# A Robust Realtime Reading-Skimming Classifier

Ralf Biedert*
Jörn Hees†
Andreas Dengel‡
German Research Center
for Artificial Intelligence

Georg Buscher§
Microsoft Bing

## Abstract

Distinguishing whether eye tracking data reflects reading or skimming already proved to be of high analytical value. But with a potentially more widespread usage of eye tracking systems at home, in the office or *on the road* the amount of environmental and experimental control tends to decrease. This in turn leads to an increase in eye tracking noise and inaccuracies which are difficult to address with current reading detection algorithms. In this paper we propose a method for constructing and training a classifier that is able to robustly distinguish reading from skimming patterns. It operates in real time, considering a window of saccades and computing features such as the *average forward speed* and *angularity*. The algorithm inherently deals with distorted eye tracking data and provides a robust, linear classification into the two classes *read* and *skimmed*. It facilitates reaction times of 750ms on average, is adjustable in its horizontal sensitivity and provides confidence values for its classification results; it is also straightforward to implement. Trained on a set of six users and evaluated on an independent test set of six different users it achieved a 86% classification accuracy and it outperformed two other methods.

**CR Categories:** I.5.4 [Computing Methodologies]: PATTERN RECOGNITION—Applications;

**Keywords:** eye tracking, reading, skimming, machine learning

## 1 Introduction

The preferred way to detect reading behavior, i.e., deciding whether observed eye movement patterns are compatible with patterns commonly defined as *reading*, is to measure fixation progress on words expressed in character units [Hyrskykari 2006], [Buscher et al. 2008]. This approach is thoroughly studied [Rayner 1998a] and works well in cases where the eye tracking accuracy is high enough to provide word level resolution.

However, many eye tracking systems and situations are unable to provide such a high resolution, especially due to stochastic noise, measurement drift, miscalibrations, and all of these in combination with small fonts and narrow line spacing. Figure 1 depicts several of these issues. Because of this noise, even though a text has been read, the individually measured gaze points can miss most words and the overall pattern usually *jumps* diagonally across multiple lines, rather than following a single line continuously.

*ralf.biedert@dfki.de
†joern.hees@dfki.de
‡andreas.dengel@dfki.de
§georgbu@microsoft.com


ETRA 2012, Santa Barbara, CA, March 28 – 30, 2012.
© 2012 ACM 978-1-4503-1225-7/12/0003 $10.00

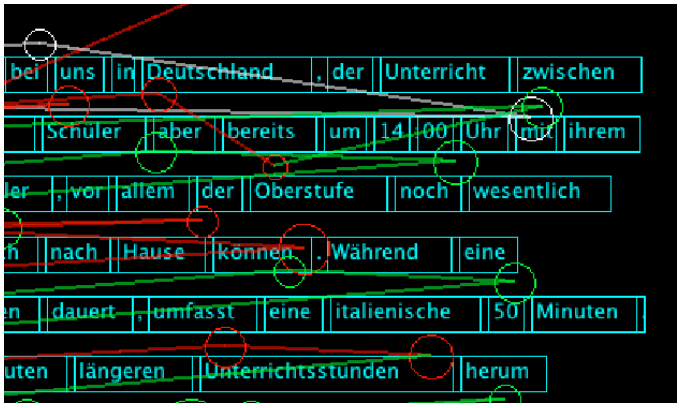


**Figure 1:** *Sample fixation data displaying problems commonly encountered when dealing with eye tracking data on text. Most of the fixations are off and it is unclear to which line of the text they apply to. Also, most saccades are not strictly horizontal but rather jump diagonally across the text.*

In psychological studies the experiment settings can be controlled sufficiently, either through the use of head fixation, the manual realignment of gaze data with the help of control points, or by presenting just a single line of text at a time. However, we believe the principal issue of noisy eye tracking data will be of special concern in mostly uncontrolled interactive scenarios. There, usually normal-sized texts are read with a partially degraded calibration, and no manual correction can be performed. We anticipate that especially future handheld eye tracking systems, similar to the recently developed C12[1] unit, will be affected by these issues. This device class suffers from three additional degrees of freedom (device rotation) in comparison to desktop mounted devices, and when being hold will be more prone to tilts, shifts and shaking.

