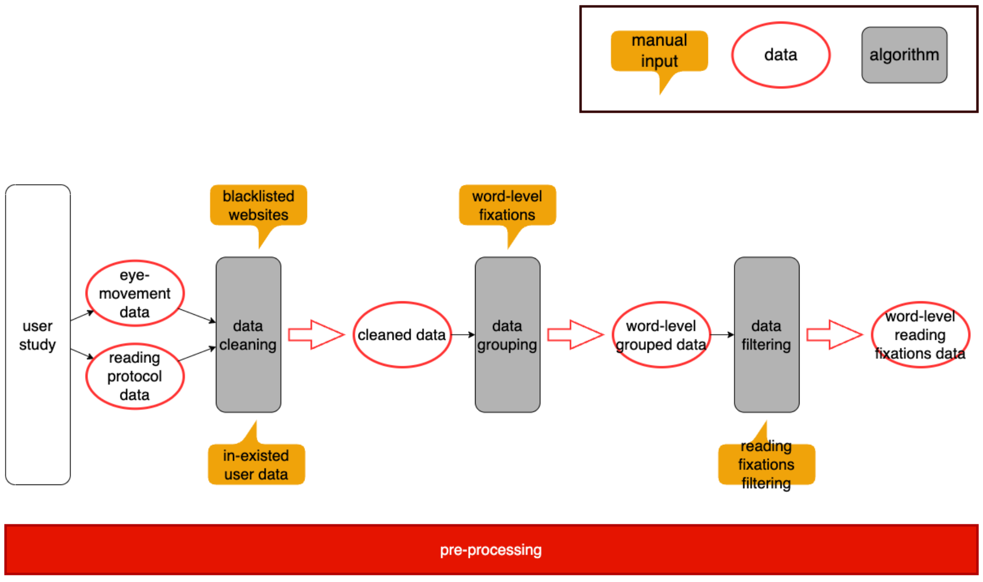


Figure 4.1: Pre-processing steps that are followed in our extended word-level work

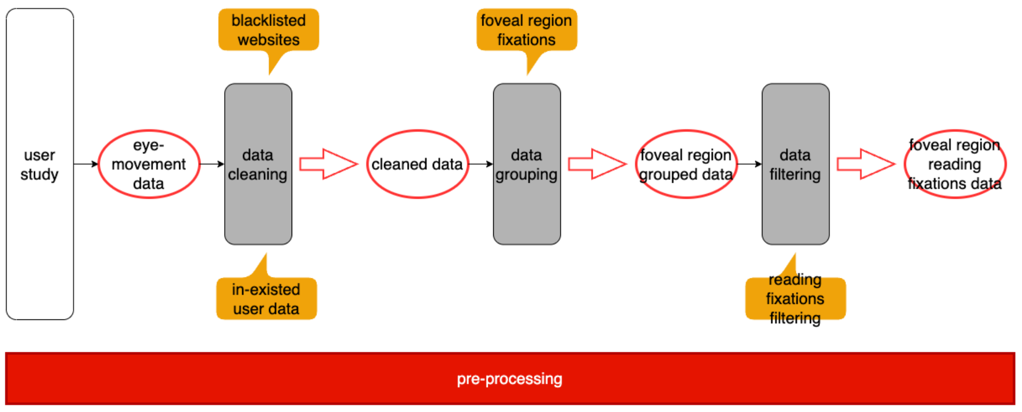


Figure 4.2: Pre-processing steps that are used to replicate the implementation of Bhattacharya and Gwizdka's work

4.1 Bhattacharya and Gwizdka and Words Level data pre-processing

Our main objective for the implementation part of the work is to predict Knowledge Gain based on reading patterns and eye movement on text. Therefore, we had to clean our data-set first and then filter it further, to achieve the goal.

4.1.1 Data Cleaning

During the web search task in the laboratory, the participants were encouraged and asked to look freely on the internet for learning materials about the topic to acquire knowledge. The search resulted in a vast collection of websites containing text and multimedia resources, including images and videos. In our work, we want to concentrate on reading patterns in text. Therefore, we excluded all websites that only contain multimedia resources, such as YouTube, because the eyes move in a different pattern when looking at a multimedia resource, unlike in reading. We also excluded all search engine pages, such as Google and Ecosia. Prior studies found that the movement of the eyes takes a different looking shape in search engine results, such as an F-shaped pattern [26]. This pattern aims to find the best result most efficiently. However, it follows a different eye-movement style, in which the eyes move more horizontally, and the person is not interested to read every word.

4.1.2 Filtering and Grouping

In this section, we will discuss two pre-processing methods we used in our work. First, to re-implement Bhattacharya and Gwizdka's work [2], and second, to extend their work even further to calculate the features down to the word level with the help of the Reading Protocol software data [29].

4.1.2.1 Bhattacharya and Gwizdka Grouping and Filtering

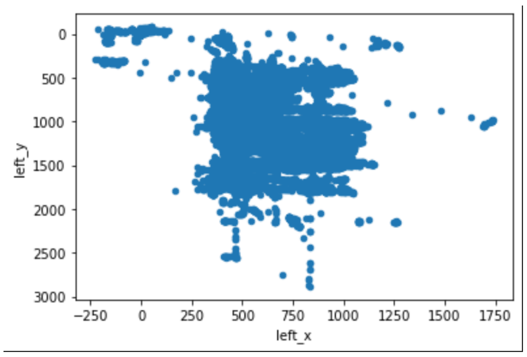
In Bhattacharya and Gwzdka's work [2], they relied on eye-gaze data points, meaning they only had access to the data regarding the coordinates of the eyes on the screen and the timestamp of the eye data points. So to translate that to the data set we have, we used the locations of the eye fixations (left eye and right eye) on the screen on both the x and y-axis, the active website the participant was surfing, and the timestamp. The eye tracking software used in the experiment registered a new fixation every seven milliseconds. Therefore, it created the challenge to identify the same eye fixations. To tackle this challenge, we introduced a method that identifies the same eye fixations and groups and fuses them as they should be. The grouping happens based on the central vision, which is the size of the foveal region radius. J. Cole et al. define it [11] as 35 pixels. Next, We check the distance between the current fixation (the mean value of the left eye and the right eye on the x and y axis) and the mean value of the previous fixations. If the distance between the two coordinates is less or equal to the size of the foveal region radius, then the new fixation belongs to the previous ones. If not, we group the previous fixations and start identifying the next ones. To calculate the distance, we used the Distance Formula that is defined by

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}. \tag{4.1}$$

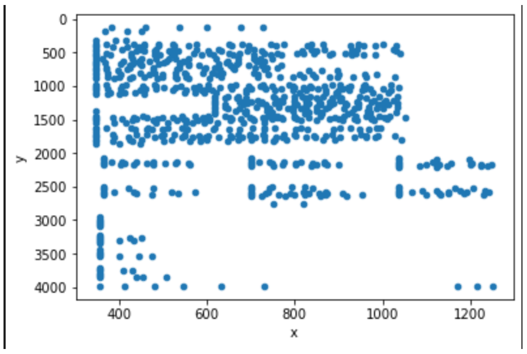
The formula calculates the distance between two given points, taking into account both the x-axis and y-axis. This fits perfectly with the nature of the eye movement on both axes on the screen. Therefore accurate distance results. We are only interested in the reading fixations. Therefore, after fusing the eye fixations, we check if the fused fixation is from a reading type. To know if fixation is from a reading type, we must check its duration by iterating over all grouped fixations and calculating their duration. Based on J. Cole et al. [10], the minimum reading fixation duration is 113 milliseconds. Accordingly, we filtered the fixations that do not add to the required duration. As a result of this phase, we ended up having a list of reading fixations ready to be further processed based on Bhattacharya and Gwizdka's implementation.

