A MinHash approach for fast scanpath classification

David Geisler
University of Tuebingen
Germany
david.geisler@uni-tuebingen.de

Nora Castner
University of Tuebingen
Germany
nora.castner@uni-tuebingen.de

Gjergji Kasneci
University of Tuebingen
Germany
gjergji.kasneci@uni-tuebingen.de

Enkelejda Kasneci
University of Tuebingen
Germany
enkelejda.kasneci@uni-tuebingen.de

ABSTRACT

The visual scanpath describes the shift of visual attention over time. Characteristic patterns in the attention shifts allow inferences about cognitive processes, performed tasks, intention, or expertise. To analyse such patterns, the scanpath is often represented as a sequence of symbols that can be used to calculate a similarity score to other scanpaths. However, as the length of the scanpath or the number of possible symbols increases, established methods for scanpath similarity become inefficient, both in terms of runtime and memory consumption. We present a MinHash approach for efficient scanpath similarity calculation. Our approach shows competitive results in clustering and classification of scanpaths compared to established methods such as Needleman-Wunsch, but at a fraction of the required runtime. Furthermore, with time complexity of $O(n)$ and constant memory consumption, our approach is ideally suited for real-time operation or analyzing large amounts of data.

CCS CONCEPTS

• **Human-centered computing** → **User models**; **User studies**; **Heuristic evaluations**; **Laboratory experiments**.

KEYWORDS

Eye-Tracking; Visual Stimulus; Visual Perception; Scene Evaluation; Sensor Fusion

ACM Reference Format:
David Geisler, Nora Castner, Gjergji Kasneci, and Enkelejda Kasneci. 2020. A MinHash approach for fast scanpath classification. In *ETRA '20: ACM Symposium on Eye Tracking Research & Applications, June 2–5, 2020, Stuttgart, Germany.* ACM, New York, NY, USA, 9 pages. https://doi.org/00.0000/0000000.0000000

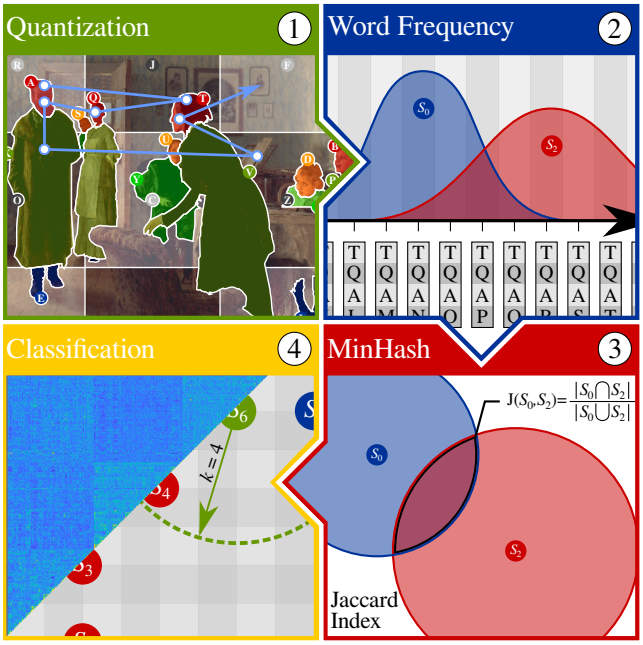

Figure 1: Workflow of the proposed MinHash approach in 4 steps: ① The recorded scanpath is quantified using RoIs on the stimulus. ② Extraction and permutation of subsequences. ③ Approximate the Jaccard Index by randomly chosen pattern matches using MinHash ④ Extract features for scanpath classification.

1 INTRODUCTION

Our eyes move around to perceive and understand the world around us in order to compensate for our limited- though clearest- foveal vision. When viewing the world, we frequently focus our attention, otherwise known as a fixation, before shifting to another area with a rapid eye movement known as a saccade. The way how we explore a scene is highly affected by our previous knowledge or expertise, cognitive state or task, and even personality [Braunagel et al. 2017; Eivazi et al. 2017; Gandomkar et al. 2018; Hoppe et al. 2018].

Besides representing cognitive states, the user's scene exploration can also be thought of as a modality of an adaptive system. One effort towards smarter interaction with systems is automated scanpath analysis, in order to infer more information about the user or the context (e.g. task-related intentions, expertise level, etc.). For example, in the automotive context, efficient scanpath analysis in an online fashion was employed to recognize the drivers' level of awareness [Braunagel et al. 2017]. Or in the context of HCI, characteristic scanpaths can

ETRA '20, June 2–5, 2020, Stuttgart, Germany

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/00.0000/0000000.0000000

be used as input to a computer interface [Esteves et al. 2015]. For example gaze-based systems for smart home control offers a new means of utilization for persons with mobile disabilities [Bissoli et al. 2019]. Furthermore, using the scanpath information as an additional modality in adaptive systems can offer new ways for customized user-interface designs. For instance, gaze analysis was often proposed as a measure for adaptive training of medical personnel [Van der Gijp et al. 2017]. However, actually working training procedures are still scarce. The main reason could be that the existing models do not generalize well. Usually they are trained and evaluated on a certain data set. The extracted features and patterns are then very specific to the classification task and cannot be transferred to other application domains. For a better generalization it is necessary to build models using data mining from large amounts of scanpaths from different application domains.

A wide spread approach in scanpath comparison and classification is to encode the scanpath as string. Subsequently, multiple scanpaths are clustered and classified by their string similarity to each other. However, the performance of those string based approaches depends strong on the information encoded in the string. At the same time, the more information are coded into the string, the larger the alphabet and string length gets. In a driving scenario, for instance, the number of relevant AoIs can be extremely high: different road users, road signs, instruments, mirrors, etc. At the same time, many applications in the driving domain require an online operating mode. But even in offline mode, depending on the scenario, hundreds of AoI transitions can quickly be observed. This, in turn, leads to long delays in the analysis of the encoded strings. Consequently, researchers and developers are forced to either reduce the number of AoIs, shorten the examination periods or reduce the sample size.

We introduce MinHash, as fast scanpath comparison and classification tool. Our approach equips the community with the ability to efficiently analyze patterns of extremely large scanpaths, and thus allows researchers more freedom in the string encoding of scanpaths and mining patterns in huge amount of dataset. Our Approach reaches classification rates, close to the state-of-the-art, but at moderate and controllable runtime. Additionally, we provide the source and ready to run binaries for linux on our GIT repository:

https://atreus.informatik.uni-tuebingen.de/geislerd/pe-tools-scanpath

In the following section we recapitulate the state-of-the-art in scanpath analysis, with focus on string based approaches. In section 3 the proposed MinHash approach is described in detail, and how it can be used in order to classify scanpaths based on their similarity to each other. Section 4 provides the evaluation of the proposed approach. Two in terms of classification accuracy, and one about runtime. Finally, section 5 and 6 state the limitations of our approach and the final remarks.