But even in such situations where the point of gaze cannot be determined exactly, it can be desirable to automatically decide to what extent eye movements resemble a reading pattern or a skimming pattern in order to automatically respond to this behavior. Examples applications include ScentHighlight [Chi et al. 2005], which highlights related sentences during reading; the eyeBook [Biedert et al. 2010a], where ambient effects are to be triggered in proximity of the reading position; or QuickSkim [Biedert et al. 2010c], where non-content words may be faded out in real time with an increase of skimming speed to make reading more efficient. Also, in the domain of information retrieval it has been shown that acquiring implicit feedback from a reading and skimming detection can significantly improve search accuracy through personalization [Buscher 2010].

In this respect it is now an interesting observation that human judges who inspect noisy eye tracking scanpaths are often able to classify what segments of the scanpath belong to reading or skimming behavior, even though the fixations do not match the underlying text[2] (e.g., consider the scanpath in Figure 2). More specifically one can

[1] See Tobii `http://tobii.com`
[2] Although the knowledge *that* the scan path was related to text—while

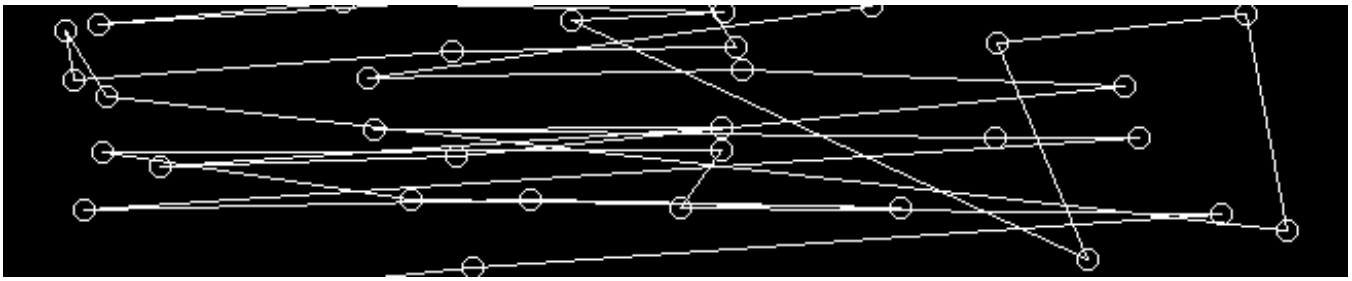

**Figure 2:** *Fixation data with the underlying surface removed. Even though the text is not visible to a human judge it is possible to estimate what parts of it point to reading behavior. The overall pattern clearly indicates reading-like behavior, the line height is visible, and since most fonts have a certain aspect ratio also the character width can be estimated, which in turn gives an estimate if the pattern rather indicates reading or skimming.*

notice that a classification of eye tracking data can usually be done already when only some limited context of the saccade in question is provided.

Based on this observation we propose a reading detector that learns to recognize reading, similar to human experts, through an analysis mostly based on scanpaths. This should enable it to perform a classification even under unfavorable conditions. Since some of our use cases also require a real time response, with reaction times less than one second, the algorithm should also be able to classify incoming gaze data with as little forward-context (i.e, fixations and which happened after the saccade being considered) as possible. Furthermore, it should be reasonably simple in terms of parameters and their meaning and be adjustable in how permissive it is with respect to various reading speeds. In its very principal structure the algorithm we propose is somewhat similar to the real time reading detection described by [Campbell and Maglio 2001], with a number of important differences. Most notably it does not distinguish icon search from reading, but is rather a method capable of distinguishing intensive, careful textual perusal (*reading for meaning*, compare [Reichle et al. 1998]) from less intensive, rather searching textual perusal (*skimming*, ditto).

However, it should be kept in mind that it is not our intention to construct an algorithm that emits the labels *reading* or *skimming* in terms of a ground truth on its own. It rather rates how closely its input resembles what it previously learned about these classes. Their actual definition might depend on a specific context, such as *book reading in a Germanic language*.

The rest of the paper is now structured as follows. We start with presenting an explorative study for generating a ground truth gaze data set for reading and skimming behavior. Using the training set we construct a classifier to detect and differentiate reading from skimming, based on features described in section *Features & Training*. In the *Evaluation* section we describe our results on these features and classifiers and present a linear model to discriminate both behaviors. This is followed by our *Conclusion* and an *Outlook* onto future work.