4.1.2.2 Word-level Grouping and Filtering

Our data-set contains ungrouped and unfiltered fixations. Therefore, we need to start the pre-processing phase by grouping the fixations first. Grouping eye fixations take another path here due to the text and word-level JSON structure data received from the Reading Protocol Software after providing it with our data to help us precisely to map the eye data



(a) Visualization of the eye fixations on a web page



(b) Visualization of the words coordinates of the same web page by using the reading protocol

Figure 4.3: Eye gaze data for participant with id equals to one (left) and the viewed website during the session ("https://www.weltderphysik.de/thema/hinter-den-dingen/gewitterblitze"). On the other hand, The data retrieved from the reading protocol for the same participant and same website (right)

points. The JSON structure contains text-level information of the words the user looked at during the web search task, we plotted the word level information to see to see the details of information about the website we get when we filter using it as we can see in figure 4.3. The information contains the word's position on the screen on the x and y-axis, the width, the height, and the stimulus. In our work, There is no need to use the foveal region radius to group fixations because we can use this information from the Reading Protocol Software to check if the eye fixation is inside the word-box bounds. If yes, we check if the next fixation falls inside the previous box. If not, then we have another fixation for another word. After grouping, we need to filter fixations that do not belong to the reading type. as mentioned in Bhattacharya and Gwizdka's part, the filtering happens based on the minimum fixation duration of the grouped fixations. If the duration is below 113 milliseconds, then we ignore the fixations. On the other side, using Reading Protocol data creates a new challenge because, at the end of the pre-processing phase, we will not be able to determine if these grouped and filtered fixations are considered consecutive reading fixations. To tackle this challenge, we marked the previous reading fixation as the 'last in reading sequence' when the current fixation is out of text bounds. As a result of this phase, we ended up having a list of marked reading fixations ready to be further processed based on our implementation that extends Bhattacharya and Gwizdka's work.

4.2 Feature Extraction

At the end of the pre-processing phase, We are ready to calculate and extract our planned features. We calculate a list of features that we think are necessary for predicting the

Knowledge Gain based on the reading pattern and the movement of the eyes. Some features are derived from Bhattacharya and Gwizdka's paper [2], and the others were chosen because they describe the participant's behavior.

4.2.1 Reading Pattern and Eye-tracking Data Processing

The first list of features was designed to explore how the participants read the text and uncover patterns that are more efficient for learning based on eye movement. These features are fixations count, fixations duration, reading length, regression count, and regression length. We relied on Bhattacharya and Gwizdka's paper [2] to extract these features. However, as mentioned in the pre-processing section, we have two implementations for them, one that replicates the implementation of their paper, and the other is our own extended approach that goes down to the word level of the pages.

4.2.1.1 Total Reading Fixations Count Calculation

A big part of calculating the total reading fixation count is done during the pre-processing phase because we get all reading fixations after cleaning, and grouping them at the end. This applies to both Bhattacharya and Gwizdka's replication of their implementation and our extended work based on the word level calculation. We extract the total reading count by returning the fixations count from the pre-processing phase's output. It is essential to mention that the result of the total reading fixation count from Bhattacharya and Gwizdka's implementation is different from our extended work. That is because we separate the fixations based on the bounds of the word itself that is looked at, unlinke in Bhattacharya and Gwizdka, where they use the foveal region radius defined by 35 pixels [11]. This means our extended work returns a more accurate result.

4.2.1.2 Fixation Duration Calculation

Similarly to the Total Reading fixation count calculation, we get the reading fixations from the output of the pre-processing phase. However, we cannot extract the total reading fixations duration just yet, because the output contains grouped fixations. Therefore, we still need to fuse every list of the same fixations inside the grouped fixations, calculate their time and then iterate over the fused reading fixations to calculate the total reading fixations time. Every list of the same fixations contains one fixation at least. Based on the pre-processing, those fixations belong to either the same foveal region radius or the same word. Each fixation contains a timestamp, which is used to compute the duration by subtracting the last fixation's timestamp of the list from the first fixation's timestamp. The list of the same fixations will be fused, and the result will be stored. After fusing all the same fixations and storing all their duration, we iterate once again over the fused fixations, but this time to calculate the total reading fixations time. The result is stored in milliseconds. Therefore, we convert the duration from milliseconds to minutes and return the value. It is worth mentioning that this applies to both Bhattacharya and Gwizdka's replication of their implementation and our extended work based on the word level calculation.

4.2.1.3 Total Reading length Calculation

Bhattacharya and Gwizdka's implementation: The input of calculating the reading length feature is the output from the pre-processing phase, which is a list of cleaned and grouped reading fixations. However, we must first identify the reading sequences among these reading fixations to calculate the total reading length feature. The reading sequence in Bhattacharya and Gwizdka's implementation is defined as the sequence of consecutive words. The count of words must be above four that construct a meaningful sentence, meaning the reading sequence that consists of fewer than four words is to be ignored because, in the definition, they do not help to acquire any information. With that in mind, we start identifying the reading sequence by iterating over the list of the grouped fixations. The list of grouped fixations contains a list of fixations within the same foveal region. Each fixation inside the foveal region contains the location of the point where the eyes (left and right eye) were looking at the screen on the x and y-axis, timestamp, and page. So, in the iteration over the list of the grouped fixations, we calculate the center point of the foveal region by computing the mean value of the left and right eyes in both axes, the x and y. After that, we compute the mean value of the same list of fixations in the x and y-axis to get the coordinates of the foveal region. As we move to the following list of fixations in the next foveal region and do the previous calculation, we ensure two conditions to add to the reading sequence. First, we ensure that the following foveal region coordinate is after the previous one by checking its location on the x-axis and comparing it. This is because the following foveal region has to have a greater x value to be considered that it comes after the first foveal region in line. Second, we make they are on the same line by checking the y value of the two foveal regions with 35 pixels margin to the top and bottom, which means it is within the same y bounds for the same line if the two checks are fulfilled we increment the reading sequence count. However, we do not just calculate the length directly because we need to construct the sequence of words, considering the word threshold, which is four. If one of the conditions fails, we reach the end of the reading sequence. When we reach the end of the reading sequence, we first check if the count of reading words surpasses the threshold. If it does not pass, we ignore the reading sequence, start over, and move to the next reading sequence. If it surpasses the threshold, we iterate over these foveal regions, calculate the distances between the points of the foveal regions, add them up, and store the result. Moreover, we proceed again, start over, and move to the next reading sequence. After iterating over all foveal regions, we add all stored values of the reading sequences and return the result in pixels.
Word-level implementation: In our extended work, we also have the same initial proce-

dures. We both take the output of the pre-processing phase as the input to start the total reading length calculation. After that, identifying the words of the reading sequences take place. In our implementation, we count on the reading protocol that enables us to map the eye fixations down to the word level of the web page. Therefore, we do not need to look for the foveal region because we could use that information to decide if fixations are in a different word precisely. We iterate over the list of the grouped fixations, and while were are doing so, we use the reading protocol data to find the word in which the fixation falls and get its width, height, and coordinates on the x and y-axis. Then, if the next fixation is inside the previous box bounds, we append it to a temporary list that contains all fixations within the word box bound. If not, we calculate the mean x and mean y of the fixations inside the word, proceed to the next fixation, and find its new word bounds. At the end of this process, we get a list of fixations grouped based on the word they fall in. a new stage starts in which we look at and verify the reading sequences in the new list of grouped fixations. Unlike Bhattacharya and Gwizdka's implementation, we have three criteria to include a word inside a reading sequence in our word-level implementation. First, The new word should be located after the previous word on the x-axis. Second, the new word falls on the same line as the previous one. Finally, we verify that the fixation is not the end of a reading sequence. We tick the first criteria by checking if the new word has a greater x value by accessing its box coordinates and, specifically, its x-axis and width from the reading protocol. Similarly, for checking the second criteria, if the new and previous words have the same y and height values, they are on the same line. The reading protocol gives us the power to map the fixations to their words, which is done in the pre-processing phase. However, one limitation is to know if the participants looked outside a text frame mid-reading and then continued reading. To overcome this limitation, we marked the fixations in which the participant's eye jumped outside a text bound in the next fixation as the last in the reading sequence. This is because we wanted to ensure that the participant's eyes were on text reading, and by doing that, we can verify the third criteria. The reading sequence is interrupted if any of these criteria do not match. At this point, we check the word threshold. Suppose we have four consecutive words or more in our temporary list. In that case, we calculate the distance between the fixations by adding their mean x and y values, and then we store it and start over to identify a new reading sequence. When we finish the iteration phase, we add up all stored values and return the result in pixels, which is our value for the total reading length.