2 RELATED WORK

Scanpath analysis is part of the standard repertoire of every eye tracking researcher. Accordingly, there are already a multitude of different approaches ranging from attention comparison to mixture models. The review from Anderson et. al. gives a comprehensive overview of the state-of-the-art in 2014 [Anderson et al. 2015]. In the following, we concentrate on the fundamental scanpath analyzing tools and complement Anderson et. al.'s review with more recent approaches.

Scanpath Similarity Besides the visual and manual evaluation of scanpaths [Berger et al. 2012; Kübler et al. 2016; Raschke et al. 2014], the most common automatic assessment is by their similarity to each other.

One of the most well known methods to calculate the scanpath similarity is MultiMatch. MultiMatch calculates the scanpath distance in multiple dimensions, e.g. location and duration. The strength of MultiMatch is that it can be applied directly to the scanpath (in the form of x-y coordinates). The similarity of two scan paths is determined purely by their attributes. Shifts in the eye tracking signal, as well as reversed fixation orders, can be compensated by MultiMatch, making it more robust [Dewhurst et al. 2012; Foulsham et al. 2012; Jarodzka et al. 2010]. However, it also means that semantic information is not included in the scanpath distance.

Similarly, Eyenalysis maps the individual fixations of a scanpath to the closest of a second scanpath. The similarity between the two scanpaths is then calculated by the spatial distance of the mapped fixations [Mathôt et al. 2012].

FuncSim, on the other hand, divides the scanpath into functional sub units and calculates the similarity of two scanpaths within the sub units [Foerster and Schneider 2014]. The subdivision of the scanpath allows the input of semantic information.

String Similarity Many approaches use a string representation in order to encode various scanpath properties and semantics into a one dimensional signal (see figure 2 as example). Despite its weaknesses due to possible quantization errors, these approaches are still very successful in the calculation of scanpath similarity and subsequent classification of cognitive states, e.g. task recognition [Kübler 2016].

The mapping of the scanpath to a string is usually done by Regions of Interest (RoIs) in the stimulus. The RoIs can either be calculated automatically: e.g. using a static grid, fixation clusters, or with image processing methods [Fuhl et al. 2018a,b; Heminghous and Duchowski 2006; Kübler et al. 2014]. They can also be labeled manually, which is generally a more accurate representation of the image or task semantics.

The gold standard in the calculation of string similarities is the Needleman-Wunsch algorithm, which calculates the costs of gap operations in both strings to align one string to another [Needleman and Wunsch 1970]. The Levenstein distance can be considered a special case of the Needleman-Wunsch algorithm. In contrast to the Needleman-Wunsch, only operations of the length one are allowed, which reduces the runtime complexity from $O(n^3)$ to $O(n^2)$ [Levenshtein 1966]. These approaches are highly investigated for scanpath comparison, and work very well on short scanpaths [Busjahn et al. 2015; Castner et al. 2018; Cristino et al. 2010; Day 2010; Deitelhoff et al. 2019].

However, the string editing distance has some drawbacks: First, the runtime is not acceptable for long scanpaths. Second, the sequence of symbols in the string usually represents the fixation sequence. It is known that especially patterns in the transition between fixations contains valuable information about the executed task or expertise [Castner et al. 2018]. However, such patterns are not reflected in a string distance.

Subsequence matching SubsMatch takes this into account and assesses the similarity of two scanpaths by the frequency of subsequences. Therefore, the string is subdivided into subsequences using a sliding window. Each extracted subsequence is counted in a dictionary. The similarity of two scanpaths is then calculated as the distance over

the frequencies of the subsequence [Kübler et al. 2014, 2017]. These subsequences are particularly significant with regard to various classification problems, such as performed task or expertise [Castner et al. 2018; Kübler 2016; Kübler et al. 2015; Kübler and Kasneci 2015].

However depending on the length of the extracted subsequences and variety of symbols in the string, the dictionary may become extremely large. Thus, iterating over a union of two dictionaries to calculate the frequency difference can be very costly. Conversely, users are restricted in the number of used RoIs, or in the length of the examined subsequences, in order to finish their work on time.

To reveal the full power of subsequence matching, our approach tackles the runtime challenge: Using MinHash. MinHash allows to calculate the similarity between the scanpaths efficiently as a Jaccard Index approximation of string's subsequences.

3 METHOD

MinHash is a widely used approach in the field of data mining and is mostly used for clustering and classifying documents [Manaa and Abdulameer 2018]. We adopted this approach to calculate scanpath similarities in order to cluster and classify them subsequently. Therefore, we quantify the scanpath on the stimulus using RoIs and extract subsequences. The frequency of the subsequences is then used to estimate the Jaccard Index of two scanpaths using MinHash. The Jaccard Index of the scanpaths to each other is then used to classify new scanpaths.

We have divided our approach into four successive steps as shown in Figure 1, which will be introduced and discussed in the following.

Step ① First, the gaze signal is quantized to a string of symbols S. The quantification can be done using a static grid over the stimulus – or if available – based on RoIs. Every time a fixation falls into a grid cell or RoI, S gets expanded by the corresponding symbol. In addition to spatial representation, other attributes such as fixation duration, saccade length or direction can also be used for quantification. For example, the fixation period is often represented as repetition of the same symbol. Or depending on the saccade length a extra symbol is attached.

Figure 2 shows an exemplary quantification of a scanpath on the "Unexpected Visitor" image from the Yarbus experiment. It combines the grid for the background with semantic RoIs on for the foreground.

Step ② Extract Word Frequencies Our method assesses the similarity of two scanpaths based on the frequency of visual scanpath subsequences. In the following steps, we will refer to a subsequence as word w. In the simplest case, a word of n consecutive symbols represents the sequence of fixation transitions. Yet, the symbol does not only represent the spatial region of the stimulus, but it also includes semantic information, and the interpretation of such a word becomes more significant. For example, the illustrated in Figure 2, the word `ATQ` not only contains the information that the fixations has fallen into a certain area of the stimulus, but also implies that there were transitions between different head-related RoIs. These patterns are known to be particularly descriptive in terms of various classification tasks, such as predicting the currently performed task of the subject [Castner et al. 2018; Kübler 2016; Kübler et al. 2015; Kübler and Kasneci 2015].