## 2 Explorative Study

In order to construct a classifier that can learn to distinguish the classes *reading* and *skimming* we need to collect a data set containing both. Thus, we start with an explorative study in which we record eye movements for a number of persons that are asked to read various texts in different *modes*. These recording constitute our training and evaluation dataset.

For the study participants are asked to assume the role of a news paper editor who has to read various types of articles under time pressure in order to answer questions at the end of each trial. A single experiment consists of four trials and each trial started with the presentation of a fictional mail from their main editor asking them to write about the given topic. Each mail also contains two to three attached documents on which their report for the trial should be based upon. In order to elicit *reading* and *skimming* behavior, some passages presented in each trail contain relevant text which has to be read intensively to successfully complete the task. Others contain irrelevant information which must be skimmed or skipped to finish in time. On average each document contains one paragraph that is related to the topic while the remaining ones are unrelated.

For our actual study we invited 12 participants to read each of the texts, their average age was 24.6 years (ranging from 20 to 31), 11 of them male, 1 female and most of them were students at the local university. Since all texts are written in German we also ensured the participant's primary language was German. Each user was instructed verbally on the tasks and was given an initial training trial which was discarded for the subsequent evaluation. The experiment took place in a web browser with a font size of 16pt, the interaction was recorded with a Tobii 1750 tracker and processed with the Text 2.0 Framework [Biedert et al. 2010b].

Overall the experiment yielded $12 * 8 = 96$ read document passes, and for our ground truth the recorded eye tracking data has to be manually classified into the main categories *read* and *skimmed*. With *reading* we subsume line-wise gaze behavior with real or assumed average saccade distances below approximately 10 characters (real when the fixations clearly matched text, assumed when it did not match the text but the distances could be extrapolated from words nearby; newline regressions within a read context also count as reading). *Skimming* subsumes all other gaze behavior that occurs on text, such as line-wise perusal with offsets larger than 10 characters and obvious vertical movements. Our distinction between reading and skimming is mostly influenced by the approximate size of the word identification span (compare, for example [Underwood 1985] and more generally [Rayner 1998b]), and we assume that a number of saccades larger than this span are an indicator that the texts *covered* by these saccades are most likely not read for meaning.

To label the data we next created a rating application. For each round it presents the visual representation of a randomly selected window of eight subsequent fixations and their saccades, including the text, compare Figure 3. The saccade in the middle is called the *saccade under consideration*, while each of the three surrounding saccades are called their *past-* and *future context*, respectively. The saccade in the middle is highlighted, and the experimenter is asked

not strictly necessary—will help to reduce false positives. However, the scenario we mainly target is when due to drift and errors the measured gaze position cannot be reliably matched to the text. While the system usually has knowledge that the user is considering text (and not icons or images) it cannot reliably determine character progress.

to give a judgement to which class the marked saccades and its context belongs.

In our study, for each of the read documents approximately 15 saccades under consideration are randomly selected. This results in a total of 1407 saccades for the training and testing, which are eventually presented to two human experts for labeling. Both experts have more than one year of eye tracking experience and rated the set independently. In the end both results are merged, and only matching ratings and considered as valid, while differing ratings were not considered for training. The agreement rate for both experts was 74.9%, resulting in 1055 saccades to which both experts gave the same class of either reading or skimming. From the remaining 352 saccades, 1.8% (25) were labeled as unknown[3] by both experts, 8% (113) were labeled as unknown by at least one expert, the remaining 15.2% (214) had truly differing ratings in terms of that they were labeled as *skimming* by one expert and *reading* by the other. Overall 40.6% (428) saccades had an agreed rating of reading, 59.4% (627) an agreed rating of skimming. Ignoring saccades that were unknown to one user, both reviewers achieved an agreement rate of 83.1% ($\kappa = 0.65$). A more detailed look at the differing saccades revealed that while in some instances the given classification was most probably mistaken by one expert, many of them were borderline cases where, depending on how the context has been interpreted, both ratings appeared to be justifiable[4]. The saccades that were labeled as *unknown* by only one expert were in most cases (85%, 96 instances) rated as skimming by the other.

We now extract two subsets from this set of rated saccades . The saccades of six randomly selected users become part of the *training set*, the saccades of the six remaining users become part of the independent test set, forming two almost equally sized sets that allows us to investigate how a trained classifier is likely to perform on unknown users, i.e., generalize. With the training set we next continued to construct a set of feature vectors and train a classifier learning on them.