4.2.1.4 Total Eye Regression Count Calculation

Bhattacharya and Gwizdka's implementation: Regression Count feature calculation has many similarities with calculating total reading length feature. In replicating Bhattacharya and Gwizdka's implementation, the total reading length was subjected to two criteria. First, the new foveal region has a greater mean x value. Second, it is on the same line as the reading fixations sequence. However, to calculate regression count, we need to

reverse the criteria to detect the regression because regression is defined as the eyes moving to the left during the reading, so we need to adjust our criteria to detect when the eyes go back to the left. Like the initial steps of the previous feature implementation (total reading length calculation), we start from the output of the pre-processing phase and start identifying regression sequences. Bhattacharya and Gwizdka considered only regressions after fulfilling a valid reading sequence of at least four words of reading sequences. which means we need to identify a reading sequence in the beginning and then detect a regression. We identify a reading sequence by iterating over the list of the grouped fixations. The list of grouped fixations contains a list of fixations within the same foveal region. We calculate the center point of the foveal region, move to the following list of fixations in the next foveal region, and do the previous calculation. We ensure that the two conditions are fulfilled to add to the reading sequence. After we pass the threshold for a meaningful reading sequence, we change our criteria to meet regression's criteria to detect regression in the eyes during reading. So if the next foveal region is to the left side of the previous foveal region, then that means a regression has happened, and we increment the regression counter. Once we finish iterating over all fixations, we return the incremented variable.
Word-level implementation: On the other hand, the total reading length was subjected to three criteria in our word-level extended work:

1. The new word is on the right side of the previous word.
2. It is on the same line as the reading fixations sequence.
3. The fixation is not marked as last in the reading sequence, which means the following fixation is outside a text frame.

As before, we need to reverse the criteria to detect eye movement to the left during the reading to calculate the regression count. We start from the output of the pre-processing phase by identifying reading sequences. to identify reading sequences, we start with iterating over the list of the grouped fixations. We calculate the word's center point from the fixations inside the word's bound after retrieving its coordinate from the reading protocol. Then, we guarantee that the three requirements are fulfilled to add to the reading sequence. Finally, after we pass the threshold for a meaningful reading sequence of four words, we change our criteria to satisfy regression's criteria to detect regression in the eyes during reading. So if the next word bound is to the left side of the previous word, then that means a regression has happened, and we increment the regression counter. Once we finish iterating over all fixations, we return the incremented variable.

4.2.1.5 Total Regression Length Calculation

Calculating the regression count feature is much like calculating the regression length feature. However, there is a different logic at the last stage of the computations in both Bhattacharya and Gwizdka's and our word-level extended implementation. In Bhattacharya and Gwizdka replicating their implementation, we feed the method with the pre-processing output. We iterate over the list of grouped fixations from the pre-processing phase, and we apply the same two criteria to the grouped fixations before incrementing the count variable of the regression. If the fixations are consecutive, meaning the foveal region of the next fixation is on the right side of the previous one and in the same line on the y-axis, then we add it to the reading sequence. When the reading sequence gets stopped by a new fixation from a different reading sequence, we carry on with a new reading sequence because no regression happens. However, detecting a fixation on the left side of the previous fixation means the participant's eyes did regression. Therefore, we increment the regression counter after ensuring that the current reading sequence fulfills the condition of having at least four foveal region fixations, which make the sequence a meaningful reading sequence. Otherwise, we ignore and look for the following reading sequence. Once we finish iterating over all grouped fixations, we return the regression counter. Similarly, for our word-level implementation, detecting that the new word is on the left side of the previous word means a regression happened. However, before incrementing the regression counter, we ensure that the reading sequence has at least four meaningful fixations, or we ignore it. Then, we start looking for a new reading sequence and try to detect regression in it.

4.2.2 Behavioral Data Processing

The second list of features was designed to explore participants' behavior during their reading and knowledge acquisition. We attempted to discover if there is a particular behavior participants do to enhance their chances of acquiring information faster. To be more precise, we were interested in knowing if the total reading time and dwell time during the learning session affect the Knowledge Gain. In this part, we have one implementation, but what is different is the passed data to the features calculation methods. Passed data can vary between Bhattacharya and Gwizdka's [2] and our word-level implementation depending on the pre-processing step applied to the data-set.

4.2.2.1 Fixating Session Time Calculation

We wanted to calculate the time the participants spent when they were fixating on textual and non-textual content during their web search. To identify the text in web pages, we rely on the reading protocol data to receive all information regarding the website's text's coordinates, width, and height. Based on this information, we will filter the fixations that fall into the exact coordinates. Then, we will start iterating over the fixations to calculate the time between those fixations. It is worth mentioning that after filtering, many fixations are not consecutive. This creates a challenge to calculate precisely the time spent. To tackle this issue, we iterated over the fixations and grouped the consecutive fixations in the data. We calculate the duration of every grouped consecutive fixation by subtracting the time of the last grouped fixation from the first. Then, we store the value, proceed to the next grouped fixations, and compute its duration. At the end of the cycle, we sum up the stored values and return the result. It is crucial to note that the duration is dealt with in milliseconds, but in our feature calculation, we convert the result from milliseconds to minutes and return it.

4.2.2.2 Dwell Time Calculation

We wanted to calculate the time the participants spent when they were not looking at text during their web search. To identify these regions in the websites, we rely on the reading protocol data to receive all information regarding the text's coordinates, width, and height. Based on the retrieved information, we will ignore fixations on words and take fixations outside the text frame. Similar to the previous feature (fixating session time feature), this will make it harder to calculate the dwell time accurately because the filtering creates gaps between the data. We used the same method to group the consecutive data. After that, we iterated over the data, calculated the dwell time for every grouped data, and stored the value to sum it up later at the end of the iteration with the other values. It is essential to mention that the duration is dealt with in milliseconds, but in our feature calculation, we convert the result from milliseconds to minutes and return it.

5 Experiments

In the Experiments chapter, we will talk first in section (5.1) about the Dataset, which includes detailed information about the executed task (5.1.1) and how it was evaluated (5.1.2). Furthermore, Its technical setup (5.1.3), the data that was collected from the study (5.1.4), and finally, a brief explanation of the features we extracted from the collected data to predict knowledge gain based on eye movements and reading patterns (5.1.5). Then, in section (5.2), we will dive deeper into the correlation between the eye-movement features and the pre, and post-test scores and knowledge gain results and analyze and interpret the outcomes. Last but not least, in section (5.3), we will look into knowledge gain prediction, its definition (5.3.1), and the metrics used to evaluate its results (5.3.2), alongside the setup of the experiment (5.3.3) and its final results (5.3.6).

5.1 Dataset

5.1.1 Task Description

The task of the experiment was to learn about the principles of thunderstorms and lightning, and the participants had to solve practical learning tasks that required an understanding of the topic. The experiment were all native German speakers (N= 114, Female = 95, Male = 19, and the average age = 22.88), and all participants were university students. It is worth to mention that, we removed one of the participants from the data-set due to the browsing behaviour. This participant log contained only results from search engines and a multimedia resource website, and to be more precise, the participant's logs were only from Google and YouTube websites, in our work we only want to concentrate on websites that contain textual information. We originally had 114 participants in total. However, we removed six participants due to some technical issues regarding extracting their mapped data from the reading protocol, in addition to that one participant mentioned above due to the searching and learning behavior. Therefore we ended up with data from 107 participants at the end of the cleaning phase. The topic chosen for the experiment is already used in multiple studies related to Knowledge Gain. The first was to study multimedia learning [42], the other was to study search, and eye gaze learning [43], and the last was to study ML features' and classifiers' impact on learning [27]. The topic was chosen because the participants must learn to solve the tasks and understand them, which means they must learn about different metrological and physical concepts to gain knowledge. Moreover, gaining comprehensive knowledge by acquiring procedural and declarative knowledge. Studying is the way to acquire the information to solve the casual task. This helps us achieve our final goal of getting a general idea about eye movement in learning and reading patterns in reading, specifically in Search as Learning domain scenarios.