To extract the words in a scanpath, our method uses a sliding window approach. We also suggest to not only extract words of a fixed length. Our approach allows to extract words of different length and with gaps. Thus, scanpath subsequences are matched even with small

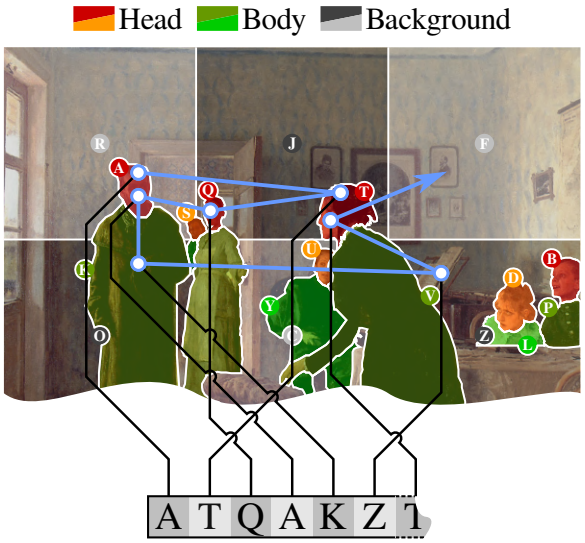

Figure 2: Quantification of the scanpath shown in blue using semantic (red and green) and spatial RoIs (gray).

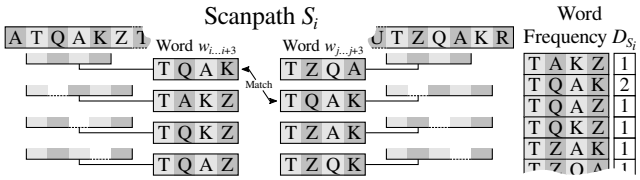

Figure 3: Shows how the words with a length of four symbols and a maximum of one gap are extracted from the scanpath S_i through a sliding window. Each extracted word is counted in the Dictionary D_{S_i}. The sub sequence `TZQAK` is mapped to `TQAK` by the gap at `Z`, leading to a frequency of two for the word `TQAK` in D_{S_i}.

variations. Figure 3 shows an example of the extraction of words with a length of four ($w_{\min} = w_{\max} = 4$) and one optional gap (w_{gap}). In addition, it is usually advantageous to scan the scanpath backwards for words. Thus, subsequences can be depicted to the same word, even if they differ in one place or are reversed.

Each extracted word w of the scanpath S is counted in the Dictionary D_S in form of a (chained) hash table with n_b buckets and the hash function $h(w) \longrightarrow [0, n_b)$. Until this step, the procedure is similar to SubsMatch. In the next step, SubsMatch would calculate the distance of the dictionaries $D_{S_0},\ldots,D_{S_n}$ with each other using any distance metric [Kübler et al. 2014], or feed it directly as a feature vector in a machine learning process [Kübler et al. 2017]. However, it is necessary to iterate over all words present in $D_{S_0},\ldots,D_{S_n}$. Depending on the alphabet size n_a and extracted word lengths, this can become extremely costly: $\sum_{w_n=w_{\min}}^{w_{\max}} n_a^{w_n}$. This extreme case becomes more likely with increasing scanpath length. Instead, MinHash compares n_h random word frequencies. Therefore, the extracted words are additionally indexed in n_h hash tables $D_{H_0},\ldots,D_{H_{n_h-1}}$ using the disjunct hash functions $h_0(w),\ldots,h_{n_h-1}$.

Algorithm 1 shows this step as pseudo code. Extracting and counting the word frequency from S needs to iterate over the whole scanpath, which leads to a runtime $O(n)$. Accessing hash tables for a constant word length is a constant operation. The inner loop does not impact the runtime since n_h is constant.

Algorithm 1: Word frequency counting

Input: S
Data: $D_{H_{0\dots n_h-1}}$
Result: D_S
1 **foreach** $w \in S$ **do**
  /* Increase the frequency of the word w in the scanpath dictionary D_S */
2 $D_S(h_0(w)) \longleftarrow D_S(h_0(w))+1$
3 **foreach** $D_{H_i} \in D_{H_{0\dots n_h-1}}$ **do**
4 **if** $h_i(w) \notin D_{H_i}$ **then**
  /* Add word w to the global dictionary D_{H_i} */
5 $D_{H_i}(h_i(w)) \longleftarrow w$

Step ③ MinHash The underlying idea is that for a optimal (collision free) hash function, h applies:

$$J_{D_{S_k},D_{S_l}} = P\left(h_{\min}(D_{S_k}) = h_{\min}(D_{S_l})\right), \quad (1)$$

where $J_{D_{S_k},D_{S_l}}$ is the Jaccard index of the words in the dictionary D_{S_k} and D_{S_l}. $h_{\min}(D)$ extracts the minimal hash value of the words in the dictionary D. $P(\dots)$ is the probability that the minimal hash value of the dictionary D_{S_k} is equal to the minimal hash value of D_{S_l}. In other words, the probability that the first word in the hash table of the dictionary D_{S_k} is equal to the first word in the hash table of the Dictionary D_{S_l} is equal to the Jaccard Index of D_{S_k} and D_{S_l}.

Then, the probability $P(\dots)$ can be approximated by building many dictionaries $D_S^{(0)},\dots,D_S^{(n_h)}$ using different hash functions $h_0(w),\dots,h_{n_h}(w)$:

$$P\left(h_{\min}(D_{S_k}) = h_{\min}(D_{S_l})\right) \approx \sum_{i=0}^{n_h} \text{match}\left(D_{S_l}^{(i)}(0), D_{S_k}^{(i)}(0)\right) \cdot n_h^{-1}, \quad (2)$$

with

$$\text{match}\left(D_{S_l}^{(i)}(0), D_{S_k}^{(i)}(0)\right) = \begin{cases} 1 & \text{if } D_{S_l}^{(i)}(0) = D_{S_k}^{(i)}(0) \\ 0 & \text{otherwise} \end{cases}, \quad (3)$$

where $D^{(i)}(0)$ is the first entry of the hash table in the dictionary D using the hash function $h_i(w)$. However in our case, words may occur multiple times in a scanpath. Therefore, we deal with word frequencies. Instead of checking whether a word is present in both scanpaths, the match$(\dots)$ function should estimate whether the frequency of the both words matches and how it contributes to the Jaccard Index. For this purpose, the word frequencies are binarized by means of n_t randomly selected hyperplanes T. If both word frequencies are on the same side of the hyperplane, it is counted as a match and the Jaccard Index is increased proportionally. Figure 4 visualizes this procedure using four hyperplanes $T_0,\dots,T_3$. The closer the respective word frequencies are to each other, the more often they fall on the same side of the hyperplane, leading to a higher Jaccard Index. The match$(\dots)$ function is, therefore, redefined as following:

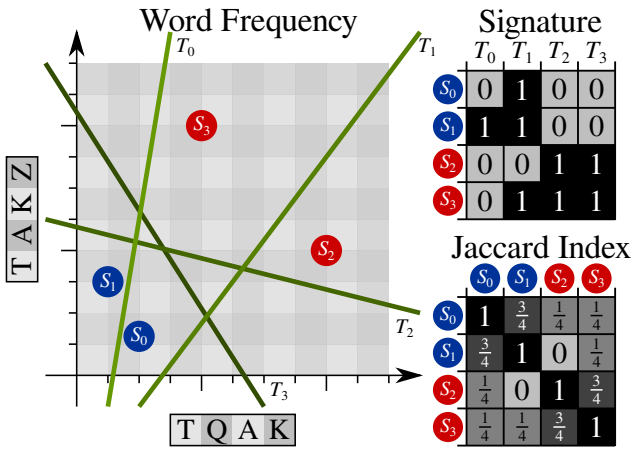

Figure 4: Shows the application of random hyperplanes in the frequency domain of the words `TAKZ` and `TQAK` for four exemplary scanpaths $S_0,\dots,S_3$. The Jaccard index is calculated from the Signature matrix, which indicates the side of the scanpath S_i regarding to a hyperplane T_i.

$$\text{match}\left(D_{S_l}^{(i)}(0), D_{S_k}^{(i)}(0)\right) = \sum_{j=0}^{n_t} \text{thresh}\left(D_{S_l}^{(i)}(0), D_{S_k}^{(i)}(0), T_j\right) \cdot n_t^{-1}, \quad (4)$$

where thresh$(\dots)$ returns 1 if both word frequencies are on the same side of the hyperplane T_j, otherwise 0.

For the calculation of many scanpaths, it is necessary to build n_h dictionaries for each scanpath, which leads to extreme memory consumption. For the application on common hardware it is advisable to do this calculation once for all scanpaths together: $D_{H_0},\dots,D_{H_{n_h-1}}$. These dictionaries also contain words that are not part of the scanpath pair D_{S_k},D_{S_l}. Thus, it is necessary to iterate over D_{H_i} until the current word is contained in D_{S_k} or D_{S_l}; but the resulting overhead is small and reduces for longer scanpaths (see the runtime evaluation in section 4).

Algorithm 2 shows the implementation of the proposed MinHash approach. In addition to the weighting by the hyperplanes, $w_{\text{weight}}^{|w|-w_{\min}}$ also includes the word length $|w|$. The idea is that longer words occur less often in two scanpaths, but are more meaningful about similarity. c accumulates all weights and normalizes the estimated Jaccard Index at the end into the value range between 0 and 1. Please note that c also provides a good confidence measure about the approximation of the Jaccard Index $J_{D_{S_k},D_{S_l}}$.

Step ④ Classification The classification of an unknown scanpath S_{test} can be realized over the Jaccard Index to the trainings scanpaths $\mathbf{S}_{\text{train}} = (S_0,\dots,S_n)$ and their related class labels $\mathbf{C}_{\text{train}} = (C_0,\dots,C_n)$. Then the majority class label of the k most similar trainings scanpath is predicted as C_{test}: a kNN classifier.

However, considering the similarity matrices in figure 6 (ETRA challenge 2019), it is apparent that one class of scanpaths (Natural) does not form a distinct cluster. Rather, the similarity of the scanpaths seems to be equally similar to all others. The prediction of scan paths

Algorithm 2: MinHash

Input: D_{S_k},D_{S_l}
Data: $D_{H_{0\dots n_h-1}}$
Result: $J_{D_{S_k},D_{S_l}}$

$c \longleftarrow 0$ and $J_{D_{S_k},D_{S_l}} \longleftarrow 0$
foreach $D_{H_i} \in D_{H_{0\dots n_h-1}}$ **do**
  foreach $w \in D_{H_i}$ **do**
    /* Only consider words that occur at least in one of the two scanpaths S_i and S_j. */
    if $D_{S_k}(h_i(w)) > 0 \vee D_{S_l}(h_i(w)) > 0$ **then**
      /* Weighting of the match/mismatch found based on the word length. */
      $f \longleftarrow w_{\text{weight}}^{|w|-w_{\min}}$
      /* Count all weights for later normalization. */
      $c \longleftarrow c+f$
      /* Apply random hyperplane thresholds $T_0,\dots,T_{n_t-1}$ for local sensitivity. */
      foreach $T \in T_{0\dots n_t-1}$ **do**
        if $D_{S_k}(h_i(w)) > T = D_{S_l}(h_j(w)) > T$ **then**
          /* S_k and S_l on the same side of the hyperplane T. Increase Jaccard Index approximation $J_{D_{S_k},D_{S_l}}$. */
          $J_{D_{S_k},D_{S_l}} \longleftarrow J_{D_{S_k},D_{S_l}} + f \cdot n_t^{-1}$
        break
$J_{D_{S_k},D_{S_l}} \longleftarrow J_{D_{S_k},D_{S_l}} \cdot c^{-1}$

of this class by the k nearest neighbors would likely fail. As long as there is only one such weakly pronounced class, however, this can be handled with a small trick. Instead of choosing the k most similar scanpaths, the k scanpaths with the most similar Jaccard Indices are chosen. Therefore, a similarity matrix $\mathbf{J}_{\text{train}}$ over all training scanpaths is calculated. Each row and column of the matrix $\mathbf{J}_{\text{train}}$ corresponds to one training scanpath. Thus, the element at the row r and column c represents the Jaccard Index $J_{D_{S_r},D_{S_c}}$. Next, the Jaccard Index of the scanpaths S_{test} is calculated to all training scanpaths $\mathbf{S}_{\text{train}}$: $\mathbf{J}_{\text{test}} = J_{D_{S_0},D_{\text{test}}},\dots,J_{D_{S_n},D_{\text{test}}}$. For the classification step, the k most similar (euclidean distance) rows in $\mathbf{J}_{\text{train}}$ to $\mathbf{J}_{\text{test}}$ are determined. Since, each row in $\mathbf{J}_{\text{test}}$ corresponds to a training scanpath, and thus to Class label, the predicted class label can be determined by a majority decision. Figure 5 shows the procedure with regard to the Jaccard similarity of the seven scanpaths $S_0,\dots,S_6$ to S_0 and S_1.

Yet, this approach can also be used with other classifiers instead of kNN. Algorithm 3 shows an abstract workflow to train and classify scanpaths based on the Jaccard Similarty. The functions fit() and predict() can be exchanged with any classification method that supports multidimensional features; which is beneficial especially if the classification must be in real time. kNN has the disadvantage of a lazy-learning approach. Meaning, the modelling does not take place during training, but at the time of prediction. An eager-learning approach, like the SVM, creates a model (hyperplane) while training, and does not run over all training data for the prediction. We had favorable outcomes using a linear SVM and tree classifier. Yet, the performance

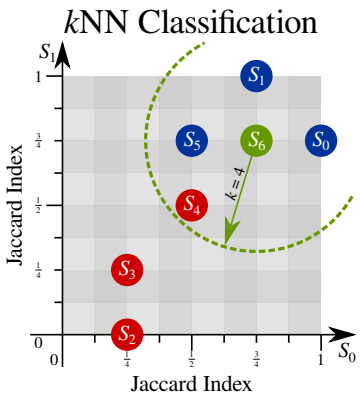

Figure 5: kNN classification using the Jaccard Index of the scanpath S_0 (horizontal axis) and S_1 (vertical axis) to the scanpaths S_0, … , S_6. The new scanpath S_6 would be classified as blue, since three of the four nearest neighbors correspond to this class.

of these methods depends strongly on its parametrization, we limited ourselves to kNN in the following to keep the evaluation neat and clear.