## 3 Features & Training

To build a function capable of distinguishing reading from skimming when given raw eye tracking in screen coordinates, some preprocessing is required to transform the incoming measurements into a more suitable representation. Given the eye tracker emits a set of raw eye tracking points $E' = e'_1, ..., e'_{cur}$ we first filter and denoise the data using a *virtual median* filter $E = vm(E')$, so that

$$e_i = (median_x(e'_{i-2}, ..., e'_{i+2}),\ median_y(e'_{i-2}, ..., e'_{i+2}))$$

which allows us to robustly rule out measurement outliers without affecting the filtered position. To the filtered data we then apply a dispersion window based fixation detection algorithm $fix_{dis}$

$$F' = f'_1, ...f'_n = fix_{dis}(E)$$

with $100ms$ temporal and $25px$ spacial dimensioning, which transforms them into screen fixations. Next we transform each of these screen fixations $f'$ into document fixations $f$ by converting them into the document coordinate system based on the document's window geometry and viewport at the time of each fixation:

$$F = to_{doc}(F')$$

[3]The raters agreed that the label *unknown* should only be given when the observed pattern was unusual and likely neither reading nor skimming, or could not clearly be identified in the evaluation tool.

[4]For example, similar to Figure 3, past contexts of skimming that were just about to enter reading-for-meaning, or skim-alike-outliers in an otherwise reading-like stream.

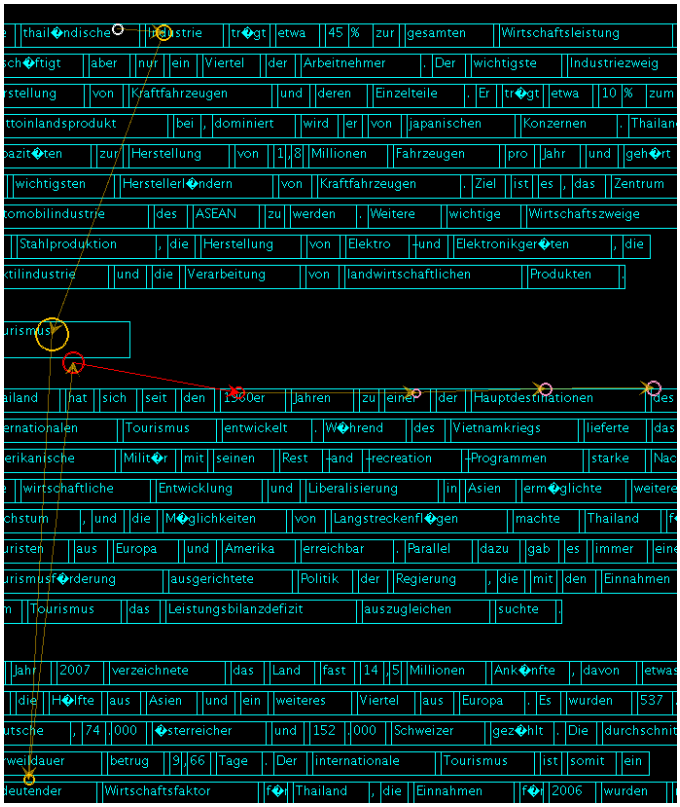

**Figure 3:** *One of the 1400 slices of fixations that were rated independently by two eye tracking experts. The red saccade (approximately horizontal in the middle of the screen) had the focus while the saccades before and after were the contributing context. The given example was rated as* skimming *by both experts, since it was at the end of a vertical skimming pattern, just about to enter fast reading.*

It should be noted that in terms of mapped fixations in real world scenarios the stream $F$ usually contains a number of empty fixations $f_\emptyset$ , since not every fixation observed on the screen actually hit the document and could therefore be converted, effectively representing outliers in the stream of observed document fixations.

The stream of mapped fixations now serves as the basic input on which we build our classifiers and evaluate feature vectors. As we are interested in a real time classification we also define a window function

$$w_{a,b}(F) := f_a, ..., f_b \in F$$

which returns a consecutive slice of fixations. Due to the real time demand the effective window size to consider is usually relatively small and we define $w_{a,b}$ to return the empty set $\emptyset$ when one of the converted elements was outside the document area, i.e., $f_\emptyset \in F$, effectively signaling that the user was probably visually distracted and no further classification should be done for the window. Hence from here on we can assume for simplicity that each extracted window contains the same defined amount of elements.