5.1.2 Procedure and Knowledge Measurements

The experiment was divided into two parts, the first part had to be taken online, and the other had to be done in the laboratory. The participants started with the first part of the experiment, the online part. First, it had a test that consisted of a 10-item multiple-choice and 4-item transfer knowledge test based on prior work [54]. Then, in addition to that, a questionnaire about achievement motivation [18] and web justification beliefs [4]. The participants had to complete it one week before the laboratory appointment. No additional tasks were further asked in the online part. In the laboratory, The participants started the lab appointment by taking assessment tests regarding their working memory capacity [12] and reading comprehension [55]. Then, they were told to write an essay about the formation of thunders and lightning, which is the main topic of the study. The lab part consisted of two phases: the first phase, the learning phase, and the later phase, the After the learning phase. In the learning phase, the participants were instructed to start learning about the study topic (formation of thunders and lightning) by looking on the web for content to learn, and there were no constraints on the type of content they should look for. Moreover, they were told to use any content they preferred to learn. In addition, the participants knew about the rules, which included 30 minutes maximum learning duration, and that they could finish learning whenever they wanted. In the second phase, after the learning, The participants were told to write in a free form everything they knew now about the topic in an essay format. Then, they were told to answer the multiple-choice questionnaires again and take another assessment task for their cognitive reflection [20] and engagement [39].

5.1.3 Technical Environment Setup

The environment of the technical setup consisted of two layers to track learn and search activities of the participants: SMI (SensoMotoric Instruments) ExperimentCenter, the first layer of the experiment, is software that lets us follow the movements of the participants' eyes on the screen through screen recordings and navigation log files. The second layer is plugins to track the browser. Those plugins help us to get and save all saved website files, such as the HTML content, track the navigated URLs and interaction data, such as mouse movement, and save them in local log files. Additionally, "ScrapBook" Plugin was used to give us the chance to save all surfed HTML pages by the participants.

5.1.4 Dataset Structure

The web search learning session helped to generate data about the participants' learning behavior. These Data are divided between, Resource, Behavioral and Knowledge data. They are described by Otto et al. [43] as follows:.

1. Resource Data
   - Screen Recordings: The web search session was recorded, and the recording started when the learning task started. Therefore, all recordings are not longer than 30 minutes due to the task's maximum duration. The video resolution is 1920x1080, 30 frames per second, and in MP4 format. Furthermore, additional multimedia features [42] were provided about the document layout and the image type classification result.
   - HTML Files: There were limitations in saving all online content due to its availability on the web. Therefore, the content of each visited website was recorded. However, due to technical issues, not all pages were successfully stored.
2. Behavioral Data
   - Browsing Timeline: a TSV (Tab Separated Value) file contains all visited websites, with their timestamps since the beginning of the learning session and in seconds, the path to its corresponding HTML file, and the acquisition date. **Table:** (p-id, timestamp, url, html-files, date-of-acquisition)
   - Gaze Data: In the study, the eye movements of the participants were recorded using an eye-tracker.The eye-tracking software is exported from the device and stored as raw data. The I-DT algorithm [53] separated the saccades and fixations by setting the entries via fixation $\in \{0, 1\}$. In the gaze data, it is possible to see values larger than 1080 for the y-coordinates, and that's because the entries are relative to the entire website, not the viewport. However, data may include incorrect entries due to tracking errors, such as negative values. Information is stored in a TSV file, and each participant has their TSV file. The file contains coordinates information of the right and left eye pupils in millisecond precision. It also contains the visible URL at the moment of gazing. **Table:** (p-id, timestamp, left-x, left-y, right-x, right-y, fixation)
   - Browsing Events: Different types of user interactions were recorded, such as.
     - focus: website came into focus.
     - blur: website is not in focus anymore.
     - beforeunload: the website is about to be closed.
     - resize: browser's window resizing, and new window size capturing.

   - scroll: triggered even to log when scrolling happens alongside the scrolling distance, and it can be in both direction horizontal and vertical.
   - mousemove: logging the x and y coordinates of the mouse when they move.
   - Click: tracks click in the location in which the click happened. The clicked HTML element path is logged as well.

  Table: (p-id, timestamp, track-id, type, value, x, y, target, url)

- Browsing Tracks: Tracks: an event that tracks the website the user is on until they move to a different website inside the same browser. The timestamp of the track exists in the TSV file, url, website title, and the viewport dimensions too. Staying time on the track and active time on the track is also included.
  Table: (p-id, timestamp, track-id, url, title, viewport width, viewport height, time stay, time active)

3. Knowledge Data and Questionnaire The state of knowledge of participants was measured several times at specific points as described in "Procedure and Knowledge Measurements'. Furthermore, the cognitive abilities and participants' assessments were taken through tests and questionnaires. as a result, multiple data-sets were generated:

   - demo-knowledge-sum.csv: It includes demographic details and a summary of knowledge scores, such as essays and multiple-choice questions. Since all participants are native German speaking, German screening instruments were used to measure their reading comprehension, and for measuring their working capacity, a reading span task was used. Pardi et al. [44] described more information regarding the span task.
     Table: (p-id, d-sex, d-age, d-field, d-no-sem, d-lang, k-mc-sum-t1, k-mc-sum-t2, kg-mc, essay-C1, essay-C2, kg-essay, LGVT-sped, LGTV-core, WMC-Recalls, WMC-Sentence, CRT-sum)
   - mc-data.csv: This TSV file consists of the scores from the laboratory phase for the multiple choices items before and after the web search task. It also has the confidence question scores, indicating whether participants were guessing their questions or not.
   - essay-data.csv: All written essays before and after the web search are stored in this file.
   - internet-specific-epistimic-justification.csv: The measurement of web specific epistemic justification [4] is contained in this TSV file.
   - selfassessment-data.csv: This file contains the number of correct answers estimation, which is called global self-assessment. It also contains the confidence score of participants of their correct answers and is called a local on-item confidence

rating. Having both a global self-assessment score and a Local on-item score gives us insights into the knowledge of participants' self-assessed performance.

- CRT-data.csv: It contains participants' scores of their cognitive reflection task (CRT [20]) tendency. It was calculated by solving five items of cognitive tasks, and solving more of these tasks means a higher response time.
- achievment-data.csv: This file contains the measures of Hope of Success (HS: range of variables which are useful in learning success) and Fear of Failure (FF) by using a scale that contains ten items and rated from 1 to 4, called achievement motives [18], the scale asses the mentioned (HS, and FF) measurements. to extract the tendency to success or fail in evaluative circumstances.
- dssq-data.csv: The Dundee Stress State Questionnaire [39] was taken by participants. The individual's mean score on these questionnaire items reveals how engaged participants' were during the learning process. In addition, the score can indicate their performance based on task engagement.

4. Reading Protocol Data: The Reading Protocol software [29] was utilized to acquire text-level information about the used gaze data from the experiment. We provided the software with the data set in a CSV (Comma Separated Values) file that included the eye gaze data of the experiment participants, in addition to that the corresponding HTML content of the visited pages, which was recorded using plugins. The Reading Protocol software's outcome was a file containing all text-level information of the pages and mapped with all the eye fixation points of the participants from the gaze data set. It also provided us with the coordinates of the words, width, and height, moreover, the timestamp, participant id, and the stimulus.
   Table: (participant-id, stimulus, texts-map, words-map, timestamp)

The study delivered a vast amount of data to predict knowledge gain. In our work, we have relied on the visited websites' HTML files and Gaze data files of the participants eye movements. Furthermore, the word-level data files from the reading protocol, and finally, the files containing the participants' pre and post-test results and knowledge gain scores.

5.1.5 Eye-movement Features

To assess the efficiency of the eye movement and reading patterns, we extracted seven different features that belong to the eye movements and reading patterns of the participants and their reading behavior:

1. **Fixating session time in minutes:** is the time the participants spent focusing on textual or non-textual content. It is important to clarify here that the purpose of the eye fixations is for the reading purpose. We calculated it by adding up the duration of all reading fixations in a web page .