Algorithm 3: Classification

Input: $S_0,\dots,S_n$ and $C_0,\dots,C_n$ and S_{test}
Result: C_{test}

/* The feature vector to train the classifier is the square matrix $\mathbf{J}$ of the pairwise Jaccard Index $J_{D_{S_k},D_{S_l}}$ of the training scanpaths $S_0,\dots,S_n$, respective their dictionaries $D_{S_0},\dots,D_{S_n}$. Please note $J_{D_{S_k},D_{S_l}} = J_{D_{S_l},D_{S_k}}$. */

$$\mathbf{J}_{\text{train}} \longleftarrow \begin{pmatrix} 1 & J_{D_{S_0},D_{S_1}} & \cdots & J_{D_{S_0},D_{S_n}} \\ J_{D_{S_1},D_{S_0}} & 1 & \ddots & \vdots \\ \vdots & \ddots & 1 & J_{D_{S_{n-1}},D_{S_n}} \\ J_{D_{S_n},D_{S_0}} & \cdots & J_{D_{S_n},D_{S_{n-1}}} & 1 \end{pmatrix}$$

$\mathbf{C}_{\text{train}} \longleftarrow \{C_0,\dots,C_n\}$ $M \longleftarrow \text{fit}(\mathbf{J}_{\text{train}},\mathbf{C}_{\text{train}})$

/* Analogous to the training input $\mathbf{J}_{\text{train}}$, the test vector is assembled as the pairwise Jaccard Index $\mathbf{J}_{\text{test}}$ of the training scanpaths $\mathbf{S}_{\text{train}}$ to the test scanpath S_{test} */

$\mathbf{J}_{\text{test}} \longleftarrow \begin{pmatrix} J_{D_{S_0},D_{S_{\text{test}}}} & \cdots & J_{D_{S_n},D_{S_{\text{test}}}} \end{pmatrix}$

$C_{\text{test}} \longleftarrow \text{predict}(M,\mathbf{J}_{\text{test}})$

Implementation details In the following we give some short remarks to the implementation details, in order to optimize the traceability in a practical application of our approach:

(1) MinHash makes extensive use of hash tables. In the most standard libraries. they are implemented as chained [Leiserson et al. 2001]. Meaning that n_b buckets are reserved. The hash function assigns any word to one of these buckets. Collisions are solved by maintaining a linked list in each bucket. Additionally, a second linked list is maintained referencing active (non empty) buckets. This enables fast iteration over the contained elements. However, it implicate two issues: (i) no permutation inside a bucket → the permutation is limited by n_b. It is therefore advisable to select n_b large enough depending on the number of different words in a scanpath. (ii) The order of the bucket activation is reconstructed by the active bucket list.

Together with (i), the insertion order is partly reconstructed when iterating. For algorithm 2, it must assured to iterate over the buckets in their logical order, not in their acivation order.

(2) The random hyperplanes in algorithm 2 can be efficiently approximated by a cosine similarity [Singhal et al. 2001].

(3) Our approach benefits from the fact that almost all steps can be parallelized: The extraction of words by using different threads for each window size and gap position. The MinHash by using dedicated threads for each permuted dictionary D_{H_i}. And the pairwise calculation of the Jaccard Index.

4 EVALUATION

The evaluation does not focus on the absolute classification accuracy, but rather on how close the classification accuracy of MinHash is to the state of the art, at a shorter runtime. We compared MinHash with SubsMatch and Needleman-Wunsch. All three approaches are string-based methods and thus use the same input features (quantized scanpath from step ①). In addition, all three approaches calculate similarities or distances between strings, which makes it possible to use a uniform classification method (kNN from step ④). This ensures that the actual novelty (step ③) can be evaluated and ranked as isolated as possible to the existing approaches.

We evaluated our approach on two different datasets: The ETRA challenge dataset from 2019, and a self recorded Yarbus dataset with three different tasks. Furthermore the runtime behavior for different scanpaths lengths and number of subjects was investigated using synthetic data.

ETRA challenge 2019 The dataset from the ETRA challenge 2019 [McCamy et al. 2014; Otero-Millan et al. 2008] consists of 345 task related recordings. 115 each for free viewing a natural scene (15 different stimuli), image puzzles (18 different stimuli), and find waldo (15 different stimuli). The gaze signal was quantizied by a fixed 7×7 grid over the stimulus. Then, all sequences of consecutive repeating symbols shorter than 8 where deleted (8 consecutive symbols correspond to ~100ms @ 75Hz), since they were considered too short as a valid fixation. Sequences longer than or equal to 8 symbols were reduced by the factor of 8, but truncated to maximal 3 consecutive symbols. E.g.:

EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAATTTTTTTTTETERRRRRRRREEAAAAAAAAAAAAAAAATT

32×E→EEE 9×T→T 8×R→R 17×A→AA

Is reduced to: EEETRAA. This leads to an enormous reduction of the scanpaths. However, the average scanpath length is still 313 symbols (min. 173, max. 500).

MinHash has been configured with $w_{\min} = 2$, $w_{\max} = 3$, a valid gap $w_{\text{gap}} = 1$, and a base weight of $w_{\text{weight}} = 1.2$. The evaluation was conducted as a 345-fold cross validation using Matlabs `crossvalind` function to partition the training and test set. The classification was performed as described in section 3 step ④ using kNN ($k = 4$) in the Jaccard Index domain, using matlabs `fitcknn` function. The prediction target was the performed task of the subject (natural viewing, puzzle, where's Waldo) across all stimuli.

Figure 6 shows the results for different numbers of hash functions $n_h = \{64,128,256,512\}$, and for SubsMatch (Needleman-Wunsch was runtime conditionally not applicable). As the number of different hash functions increases, the accuracy of the classification increases until it saturates at about 88% at $n_h = 256$. SubsMatch achieves an accuracy of 91%. However, MinHash takes only the half of the processing time for $n_h = 256$, by almost same performance as subsmatch.