Now for the purpose of building feature vectors for machine learning, the outputs of function $w_{a,b}$ are the principal elements we consider. In training and evaluation they are constructed around a rated fixation extracted from the gaze stream such that $a = x - \delta$ and $b = x + \delta$, where $x$ is some position in the stream, and $2\delta$ the mentioned window size comprising $\delta$ saccade elements of past and $\delta - 1$ elements future context. During a realtime classification task $x$ would naturally represent the $(\delta+1)$-latest observed fixation, and

$\delta$ would be selected as the smallest value that still gives acceptable results in terms of overall accuracy. In a non-realtime scenario the value should be selected so that it simply maximizes the accuracy. For example, the samples used in training, like depicted in Figure 3, correspond to $w_{x-\delta,x+\delta}$ slices with $\delta = 4$.

From here on we explore three main variants to construct feature vectors. The first variant, baseline$_1$, is a traditional reading detection based on actual character offsets. It maps the observed fixations to words and measure the progress in the document expressed in character units. The second variant, baseline$_2$, is some sort of a naïve implementation of the idea not to express saccades in real character progress, but rather define and detect reading in terms of relative saccade patterns. These first two variants serve as our baselines against which our actual feature generator will be compared. Our main feature generator, the *contextual saccade shape*, uses the relative saccade patterns, but tries to interpret implicitly encoded semantical information to simplify the actual classification task. Its main idea is to express the *shape* of $w$ with as few but as expressive parameters as possible.

We consider baseline$_1$ to approximately reflect the state of the art of character-based classification methods [Buscher et al. 2008] [Hyrskykari 2006] in *quasi*-realtime scenarios, in which no manual correction of gaze data is or can be performed. In addition baseline$_2$ serves as a comparison for the contactual saccade shape algorithm to test if the transformation of gaze data into a more abstract format yields any additional benefit. Very coarsely it mimics some of the vector based approaches reported earlier [Campbell and Maglio 2001] [Kollmorgen and Holmqvist 2007] [Holmqvist et al. 2003] [Holmqvist et al. 2011][5].

Technically all three variants accept an emitted window $w$ and return a feature vector $v = \nu_1, ..., \nu_m$, which usually consists of one or more numeric values $\nu_i$. These values, along with a label, can then be given to classifiers such as SVMs.

### 3.1 Character Based Reading Detection

As mentioned reading behavior is traditionally described by the average amount of character positions the reader advances with a saccade, measured sequentially from the start of the given block of text. Thus for our baseline$_1$ we implemented a reading detection algorithm similar to [Buscher et al. 2008]. Given the set of fixations in $w$, each one is *magnetically* mapped [Hyrskykari 2006] to the closest word in the document and its character position with respect to the whole text is extracted. An actual feature vector $v_{b1}$ is therefore composed of the character offsets $o_1, ..., o_{m-1}$ in the document

$$v_{b1} = o_1, ..., o_{m-1}$$

between the fixations $f_1, ..., f_m$ of the given window. It should be noted that in contrast to the original algorithm, due to the high amount of vertical saccades in our scenarios, the employed magnetic match had to be considered on a word level basis and not on a line level. Thus in a setting where skimming mostly consists of horizontal saccades and realtime is of less concern a line based magnetic or sticky approach (again, compare [Hyrskykari 2006]) will probably perform better than baseline$_1$.

### 3.2 Normalized Saccade Vectors

Since it is our intention to perform a classification mostly based on the gaze data alone and with as little *surface knowledge* as necessary we implemented what could be considered as a naïve approach for baseline$_2$. In it we use the fixation pattern almost directly, only applying a conversion into a relative representation.

Comparing baseline$_1$ against baseline$_2$ serves both the purpose of verifying if not a relative conversion alone might already be sufficient and if further processing the raw data into a more compact format was justified, since such a *compression* could either remove data necessary for a proper classification, or it could denoise the data and improve accuracy.

For the creation of the feature vector in baseline$_2$ we first transform the sequence of fixations within $w$ into a sequence of saccades $S = s_1, ..., s_{m-1}$. In $S$, each saccade is represented in polar coordinates, $s_i = \theta_i, r_i$, denoting the angle and normalized length in *virtual character units* from $f_i$ to $f_{i+1}$.