Table 5.1: Correlation between the features and pre-test, post-test scores and knowledge gain results

Features	pre	post	kg
fixating_session_time_m	**0.271**	**0.162**	**-0.123**
fixation_count	0.173	0.076	-0.102
fixation_duration_m	0.092	0.061	-0.037
regression_count	0.138	0.081	-0.064
regression_length_pixels	0.109	0.051	-0.062
reading_length_pixels	0.151	0.112	-0.048
dwell_time_m	0.140	0.098	-0.050

2. **Fixation count:** is the number of fixations from the reading type on the textual content of a web page, in which the minimum duration of the fixation is at least 113 ms [10].
3. **Fixation duration in minutes:** is the time the participants spent only during their eye fixation on the textual content. It is essential to mention that the fixation is from a reading type.
4. **Regression count:** the number of times the participants' eyes moved back to the left while reading the text. In more precise words, they are the reading fixations that are located on the left side of the previous reading fixation and, after at least four consecutive reading fixations, that form a meaningful sentence.
5. **Regression length in pixels:** is the distance the participants' eyes traveled back to the left during their reading. It is in pixels because participants were reading text on the screen. At least four reading fixations must be consecutive in calculating the feature before detecting the regression in the eyes [11]. The distance will be computed by using the Euclidean formula 4.1.
6. **Reading length in pixels:** is the distance the participants' eyes traveled during their reading. In calculating the feature, at least four reading fixations must be consecutive to calculate the distance the eye traveled between two eye fixations. The distance between the reading fixation in which the regression happened and the previous reading fixation will be calculated by using the Euclidean distance formula 4.1.
7. **Dwell time in minutes:** is the time the participant spends fixating their eyes on non-textual content on a web page during the web session.

5.2 Correlation

We relied on the Pearson Correlation Coefficient to interpret our correlation results between our set of features, pre, post-test, and knowledge gain, which we explained in the Foundation chapter. Our main finding that we can interpret from the correlation table states that the participants who fixated their eyes more on textual and non-textual content had the highest correlation to the pre-test score, meaning they were familiar with the topic. Therefore, it had the highest correlation with the post-test score, resulting in the highest correlation in our table to knowledge gain, with 0.123 in the negative direction. Nevertheless, in the degrees of the Pearson correlation coefficient, it is classified as a weak correlation. Analyzing these results indicates that the participants with higher knowledge gain results were keen to focus more on the web page. Therefore, consuming longer time fixating their eyes on a reading purpose to acquire the information, not necessarily focusing on textual content, but indeed allocating attention for longer fixating. This result also aligns with Pardi et al. [44], which found a negative correlation between the learning outcome and the time spent on text-dominated websites and multimedia resources. The following important finding in our correlation table is the count of the reading fixations. As we can see from the correlation with the pre-test score, participants who were familiar with the topic had a higher reading fixation count, leading to a negative correlation with knowledge gain of -0.102. Therefore, we can interpret that the participants who attempted to read more during the web search session got better results because it can be that they consumed the most textual information due to their eye movements and reading patterns. More fixations more a better reading technique that involves a quick consecutive eye movement to the right and less backward movement to the left, which indicates a regression. One of the surprising discoveries is that the reading length correlates the least with knowledge gain. However, one of the interpretations that can be seen is that participants who were familiar with the topic did not need to read everything closely to acquire knowledge and that they were scanning more in rapid eye movements to learn about the topic. Regression count and regression length have a low correlation with knowledge gain as well. We think the case is similar to our previous interpretation of reading length, which is that the participants who were familiar with the task topic had a better reading technique. Their eyes did not have to go back to re-read the word or sentence and were moving faster to scan the textual content.

5.3 Knowledge Gain Prediction

In this section, we will report our findings for predicting knowledge gain based on eye movement and reading pattern features. The same evaluation settings were used in order to get a fair comparison. Furthermore, the same hyperparameter optimization was used for all experiments. Finally, those settings were applied for our replication of Bhattacharya and Gwizdka's [2] implementation and our word-level implementation with our evaluation procedure. It is essential to mention that in this work, we followed similar steps to what Gritz et al. [27] did due to their clear chapter structuring.

5.3.1 Knowledge Gain Definition

For the Knowledge Gain definition we follow the approach of Gritz et al. [27]. They described the procedure as follows. We relied on the typical procedure of measuring knowledge gain [24, 42] to classify the participants' web search sessions. The classification contains 3 classes C= Low, Moderate, High. The participants were classified based on the Standard Deviation Classification approach. The knowledge gain Xi of participants (i) is z-normalized $(\hat{X}_i)$ as follows in the equation 5.1

$$\hat{X}_i = \frac{X_i - \mu}{\sigma} \tag{5.1}$$

- σ: Standard deviation of all knowledge gain measures X
- μ: Mean of all knowledge gain measures X

For every z-normalized knowledge gain $\hat{X}_i$ the class is given as follows:

$$C(X_i) := \left\{ \begin{array}{cc} Low, & \text{if } \hat{X}_i < -\frac{1}{2} \\ Moderate, & \text{if } -\frac{1}{2} \leq \hat{X}_i \leq \frac{1}{2} \\ High, & \text{if } \hat{X}_i > \frac{1}{2} \end{array} \right\}$$

The class distribution is yielded by: $|X_{High}| = 25, |X_{Moderate}| = 41, |X_{Low}| = 41$

5.3.2 Metrics

The classification outcome was evaluated using recall, accuracy, precision, and F1 score, and they are defined as follows:

$$recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.3}$$

$$precision = \frac{TP}{TP + FP} \tag{5.4}$$

$$F_1 score = \frac{precision * recall}{precision + recall} \tag{5.5}$$

- TP (True Positive): positive values that are classified correctly
- TN (True Negative): negative values that are classified correctly
- FP (False Positive): positive values that are classified incorrectly
- FN (False Negative): false values that are classified incorrectly

5.3.3 Experimental Setup

To assess the efficiency of the classification outcome, we used Cross-validation because each feature vector behaves as a test sample in every single fold. Therefore, we chose 10-fold cross-validation with a 10% test and 90% train and validation set split. Taking this approach, each test set in the cross-validation consists of 10 elements per class. To normalize every feature of the 90%, we use min-max normalization to the interval [0, 1]. Some classifiers, such as Support Vector Machine, require this step. The rest of the set, the 10% test set, is normalized by using the maximum and minimum of 90% for evaluation. Some of the values can fall outside of the [0, 1] limit of the interval. To not lose any information because of normalization, we decided not to clip those values.

5.3.4 Hyper-parameter Optimization

The chosen hyper-parameters influence the performance of the classification algorithm. In our case, we have changeable validation and test data in every iteration due to cross-validation. Therefore the training can not be decided only once and used for the whole evaluation. We executed hyper-parameters optimization in each of the ten iterations to get proper outcomes. To find a good configuration efficiently, we relied on Optuna [1] for the Bayesian search. Moreover, limit the number of runs to 500 to decrease the computing cost. Another split is performed on 90% of the split data. It is split that 90% is the training data and the 10% is validation data. Maximizing the F1 score weight as the optimization objective. This is to avoid the imbalance in class from making the underrepresented class High less influential, as it would be, such as with the overall accuracy.

5.3.5 Classifiers

We counted on several classifiers, which are described in the Foundation chapter, to predict knowledge gain. Therefore, exploring the best configurations and most efficient set of features with the best performing classifier for this task. We used Adaptive Boosting (Adaboost) [21], Decision Tree [6], Random Forest [5], K-Nearest Neighbour [19], Support Vector Machines [13], and Multi-layer Perceptron [52].

5.3.6 Experimental Results

The best-performing classifier for our work in accuracy to predict knowledge gain based on the eye movements and reading pattern from the extracted features was a Multilayer Perceptron classifier (MLP) with 42.1% accuracy, followed by AdaBoost with 34.6%, K-Nearest Neighbor (KNN) with 33.6%, Support Vector Machine (SVM) with 31.8%, Decision Tree with 30.8%, and last but not least Random Forest classifier with 26.2%. MLP classifier

Table 5.2: The Classifiers (clf) we used in our knowledge gain prediction to calculate precision (p), recall (r), f1-score (f1) and accuracy (acc) are: Adaboost (ada), Decision Tree (dt), K-Nearest Neighbor (knn), Multi-layer Perceptron (mlp), Random Forest (rf), and Support Vector Machine (svm).