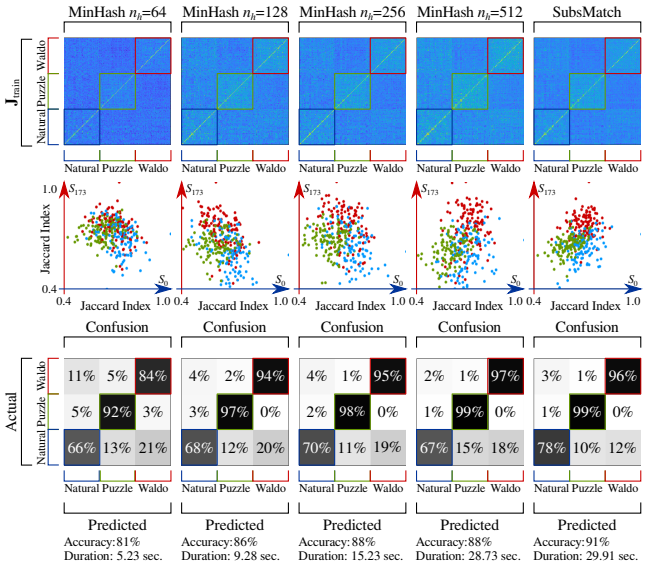

Figure 6: Each column represents a cross validation using the ETRA 2019 challenge dataset. The first 4 columns show the classification using MinHash across different numbers of hash functions. The last column shows the result of SubsMatch. The first column shows the pairwise calculated similarity of two scanpaths. Each row and column corresponds to one scanpath. For the classes Puzzle and Waldo, distinct clusters are recognizable. The second row shows the similarity values of the first row in the similarity domain of scanpath S_0 and S_{173}. Even when considering only 2 of 345 dimension, a clear cluster formation is recognizable with increasing number of hash functions. The last row shows classification performance of the cross validation as confusion matrix.

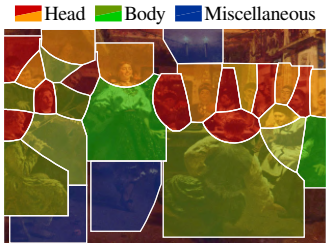

Figure 7: Grown semantic RoIs of heads, bodies, and various scene parts on one of the Defending Yarbus [Borji and Itti 2014] stimuli.

Yarbus This dataset is much more challenging in terms of classification performance compared to the ETRA2019 challenge dataset. The recording includes 23 subjects. 10 images from the Defending Yarbus [Borji and Itti 2014] were presented to each subject in randomly permuted order for 10 seconds. For each image, the subject was asked to solve one of three different tasks:

(1) Estimate ages of the people
(2) Remember clothes
(3) Estimate how long the visitor had been away

The scanpaths were created using semantic RoIs with a background grid of 3×3. Therefore heads, bodies, and other prominent image

areas were marked with rectangles. In order to compensate for noise and inaccuracies in the gaze signal, and to eliminate overlapping RoIs, we grow the RoIs until they encountered each other or reached a maximum size (see figure 7). The resulting scanpath was reduced similar to the ETRA 2019 challenge dataset, but with a reduction factor of 6 (60Hz eye tracker). This resulted in 10×23 scanpaths with an average length of 26 symbols (min. 5, max. 36).

Analogous to the ETRA 2019 Challenge, a cross validation with $w_{\min}=2$, $w_{\max}=3$, a valid gap $w_{\text{gap}}=1$, a base weight of $w_{\text{weight}}=1.2$, and kNN with $k=4$ was performed. Figure 8 shows the classification performance of MinHash with $n_h=128$, SubsMatch, and Needleman-Wunsch. The prediction target was the task performed per stimulus. It is noticeable that this challenge is much more difficult than the ETRA 2019 challenge. The best result was achieved by SubsMatch with an accuracy of 59% and a chance of 33%. However, this evaluation clearly shows the limits of the MinHash approach. For such short scanpaths and few subjects, MinHash is not able to play out its speed boost significantly: Even Needleman-Wunsch is faster. At the same time, due to its design, MinHash can never achieve better accuracy than SubsMatch.

Runtime MinHash has a complex runtime behavior. In theory, MinHash's runtime is $O(n)$, since it needs n operations to scan the input scanpaths and to build up the hash tables. However, since all considered algorithms have to read the input with a runtime of $O(n)$, we only examine the part of the actual similarity calculation below. In this case, the runtime of MinHash is constant for large n.

For the runtime evaluation we used a desktop PC with moderate hardware: i5-4590 @ 3.30GHz and 16GB RAM. All methods where tested using a single thread. The evaluation was done by random generated scanpaths, since it reflects the worst case scenario for the MinHash and SubsMatch algorithm. Therefore, a uniform distributed random sequence of symbols of a given alphabet and a fixed scanpath length was generated. Thus, each word has the same likelihood to be part of the scanpath, which leads to huge hash tables. For short but many scanpaths, MinHash suffers from many different words in the dictionary, but a low probability to find them in a single scanpath. Therefore, it iterates through the Dictionary until the word appears in one of the compared scanpaths (see algorithm 2 line 4). This event can be prevented by creating n_h dedicated hash tables for each scanpath pair. These hash tables contain only words that occur in at least one of the two scanpaths. Instead of iterating through D_{H_i}, the first element can be accessed directly. However, this approach is not technically feasible in most cases in terms of memory consumption. In addition, the overhead would remain linear during indexing, but still increases significantly.

Additionally, we can show that the actual number of iterations is comparatively low for large scanpaths. The probability that q iterations on the global dictionary D_{H_i} are necessary in order to obtain a word that is part of the scanpath dictionaries D_{S_k} or D_{S_l} can be described as a geometric distribution with an expected value of [Dekking et al. 2005]:

$$E[P(X=q)]=(1-p)\cdot p^{-1}, \tag{5}$$

where p is the probability to find a word that is part of one of the both scanpaths at a random position in D_{H_i}:

$$p=\left|D_{S_k}\cup D_{S_l}\right|\cdot\left|D_{H_i}\right|^{-1}, \quad \text{whereby} \quad D_{H_i}=\bigcup_{j=0}^{n} D_{S_j}, \tag{6}$$

Figure 8: Evaluation of MinHash, SubsMatch, and Needleman-Wunsch. The first column shows the 10 presented stimuli [Borji and Itti 2014]. For each stimulus, subjects completed one of three tasks: (Task 1) Estimate ages of the people, (Task 2) Remember clothes, (Task 3) Estimate how long the visitor had been away. The following three columns show the confusion matrix for the classification of the task performed by the subject.