A virtual character unit, in turn, specifies how many pixels width a single character of the underlying text approximately requires. It is computed by acquiring all word boxes that were covered by the saccades in $w$ in the document area, and the sum of their widths in pixels is divided by the sum of their lengths in character units. For the computation of $r_i$ we therefore divide the length of pixels of $s_i$ by the amount of character units.

An actual feature vector $v_{b2}$ thus consists of the individual angles $\theta$ and normalized lengths $r$.

$$v_{b2} = \theta_1, r_1, ..., \theta_n, r_n$$

While this feature vector serves as the baseline$_2$, it also serves as the foundation for the computation of our shape based representation.

### 3.3 Contextual Saccade Shape

After working with slices of fixations, their visualizations, and comparing the data labeled as *read* with the data labeled as *skimmed* it appears there are two major factors that contribute to this distinction. One can best be described as the average *angularity* of the saccade and its context, the other is its average forward speed, also compare Figure 4. Hence we decided to use these two attributes as the main features for our classifier.

Angularity $h$ in this respect denotes how *bent* or vertical the window's saccades, when considering their concatenated vector shape, on average appear in contrast to a single horizontal line. Obviously a number of subsequent fixations within one line of text should ideally deliver almost no angularity, while vertical perusal of text should yield a high value. For a given window $w$ we compute it as

$$h = atan2(\sum_{s \in w} \begin{pmatrix} |\vec{s_x}| \\ |s_y| \end{pmatrix})$$

where $s$ are all normalized saccades generated from the window as described in the previous section. This means a high horizontality will be expressed in values close to 0, while a mostly vertical pattern yields values up to $\frac{\pi}{2}$.

The average forward speed denotes how much progress in the reading direction of the text was made within the given window. It reflects how many characters have approximately been read on average during saccades compatible with reading, where the compatibility is estimated by angular bounds.

$$p = \varnothing\{r_i \in w : |\theta_i| < \frac{\pi}{3}\}$$

In other words the speed $p$ is computed as the average length of all saccades which point approximately towards the right. The bounds for this check are deliberately set to the large limits of $\pm\frac{\pi}{3}$ since the

[5] Also see $v_{ld*}$ in Section 4.1 and the detailed discussion in Section 5 on comparability.

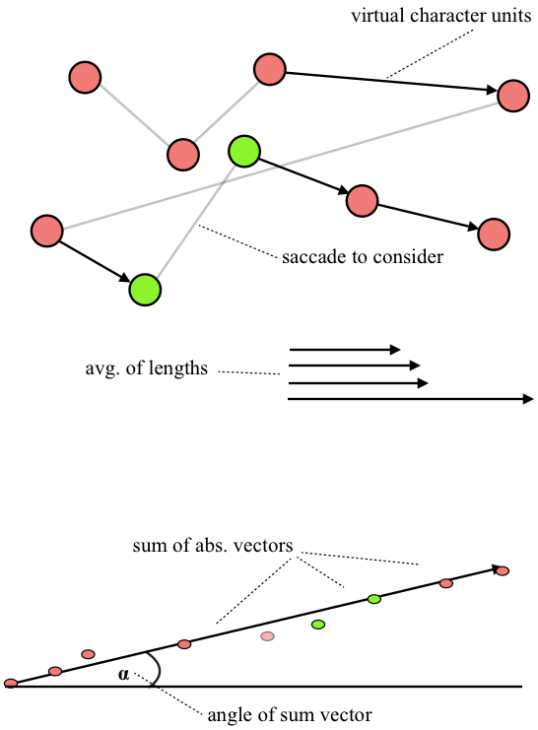


**Figure 4:** *Graphical outline of the generation of the $v_{css}$ feature vectors. For a given saccade we consider a window around it and compute the average lengths, expressed in virtual character units, of all saccades that have an approximate angle aligned with the direction of the text and the angle of the sum of all absolute vectors. Both values form the basis for classification.*

angularity feature should already sufficiently represent and handle the vertical and diagonal skimming patterns. Now our final set of feature vectors for the contextual saccade shape can be expressed as

$$v_{css} = h, p$$

With this set of features we next perform an evaluation of the algorithms' performances.