	Low			**Moderate**			**High**			**Macro**			
clf	**p**	**r**	**f1**	**p**	**r**	**f1**	**p**	**r**	**f1**	**p**	**r**	**f1**	**acc**
ada	36.1	**53.7**	43.1	30.2	31.7	31.0	**66.7**	08.0	14.3	**44.3**	31.1	29.5	34.6
dt	28.3	31.7	29.9	31.7	31.7	31.7	35.0	**28.0**	**31.1**	31.7	30.5	30.9	30.8
knn	29.3	29.3	29.3	40.9	43.9	42.4	27.3	24.0	25.5	32.5	32.4	32.4	33.6
mlp	**43.2**	46.3	**44.7**	**43.1**	**61.0**	**50.5**	20.0	04.0	06.7	35.4	**37.1**	**34.0**	**42.1**
rf	26.4	34.1	29.8	27.9	29.3	28.6	18.2	08.0	11.1	24.2	23.8	23.2	26.2
svm	37.9	26.8	31.4	34.0	41.5	37.4	21.4	24.0	22.6	31.1	30.8	30.5	31.8

was not only the best-performing in accuracy but also in average F1-score and average recall, moreover, the second in average precision. The classes in MLP classifier are distributed as follows: The class High reaches 06.7% in f1-score, 0.04% in recall, and 2% in precision. However, for the class Moderate, the numbers bounce to 50.5% in f1-score, 61% in recall, and 43.1% in precision. Finally, for the class Low, 44.7% in f1-score, 46.3% in recall, and 43.2% in precision. We can see a big difference between the classes Low and Moderate and between the class High. This is expected due to a data-set imbalance regarding the class High, meaning the number of entries for the class High is not equal to the number of entries in the other classes. Therefore, our model gets exposed to less data from the High class, and as a result, we get a lower efficiency in KG prediction. In other words, our classifier will perform poorly in KG prediction for the High class. We can observe the same pattern for the other classifiers in predicting the class High. Except for the Decision Tree classifier, which jumps up to 31.1% in f1-score, 28.0% in recall, and 35.0% in precision, however, this is not an indicator that the classifier can perform well in predicting the knowledge gain in class, as it performs poorly in distinguishing the other two classes. In conclusion, our model can distinguish well between the Low and Moderate classes. However, it performs poorly for the class High. In order to improve the predictability of the class High, we need more data entries for the class High so that data would be balanced and, therefore, the prediction numbers would jump up to the other classes' numbers. Moreover, It can be needed for more features to be included in the model that would help to distinguish the class High from the other two and increase its accuracy, precision, recall, and f1-score.

6 Conclusion

In this final chapter, we arranged our thoughts to briefly discuss the summary of our work in section (6.1). Furthermore, we conclude the limitations of the work and present ideas to explore that would help to overcome the limitations in the future work section (6.2).

6.1 Summary

In this work, we have explored the Search as Learning (SAL) domain that investigates detecting the learning process during a web search. Moreover, we dived into SAL deeper to focus on knowledge gain prediction during a web search, which is one of the goals of this new domain. Along the way, we have researched the work of the prior researchers and looked into the methods they applied to predict knowledge gain during a web search. Moreover, we attempted to replicate the implementation of Bhattacharya and Gwizdka's work [2] by relying on their eye gaze and reading pattern features for the knowledge gain prediction. In addition, we have extended their implementation by introducing a word-level precision method to calculate eye movement and reading pattern features. To achieve this level of precision, we relied on a reading software protocol [29] that maps the eye fixation data on a website page's textual content, allowing us to detect the reading patterns of the participants and differentiate between eye fixations on the textual content or non-textual content. We also proposed a different approach to filter and group eye fixations based on their similarity if they belong to the same word or based on fixation type if they are of a reading type. The results show that it is possible to predict knowledge gain based on how the eyes move on the text on web pages during the reading process. Moreover, we also found a correlation between the participants' reading fixations number and knowledge gain. Those results demonstrate that the reading pattern of the participants and their eye movements play an essential role in determining knowledge gain. That means skillful readers who fixate more on the text and move their eyes back to the left less have a higher gain in knowledge. Based on the eye movement and reading pattern features, we built a machine-learning model and used different classifiers to predict the knowledge gain of the participants. The classifiers' results reveal the ability of the built model to distinguish between the Low and Moderate classes of knowledge gain. However, it was not the case for the class High. One of the reasons can be an imbalance in the data set regarding this class.

6.2 Future Work

Even though our work uses a word-level accuracy to map the fixations, some limitations can be further worked on to improve knowledge gain prediction. One of our limitations was working with an imbalanced data set in the class High that caused low predictability scores for our model to predict high learning gains. To avoid this limitation, more entries from this class can be included in the data set. However, it can also be that more features are needed for this class to increase the knowledge gain prediction score, and our set of features is not enough. Future work can extend the word-level precision of the eye fixations by looking even deeper into the word and fixation to detect the punctuation of the reading sentence and multi-line reading sequences. Additionally, more fixation types, such as scanning or skimming, can be considered, not only reading.

Bibliography

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019.* Ed. by Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis. ACM, 2019, pp. 2623–2631. DOI: `10.1145/3292500.3330701`. URL: `https://doi.org/10.1145/3292500.3330701`.

[2] Nilavra Bhattacharya and Jacek Gwizdka. "Measuring Learning During Search: Differences in Interactions, Eye-Gaze, and Semantic Similarity to Expert Knowledge". In: *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval, CHIIR 2019, Glasgow, Scotland, UK, March 10-14, 2019.* Ed. by Leif Azzopardi, Martin Halvey, Ian Ruthven, Hideo Joho, Vanessa Murdock, and Pernilla Qvarfordt. ACM, 2019, pp. 63–71. DOI: `10.1145/3295750.3298926`. URL: `https://doi.org/10.1145/3295750.3298926`.

[3] Gérard Biau and Erwan Scornet. "A random forest guided tour". In: *Test* 25.2 (2016), pp. 197–227.

[4] Ivar Bråten, Christian Brandmo, and Yvonne Kammerer. "A validation study of the internet-specific epistemic justification inventory with Norwegian preservice teachers". In: *Journal of Educational Computing Research* 57.4 (2019), pp. 877–900.

[5] Leo Breiman. "Random Forests". In: *Mach. Learn.* 45.1 (2001), pp. 5–32. DOI: `10.1023/A:1010933404324`. URL: `https://doi.org/10.1023/A:1010933404324`.

[6] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984. ISBN: 0-534-98053-8.

[7] Andrei Broder. "A taxonomy of web search". In: *ACM Sigir forum.* Vol. 36. 2. ACM New York, NY, USA. 2002, pp. 3–10.

[8] Bahzad Charbuty and Adnan Abdulazeez. "Classification based on decision tree algorithm for machine learning". In: *Journal of Applied Science and Technology Trends* 2.01 (2021), pp. 20–28.

[9] Kittipong Chomboon, Pasapitch Chujai, Pongsakorn Teerarassamee, Kittisak Kerdprasop, and Nittaya Kerdprasop. "An empirical study of distance metrics for k-nearest neighbor algorithm". In: *Proceedings of the 3rd international conference on industrial application engineering.* 2015, pp. 280–285.

[10] Michael J. Cole, Jacek Gwizdka, Chang Liu, Nicholas J. Belkin, and Xiangmin Zhang. "Inferring user knowledge level from eye movement patterns". In: *Inf. Process. Manag.* 49.5 (2013), pp. 1075–1091. DOI: `10.1016/j.ipm.2012.08.004`. URL: `https://doi.org/10.1016/j.ipm.2012.08.004`.

[11] Michael J. Cole, Jacek Gwizdka, Chang Liu, Ralf Bierig, Nicholas J. Belkin, and Xiangmin Zhang. "Task and user effects on reading patterns in information search". In: *Interact. Comput.* 23.4 (2011), pp. 346–362. DOI: 10.1016/j.intcom.2011.04.007. URL: https://doi.org/10.1016/j.intcom.2011.04.007.

[12] Andrew RA Conway, Michael J Kane, Michael F Bunting, D Zach Hambrick, Oliver Wilhelm, and Randall W Engle. "Working memory span tasks: A methodological review and user's guide". In: *Psychonomic bulletin & review* 12.5 (2005), pp. 769–786.

[13] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Mach. Learn.* 20.3 (1995), pp. 273–297. DOI: 10.1007/BF00994018. URL: https://doi.org/10.1007/BF00994018.

[14] Barry De Ville and Padraic Neville. *Decision trees for analytics: using SAS Enterprise miner*. SAS Institute Cary, NC, 2013.