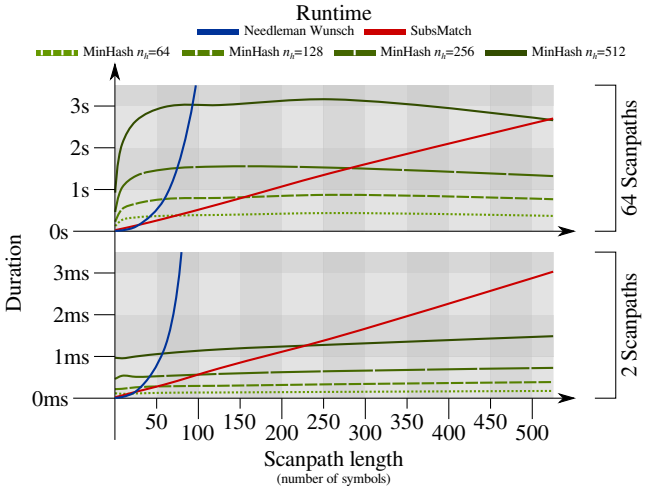

Figure 9: Shows the runtime behavior of the MinHash approach as well as of SubsMatch and Needleman-Wunsch with increasing scanpath length. Needleman-Wunsch is $O(n^3)$ (or $O(n^2)$ as Levenstein distance). Assuming that a scanpath is a random process, the runtime of SubsMatch and MinHash saturates for long scanpaths.

which resolves to:

$$E[P(X=q)] = \left|D_{H_i}\right| \cdot \left|D_{S_k} \cup D_{S_l}\right|^{-1} - 1. \tag{7}$$

Further, for long scanpaths applies:

$$\lim_{|S_k, S_l| \to \infty} \left|D_{S_k} \cup D_{S_l}\right| = \left|D_{H_i}\right|, \tag{8}$$

respectively:

$$\lim_{\left|D_{S_k} \cup D_{S_l}\right| \to \infty} E[P(X=q)] = 0, \tag{9}$$

under the assumption that a scanpath is a random process of symbols. Fun fact: p is the Jaccard Index of the union of the Dictionaries D_{S_k} and D_{S_l} to the global dictionary D_{H_i}.

Figure 9 shows the runtime of the Needleman-Wunsch, SubsMatch, and MinHash algorithm for different scanpath lengths. For two scanpaths, the duration of MinHash is almost constant for different n_h but increasing scanpath length $|S|$. However the runtime behavior at 64 scanpaths shows an interesting effect. First, the runtime increases for larger scanpaths. This is expected, since the number of words in the dictionary grows faster than the likelihood of a word to be in the scanpath. However, after a certain point the runtime decreases, and saturates on a certain level. This is a remarkable behavior for an algorithm, and is attributed to two competing effects: First, the dictionary size saturates at the level where all possible words are indexed. At this point, the negative impact of larger scanpaths also saturates. Second, the likelihood of a word to be part of a scanpath increases, and thus, less iterations for each hash permutations are necessary to find a word that is part of at least one of the compared scanpaths. Figure 10 shows this effect more emphasized. Comparing more than 32 scanpaths, MinHash is slower for short scanpaths (256 symbols) than for long scanpaths (512 symbols).

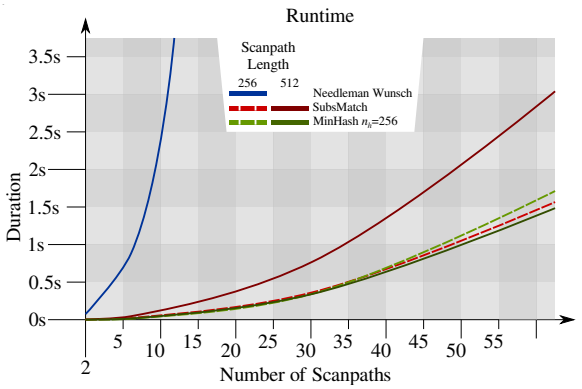

Figure 10: Shows the influence of the increasing number of scan paths for a fixed scanpath length.

The first effect also applies to SubsMatch. Yet, SubsMatch does not use a global dictionary, but uses the dictionaries of the respective scanpaths: $D_{H_0} = D_{S_k} \cup D_{S_l}$. With increasing scanpath size, the probability increases that D_{H_0} contains all words, and the runtime saturates.

5 LIMITATIONS

The evaluation on the Yarbus dataset shows that MinHash is not suited for every dataset. Only if the scanpath length in relation to the number of scanpaths is large enough, MinHash becomes really fast compared to SubsMatch (and to Needleman-Wunsch anyway). It should also be noted again, that MinHash approximates the Jaccard Index by testing n_h words, and therefore cannot achieve the classification performance of SubsMatch. SubsMatch calculates the Jaccard Index completely over all extracted words.

In addition, despite the global dictionary, the memory requirements for many words can become very extensive. For the calculation of the similarity matrix $\mathbf{J}_{\text{test}}$ in the ETRA 2019 challenge evaluation, ~170MB of the working memory is required with $n_h = 256$, $w_{\min} = 2$ and $w_{\max} = 3$. For the calculation with $w_{\max} = 4$, there are already more than 500MB required.

6 FINAL REMARKS

We have presented a method based on MinHash that efficiently estimates the Jaccard Index in a quantizied scanpath with large length and word variety. On the Etra 2019 challenge dataset, we were able to almost achieve the classification performance of subsmatch at a significantly lower runtime. The evaluation also showed that this runtime advantage increases with increasing input size. Thus, MinHash is particularly suitable for data mining in large quantities of long scanpaths. This provides the community a tool that allows to encode extensive information when quantizing scanpaths without the exploding alphabet and scanpath size affecting the classification runtime. At the same time, the Jaccard Index approximation of MinHash is mathematically reasoned and cleanly proven, which supports the validity of derived propositions.

Our C++ implementation of MinHash, SubsMatch, and Needleman-Wunsch is freely available (including Matlab interface) on our public GIT repository. Linux users can also use the ready to run binaries:
https://atreus.informatik.uni-tuebingen.de/geislerd/pe-tools-scanpath

REFERENCES

Nicola C Anderson, Fraser Anderson, Alan Kingstone, and Walter F Bischof. 2015. A comparison of scanpath comparison methods. *Behavior research methods* 47, 4 (2015), 1377–1392.

Christoph Berger, Martin Winkels, Alexander Lischke, and Jacqueline Höppner. 2012. GazeAlyze: a MATLAB toolbox for the analysis of eye movement data. *Behavior research methods* 44, 2 (2012), 404–419.

Alexandre Bissoli, Daniel Lavino-Junior, Mariana Sime, Lucas Encarnação, and Teodiano Bastos-Filho. 2019. A Human–Machine Interface Based on Eye Tracking for Controlling and Monitoring a Smart Home Using the Internet of Things. *Sensors* 19, 4 (2019), 859.

Ali Borji and Laurent Itti. 2014. Defending Yarbus: Eye movements reveal observers' task. *Journal of vision* 14, 3 (2014), 29–29.

Christian Braunagel, David Geisler, Wolfgang Rosenstiel, and Enkelejda Kasneci. 2017. Online recognition of driver-activity based on visual scanpath classification. *IEEE Intelligent Transportation Systems Magazine* 9, 4 (2017), 23–36.

Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. 2015. Eye movements in code reading: Relaxing the linear order. In *Program Comprehension (ICPC), 2015 IEEE 23rd International Conference on*. IEEE, 255–265.