## 4 Evaluation

As mentioned the recorded and labeled data set was split into a training set including six users, and an independent testing set, including six other users. Even though generated from different users, both sets had similar distributions of reading to skimming patterns. The training set consists of 56% skimming data, while the testing set contains 51%. For the actual evaluation the feature vectors are generated on both sets. While the training set is used to find the best classifier and parameters the testing set is left untouched until optimal parameters are found and only eventually used to verify the findings.

### 4.1 Overall Performance Measures

There are two main tasks we need the training set for. The first is to find the best possible classifiers for the given feature vector representation, with the best variants we then perform an analysis of the window size $\delta$. While on the feature level we already selected a number of variants, for the classifier we put our focus on a

| | | | | Read | | Skim | | |
|---|---|---|---|---|---|---|---|---|
| | Vector | C | $\gamma$ | P | R | P | R | Acc. |
| $\delta = 1$ | $v_{b1}$ | .12 | $10^{-4}$ | .63 | .74 | .79 | .69 | .71 |
| | $v_{b2}$ | 8192 | $10^{-4}$ | .71 | .66 | .77 | .81 | .75 |
| | $v_{css}$ | 8192 | 5.02 | .76 | .82 | .86 | .82 | .82 |
| $\delta = 2$ | $v_{b1}$ | .58 | $10^{-4}$ | .78 | .83 | .86 | .82 | .77 |
| | $v_{b2}$ | .58 | .0025 | .73 | .69 | .78 | .82 | .76 |
| | $v_{css}$ | 339.4 | .06 | .85 | .82 | .88 | .89 | .86 |
| $\delta = 3$ | $v_{b1}$ | 2.86 | $10^{-4}$ | .75 | .79 | .84 | .81 | .8 |
| | $v_{b2}$ | 339.4 | .012 | .8 | .66 | .78 | .88 | .79 |
| | $v_{css}$ | 339.4 | .012 | .88 | .82 | .88 | .92 | .88 |
| $\delta = 4$ | $v_{b1}$ | .58 | $10^{-4}$ | .82 | .64 | .77 | .89 | .79 |
| | $v_{b2}$ | .58 | .0025 | .83 | .6 | .75 | .91 | .77 |
| | $v_{css}$ | 8192 | $10^{-4}$ | .86 | .78 | .84 | .9 | .85 |
| testing $\delta = 3$ | $v_{b1}$ | – | – | .84 | .82 | .86 | .88 | .86 |
| | $v_{b2}$ | – | – | .75 | .49 | .69 | .87 | .70 |
| | $v_{css}$ | – | – | .86 | .86 | .89 | .89 | .88 |
| rel | $v_{ld3}$ | 1.63 | 4 | .68 | .66 | .75 | .77 | .72 |
| | $v_{ld4}$ | 1.37 | 3.07 | .69 | .74 | .79 | .74 | .74 |

**Table 1:** *The upper part contains the results on the training set when testing for window sizes ($\delta$) and features with tenfold cross-validation. The best values for C and $\gamma$ parameters were determined by a grid search, precision (P) and recall (R) values are given for the classes reading and skimming, as well as the overall accuracy. The middle part contains the results of the best trained classifiers on the testing set, i.e., the results a classifier trained with one set of users achieved when exposed to a new set of users. In the lower part we show the results for two feature vectors (consisting of saccade length and fixation duration) similar to those described in related work.*

RBF-SVM for which we conducted a grid search over $C$ and $\gamma$; a method fairly established in machine learning. Also, since the average reaction time of the final classifier should be below 1000ms on average, this threshold determines the upper bounds for the window size to consider. Assuming an average fixation time of 250ms, $\delta$ was therefore limited to 4 in this study[6].

Using the labeled training we first do an initial comparison of the methods $v_{b1}$, $v_{b2}$ and $v_{css}$. The upper part of Table 1 lists the results of the classification runs on the training data as well as the best found parameters for the RBF-SVM, the precision and recall values for the *reading* and *skimming* class as well as the overall accuracy. It can be seen that in all cases the accuracy delivered by the $v_{css}$ feature set is superior to the character based feature vector baseline$_1$ and the raw feature set baseline$_2$, the same holds for the precision and recall values of both classes.

We also investigate the influence of the window size on the overall classification accuracy, compare Figure 5. For all three investigated feature types the accuracy increases from $\delta = 1$, which equals approximately 250ms reaction time after the fixation beginning the saccade under consideration, up to $\delta = 3$, approximately 750ms reaction time, beyond which the accuracy drops again. This means on the training set, that if after three more fixations a classification is performed on a saccade under consideration, approximately nine out of ten of the label emitted by $v_{css}$-SVM classifier are correct and also approximately in nine out of ten true instances of skimming (eight out of ten for reading) the classifier manages to detect these.