[15] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. "A survey on ensemble learning". In: *Frontiers Comput. Sci.* 14.2 (2020), pp. 241–258. DOI: 10.1007/s11704-019-8208-z. URL: https://doi.org/10.1007/s11704-019-8208-z.

[16] Grant Eckstein, Wesley Schramm, Madeline Noxon, and Jenna Snyder. "Reading L1 and L2 Writing: An Eye-Tracking Study of TESOL Rater Behavior." In: *TESL-EJ* 23.1 (2019), n1.

[17] Carsten Eickhoff, Jaime Teevan, Ryen White, and Susan T. Dumais. "Lessons from the journey: a query log analysis of within-session learning". In: *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*. Ed. by Ben Carterette, Fernando Diaz, Carlos Castillo, and Donald Metzler. ACM, 2014, pp. 223–232. DOI: 10.1145/2556195.2556217. URL: https://doi.org/10.1145/2556195.2556217.

[18] Stefan Engeser. "Messung des expliziten Leistungsmotivs: Kurzform der Achievement Motives Scale". In: *Retrieved* 10 (2005), p. 2010.

[19] Evelyn Fix and Joseph Lawson Hodges. "Discriminatory analysis. Nonparametric discrimination: Consistency properties". In: *International Statistical Review/Revue Internationale de Statistique* 57.3 (1989), pp. 238–247.

[20] Shane Frederick. "Cognitive reflection and decision making". In: *Journal of Economic perspectives* 19.4 (2005), pp. 25–42.

[21] Yoav Freund and Robert E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Computational Learning Theory, Second European Conference, EuroCOLT '95, Barcelona, Spain, March 13-15, 1995, Proceedings*. Ed. by Paul M. B. Vitányi. Vol. 904. Lecture Notes in Computer Science. Springer, 1995, pp. 23–37. DOI: 10.1007/3-540-59119-2_166. URL: https://doi.org/10.1007/3-540-59119-2%5C_166.

[22] Yoav Freund and Robert E. Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *J. Comput. Syst. Sci.* 55.1 (1997), pp. 119–139. DOI: 10.1006/jcss.1997.1504. URL: https://doi.org/10.1006/jcss.1997.1504.

[23] Giovanni Fulantelli, Ivana Marenzi, Qazi Asim Ijaz Ahmad, and Davide Taibi. "SaR-Web-A tool to support search as learning processes". In: *SAL@ SIGIR*. 2016.

[24] Ujwal Gadiraju, Ran Yu, Stefan Dietze, and Peter Holtz. “Analyzing Knowledge Gain of Users in Informational Search Sessions on the Web”. In: *Proceedings of the 2018 Conference on Human Information Interaction and Retrieval, CHIIR 2018, New Brunswick, NJ, USA, March 11-15, 2018*. Ed. by Chirag Shah, Nicholas J. Belkin, Katriina Byström, Jeff Huang, and Falk Scholer. ACM, 2018, pp. 2–11. DOI: `10.1145/3176349.3176381`. URL: `https://doi.org/10.1145/3176349.3176381`.

[25] Souvick Ghosh, Manasa Rath, and Chirag Shah. “Searching as Learning: Exploring Search Behavior and Learning Outcomes in Learning-related Tasks”. In: *Proceedings of the 2018 Conference on Human Information Interaction and Retrieval, CHIIR 2018, New Brunswick, NJ, USA, March 11-15, 2018*. Ed. by Chirag Shah, Nicholas J. Belkin, Katriina Byström, Jeff Huang, and Falk Scholer. ACM, 2018, pp. 22–31. DOI: `10.1145/3176349.3176386`. URL: `https://doi.org/10.1145/3176349.3176386`.

[26] Laura Granka, Matthew Feusner, and Lori Lorigo. “Eye monitoring in online search”. In: *Passive eye monitoring*. Springer, 2008, pp. 347–372.

[27] Wolfgang Gritz, Anett Hoppe, and Ralph Ewerth. “On the Impact of Features and Classifiers for Measuring Knowledge Gain during Web Search - A Case Study”. In: *Proceedings of the CIKM 2021 Workshops co-located with 30th ACM International Conference on Information and Knowledge Management (CIKM 2021), Gold Coast, Queensland, Australia, November 1-5, 2021*. Ed. by Gao Cong and Maya Ramanath. Vol. 3052. CEUR Workshop Proceedings. CEUR-WS.org, 2021. URL: `http://ceur-ws.org/Vol-3052/paper6.pdf`.

[28] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. “Multi-class adaboost”. In: *Statistics and its Interface* 2.3 (2009), pp. 349–360.

[29] Daniel Hienert, Dagmar Kern, Matthew Mitsui, Chirag Shah, and Nicholas J. Belkin. “Reading Protocol: Understanding what has been Read in Interactive Information Retrieval Tasks”. In: *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval, CHIIR 2019, Glasgow, Scotland, UK, March 10-14, 2019*. Ed. by Leif Azzopardi, Martin Halvey, Ian Ruthven, Hideo Joho, Vanessa Murdock, and Pernilla Qvarfordt. ACM, 2019, pp. 73–81. DOI: `10.1145/3295750.3298921`. URL: `https://doi.org/10.1145/3295750.3298921`.

[30] Christoph Hölscher and Gerhard Strube. “Web search behavior of Internet experts and newbies”. In: *Computer networks* 33.1-6 (2000), pp. 337–346.

[31] Anett Hoppe, Peter Holtz, Yvonne Kammerer, Ran Yu, Stefan Dietze, and Ralph Ewerth. “Current challenges for studying search as learning processes”. In: *7th Workshop on Learning & Education with Web Data (LILE2018), in conjunction with ACM Web Science*. 2018.

[32] George K Hung. “Oculomotor Control”. In: *Wiley Encyclopedia of Biomedical Engineering* (2006).

[33] Md Zahidul Islam. “EXPLORE: A novel decision tree classification algorithm”. In: *British National Conference on Databases*. Springer. 2010, pp. 55–71.

[34] Vikramaditya Jakkula. “Tutorial on support vector machine (svm)”. In: *School of EECS, Washington State University* 37.2.5 (2006), p. 3.

[35] SR Jiao, J Song, and B Liu. “A review of decision tree classification algorithms for continuous variables”. In: *Journal of Physics: Conference Series*. Vol. 1651. 1. IOP Publishing. 2020, p. 012083.

[36] Rishita Kalyani and Ujwal Gadiraju. “Understanding User Search Behavior Across Varying Cognitive Levels”. In: *Proceedings of the 30th ACM Conference on Hypertext and Social Media, HT 2019, Hof, Germany, September 17-20, 2019*. Ed. by Claus Atzenbeck, Jessica Rubart, and David E. Millard. ACM, 2019, pp. 123–132. DOI: `10.1145/3342220.3343643`. URL: `https://doi.org/10.1145/3342220.3343643`.

[37] Vojislav Kecman. “Support vector machines–an introduction”. In: *Support vector machines: theory and applications*. Springer, 2005, pp. 1–47.

[38] Xingshan Li, Linjieqiong Huang, Panpan Yao, and Jukka Hyönä. “Universal and specific reading mechanisms across different writing systems”. In: *Nature Reviews Psychology* 1.3 (2022), pp. 133–144.

[39] Gerald Matthews, Sian E Campbell, Shona Falconer, Lucy A Joyner, Jane Huggins, Kirby Gilliland, Rebecca Grier, and Joel S Warm. “Fundamental dimensions of subjective state in performance settings: task engagement, distress, and worry.” In: *Emotion* 2.4 (2002), p. 315.

[40] Felipe Moraes, Sindunuraga Rikarno Putra, and Claudia Hauff. “Contrasting Search as a Learning Activity with Instructor-designed Learning”. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. Ed. by Alfredo Cuzzocrea et al. ACM, 2018, pp. 167–176. DOI: `10.1145/3269206.3271676`. URL: `https://doi.org/10.1145/3269206.3271676`.

[41] Fionn Murtagh. “Multilayer perceptrons for classification and regression”. In: *Neurocomputing* 2.5 (1990), pp. 183–197. DOI: `10.1016/0925-2312(91)90023-5`. URL: `https://doi.org/10.1016/0925-2312(91)90023-5`.