Nora Castner, Enkelejda Kasneci, Thomas Kübler, Katharina Scheiter, Juliane Richter, Thérése Eder, Fabian Hüttig, and Constanze Keutel. 2018. Scanpath comparison in medical image reading skills of dental students: distinguishing stages of expertise development. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. ACM, 39.

Filipe Cristino, Sebastiaan Mathôt, Jan Theeuwes, and Iain D Gilchrist. 2010. ScanMatch: A novel method for comparing fixation sequences. *Behavior research methods* 42, 3 (2010), 692–700.

Rong-Fuh Day. 2010. Examining the validity of the Needleman–Wunsch algorithm in identifying decision strategy with eye-movement data. *Decision Support Systems* 49, 4 (2010), 396–403.

Fabian Deitelhoff, Andreas Harrer, and Andrea Kienle. 2019. The influence of different AOI models in source code comprehension analysis. In *Proceedings of the 6th International Workshop on Eye Movements in Programming*. IEEE Press, 10–17.

Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. 2005. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media.

Richard Dewhurst, Marcus Nyström, Halszka Jarodzka, Tom Foulsham, Roger Johansson, and Kenneth Holmqvist. 2012. It depends on how you look at it: Scanpath comparison in multiple dimensions with MultiMatch, a vector-based approach. *Behavior research methods* 44, 4 (2012), 1079–1100.

Shahram Eivazi, Ahmad Hafez, Wolfgang Fuhl, Hoorieh Afkari, Enkelejda Kasneci, Martin Lehecka, and Roman Bednarik. 2017. Optimal eye movement strategies: a comparison of neurosurgeons gaze patterns when using a surgical microscope. *Acta neurochirurgica* 159, 6 (2017), 959–966.

Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. 2015. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 457–466.

Rebecca M Foerster and Werner X Schneider. 2014. Functionally sequenced scanpath similarity method (FuncSim): Comparing and evaluating scanpath similarity based on a task's inherent sequence of functional (action) units. (2014).

Tom Foulsham, Richard Dewhurst, Marcus Nyström, Halszka Jarodzka, Roger Johansson, Geoffrey Underwood, and Kenneth Holmqvist. 2012. Comparing scanpaths during scene encoding and recognition: A multi-dimensional approach. (2012).

Wolfgang Fuhl, Thomas C Kübler, Hanna Brinkmann, Raphael Rosenberg, Wolfgang Rosenstiel, and Enkelejda Kasneci. 2018a. Region of interest generation algorithms for eye tracking data.. In *ETVIS@ ETRA*. 10–1.

Wolfgang Fuhl, Thomas C Kübler, Thiago Santini, and Enkelejda Kasneci. 2018b. Automatic Generation of Saliency-based Areas of Interest for the Visualization and Analysis of Eye-tracking Data.. In *VMV*. 47–54.

Ziba Gandomkar, Kevin Tay, Patrick C Brennan, and Claudia Mello-Thoms. 2018. Recurrence quantification analysis of radiologists' scanpaths when interpreting mammograms. *Medical physics* 45, 7 (2018), 3052–3062.

John Heminghous and Andrew T Duchowski. 2006. iComp: a tool for scanpath visualization and comparison.. In *SIGGRAPH Research Posters*. 186.

Sabrina Hoppe, Tobias Loetscher, Stephanie A Morey, and Andreas Bulling. 2018. Eye movements during everyday behavior predict personality traits. *Frontiers in human neuroscience* 12 (2018), 105.

Halszka Jarodzka, Kenneth Holmqvist, and Marcus Nyström. 2010. A vector-based, multidimensional scanpath similarity measure. In *Proceedings of the 2010 symposium on eye-tracking research & applications*. ACM, 211–218.

Thomas Kübler. 2016. *Algorithms for the Comparison of Visual Scan Patterns*. Ph.D. Dissertation. Eberhard Karls Universität Tübingen.

Thomas Kübler, Shahram Eivazi, and Enkelejda Kasneci. 2015. Automated visual scanpath analysis reveals the expertise level of micro-neurosurgeons. In *MICCAI workshop on interventional microscopy*. 1–8.

Thomas Kübler, Wolfgang Fuhl, Raphael Rosenberg, Wolfgang Rosenstiel, and Enkelejda Kasneci. 2016. Novel methods for analysis and visualization of saccade trajectories. In *European Conference on Computer Vision*. Springer, 783–797.

Thomas C Kübler and Enkelejda Kasneci. 2015. Automated Comparison of Scanpaths in Dynamic Scenes. In *SAGA-International Workshop on Solutions for Automatic Gaze Data Analysis: Proceedings*.

Thomas C Kübler, Enkelejda Kasneci, and Wolfgang Rosenstiel. 2014. Subsmatch: Scanpath similarity in dynamic scenes based on subsequence frequencies. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 319–322.

Thomas C Kübler, Colleen Rothe, Ulrich Schiefer, Wolfgang Rosenstiel, and Enkelejda Kasneci. 2017. SubsMatch 2.0: Scanpath comparison and classification based on subsequence frequencies. *Behavior research methods* 49, 3 (2017), 1048–1064.

Charles Eric Leiserson, Ronald L Rivest, Thomas H Cormen, and Clifford Stein. 2001. *Introduction to algorithms*. Vol. 6. MIT press Cambridge, MA.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.

Mehdi Ebady Manaa and Ghufran Abdulameer. 2018. Web Documents Similarity using K-Shingle tokens and MinHash technique. *J. Eng. Appl. Sci* 13 (2018), 1499–1505.

Sebastiaan Mathôt, Filipe Cristino, Iain D Gilchrist, and Jan Theeuwes. 2012. A simple way to estimate similarity between pairs of eye movement sequences. (2012).

Michael B McCamy, Jorge Otero-Millan, Leandro Luigi Di Stasi, Stephen L Macknik, and Susana Martinez-Conde. 2014. Highly informative natural scene regions increase microsaccade production during visual scanning. *Journal of neuroscience* 34, 8 (2014), 2956–2966.

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* 48, 3 (1970), 443–453.

Jorge Otero-Millan, Xoana G Troncoso, Stephen L Macknik, Ignacio Serrano-Pedraza, and Susana Martinez-Conde. 2008. Saccades and microsaccades during visual fixation, exploration, and search: foundations for a common saccadic generator. *Journal of vision* 8, 14 (2008), 21–21.

Michael Raschke, Dominik Herr, Tanja Blascheck, Thomas Ertl, Michael Burch, Sven Willmann, and Michael Schrauf. 2014. A visual approach for scan path comparison. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 135–142.

Amit Singhal et al. 2001. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.* 24, 4 (2001), 35–43.

A Van der Gijp, CJ Ravesloot, H Jarodzka, MF Van der Schaaf, IC Van der Schaaf, Jan PJ van Schaik, and Th J Ten Cate. 2017. How visual search relates to visual diagnostic performance: a narrative systematic review of eye-tracking research in radiology. *Advances in Health Sciences Education* 22, 3 (2017), 765–787.