[42] Christian Otto, Ran Yu, Georg Pardi, Johannes von Hoyer, Markus Rokicki, Anett Hoppe, Peter Holtz, Yvonne Kammerer, Stefan Dietze, and Ralph Ewerth. “Predicting Knowledge Gain During Web Search Based on Multimedia Resource Consumption”. In: *Artificial Intelligence in Education - 22nd International Conference, AIED 2021, Utrecht, The Netherlands, June 14-18, 2021, Proceedings, Part I*. Ed. by Ido Roll, Danielle S. McNamara, Sergey A. Sosnovsky, Rose Luckin, and Vania Dimitrova. Vol. 12748. Lecture Notes in Computer Science. Springer, 2021, pp. 318–330. DOI: `10.1007/978-3-030-78292-4\_26`. URL: `https://doi.org/10.1007/978-3-030-78292-4%5C_26`.

[43] Christian Otto et al. “SaL-Lightning Dataset: Search and Eye Gaze Behavior, Resource Interactions and Knowledge Gain during Web Search”. In: *CHIIR ’22: ACM SIGIR Conference on Human Information Interaction and Retrieval, Regensburg, Germany, March 14 - 18, 2022*. Ed. by David Elsweiler. ACM, 2022, pp. 347–352. DOI: `10.1145/3498366.3505835`. URL: `https://doi.org/10.1145/3498366.3505835`.

[44] Georg Pardi, Johannes von Hoyer, Peter Holtz, and Yvonne Kammerer. “The Role of Cognitive Abilities and Time Spent on Texts and Videos in a Multimodal Searching as Learning Task”. In: *CHIIR ’20: Conference on Human Information Interaction and Retrieval, Vancouver, BC, Canada, March 14-18, 2020*. Ed. by Heather L. O’Brien, Luanne Freund, Ioannis Arapakis, Orland Hoeber, and Irene Lopatovska. ACM, 2020, pp. 378–382. DOI: `10.1145/3343413.3378001`. URL: `https://doi.org/10.1145/3343413.3378001`.

[45] Leif E Peterson. “K-nearest neighbor”. In: *Scholarpedia* 4.2 (2009), p. 1883.

[46] Mezbahur Rahman and Qichao Zhang. "Comparison among pearson correlation coefficient tests". In: *Far East J Math Sci (FJMS)* 99 (2016), pp. 237–255.

[47] Hassan Ramchoun, Youssef Ghanou, Mohamed Ettaouil, and Mohammed Amine Janati Idrissi. "Multilayer perceptron: Architecture optimization and training". In: (2016).

[48] Hassan Ramchoun, Mohammed Amine Janati Idrissi, Youssef Ghanou, and Mohamed Ettaouil. "Multilayer Perceptron: Architecture Optimization and Training". In: *Int. J. Interact. Multim. Artif. Intell.* 4.1 (2016), pp. 26–30. DOI: `10.9781/ijimai.2016.415`. URL: `https://doi.org/10.9781/ijimai.2016.415`.

[49] *Random Forest classifier tutorial: How to use tree-based algorithms for machine learning.* `https://www.freecodecamp.org/news/how-to-use-the-tree-based-algorithm-for-machine-learning/`. Accessed: 2022-10-09.

[50] Keith Rayner. "Eye movements in reading and information processing: 20 years of research." In: *Psychological bulletin* 124.3 (1998), p. 372.

[51] Erik D Reichle, Keith Rayner, and Alexander Pollatsek. "The EZ Reader model of eye-movement control in reading: Comparisons to other models". In: *Behavioral and brain sciences* 26.4 (2003), pp. 445–476.

[52] Frank Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms.* Tech. rep. Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[53] Dario D. Salvucci and Joseph H. Goldberg. "Identifying fixations and saccades in eye-tracking protocols". In: *Proceedings of the Eye Tracking Research & Application Symposium, ETRA 2000, Palm Beach Gardens, Florida, USA, November 6-8, 2000.* Ed. by Andrew T. Duchowski. ACM, 2000, pp. 71–78. DOI: `10.1145/355017.355028`. URL: `https://doi.org/10.1145/355017.355028`.

[54] Florian Schmidt-Weigand and Katharina Scheiter. "The role of spatial descriptions in learning from multimedia". In: *Comput. Hum. Behav.* 27.1 (2011), pp. 22–28. DOI: `10.1016/j.chb.2010.05.007`. URL: `https://doi.org/10.1016/j.chb.2010.05.007`.

[55] Wolfgang Schneider, Matthias Schlagmüller, and Marco Ennemoser. *LGVT 6-12: Lesegeschwindigkeit und-verständnistest für die Klassen 6-12.* Hogrefe Göttingen, 2007.

[56] Patrick Schober, Christa Boer, and Lothar A Schwarte. "Correlation coefficients: appropriate use and interpretation". In: *Anesthesia & Analgesia* 126.5 (2018), pp. 1763–1768.

[57] ER Sruthi. "Understanding Random Forest". In: *Data Science Bloagathon.* Analytics Vidhya, 2021.

[58] *SVM: Feature Selection and Kernels.* https://towardsdatascience.com/svm-feature-selection-and-kernels-840781cc1a6c.

[59] Rohail Syed, Kevyn Collins-Thompson, Paul N. Bennett, Mengqiu Teng, Shane Williams, Wendy W. Tay, and Shamsi T. Iqbal. "Improving Learning Outcomes with Gaze Tracking and Automatic Question Generation". In: *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020.* Ed. by Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen. ACM, 2020, pp. 1693–1703. DOI: `10.1145/3366423.3380240`. URL: `https://doi.org/10.1145/3366423.3380240`.

[60] Rui Tang, Ran Yu, Markus Rokicki, Ralph Ewerth, and Stefan Dietze. "Domain-Specific Modeling of User Knowledge in Informational Search Sessions". In: *Proceedings of the CIKM 2021 Workshops co-located with 30th ACM International Conference on Information and Knowledge Management (CIKM 2021), Gold Coast, Queensland, Australia, November 1-5, 2021.* Ed. by Gao Cong and Maya Ramanath. Vol. 3052. CEUR Workshop Proceedings. CEUR-WS.org, 2021. URL: `http://ceur-ws.org/Vol-3052/paper8.pdf`.

[61] Suryakanthi Tangirala. "Evaluating the impact of GINI index and information gain on classification using decision tree classifier algorithm". In: *International Journal of Advanced Computer Science and Applications* 11.2 (2020), pp. 612–619.

[62] Zhuo Wang, Jintao Zhang, and Naveen Verma. "Realizing low-energy classification systems by implementing matrix multiplication directly within an ADC". In: *IEEE transactions on biomedical circuits and systems* 9.6 (2015), pp. 825–837.

[63] *What is a decision tree how to make one.* `https://venngage.com/blog/what-is-a-decision-tree/`. Accessed: 2022-10-09.

[64] Ran Yu, Ujwal Gadiraju, and Stefan Dietze. "Detecting, Understanding and Supporting Everyday Learning in Web Search". In: *CoRR* abs/1806.11046 (2018). arXiv: `1806.11046`. URL: `http://arxiv.org/abs/1806.11046`.

[65] Ran Yu, Ujwal Gadiraju, Peter Holtz, Markus Rokicki, Philipp Kemkes, and Stefan Dietze. "Predicting User Knowledge Gain in Informational Search Sessions". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018.* Ed. by Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz. ACM, 2018, pp. 75–84. DOI: `10.1145/3209978.3210064`. URL: `https://doi.org/10.1145/3209978.3210064`.

[66] Wei Zhang, Xiaohui Chen, Yueqi Liu, and Qian Xi. "A distributed storage and computation k-nearest neighbor algorithm based cloud-edge computing for cyber-physical-social systems". In: *IEEE Access* 8 (2020), pp. 50118–50130.

[67] Yao Zhang and Chang Liu. "Users' Knowledge Use and Change during Information Searching Process: A Perspective of Vocabulary Usage". In: *JCDL '20: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, Virtual Event, China, August 1-5, 2020.* Ed. by Ruhua Huang, Dan Wu, Gary Marchionini, Daqing He, Sally Jo Cunningham, and Preben Hansen. ACM, 2020, pp. 47–56. DOI: `10.1145/3383583.3398532`. URL: `https://doi.org/10.1145/3383583.3398532`.