Based on these findings we select $\delta = 3$ as the preferred window size and for each feature set the best trained classifier is chosen

[6]It should be noted that the actual reaction time can be significantly slower or faster, since the fixations times are merely assumed averages and can vary from less than 100ms to several hundreds of milliseconds (compare [Just and Carpenter 1980]), thus the true reaction times could range from below 300ms to over 3000ms.

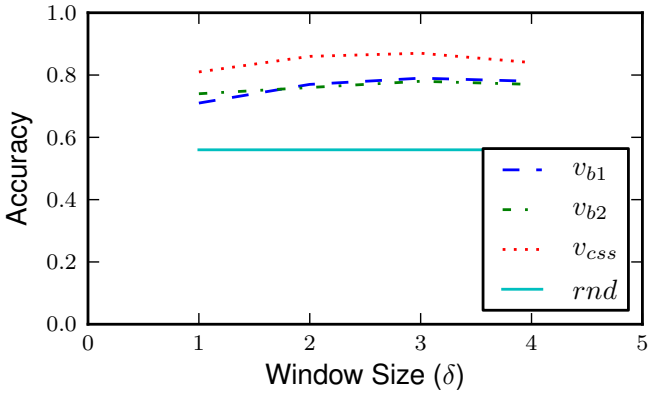


**Figure 5:** *Influence of the window size on the best classification accuracy for the given feature type. The overall optimum was produced by $v_{css}$ at $\delta = 3$ with an overall accuracy of* 88%. *Considering a window size of* 4 *already decreases the performance slightly. Each step of window size changes the reaction time approximately 250ms on average. The absolute baseline is set by rnd, which equals the guessing level base distribution of* reading *to* skimming *on our training set.*

and evaluated on the test set with containing the six other users, compare the middle part of Table 1. For the feature set $v_{b1}$ an RBF-SVM with $C = 2.86$ and $\gamma = 10^{-4}$ is the selected candidate, and for the $v_{b2}$ and $v_{css}$ features the ones with the values $C = 339.4$, $\gamma = 0.012$ respectively. Using these classifiers it turns out that while the performance of the classifier trained on the $v_{css}$ feature set achieves the same overall accuracy of 0.88, the performance of the character based approach increases significantly, while the performance of the classifier trained on the raw feature set drops down to 0.7; we will discuss these findings in the conclusion.

Eventually we also compute two variants of a feature type we find in the related work. The features $v_{ld3}$ and $v_{ld4}$ (compare the lower part of Table 1) are similar to the feature $Q'$ described in [Kollmorgen and Holmqvist 2007]. The vector $v_{ld4}$ is built using a window of four fixations around the saccade under consideration. It consists of the normalized lengths $r_i$, as well as the fixation durations of all fixations involved. In contrast to $Q'$ we did not take into account blink events. The vector $v_{ld3}$ is similar to $v_{ld4}$, with the exception that it has a window size of 3, the optimal value we found for our other feature types so far. Both features are trained using grid search on a RBF-SVM in 10-fold cross validation. We can see the overall accuracy values with 0.72 and 0.74 for both $v_{ld}$ types are somewhat low compared to the baselines. We will also come back to this in the conclusion.

### 4.2 Linear Classification and Model Adjustments

Since we now established somewhat of a peak classification accuracy at which the separability of both classes was highest our last step is to simplify the classification method itself. Considering that the feature set $v_{css}$ yields the best results, visualizing it reveals that both classes (compare Figure 6), while being partially overlapping, apparently have a shape simple enough for linear classification.

In contrast to a SVM a linear model with similar performance measures has several advantages, amongst them that it is easier to implement and more economically to execute, especially on small scale devices like tablets.

So taking the training set we construct a linear classifier and evaluate it with tenfold cross-validation, which gives an overall accuracy of 85.5% and precision/recall values of 0.81 / 0.86 for reading and 0.90 / 0.85 for skimming respectively. Evaluating the same classifier on the six users of the testing set yields an overall accuracy of 86.1% and almost the same precision/recall values. The final model is generated as

$$class = -2.97 + 5.36h + 0.17p$$

where *class* yields values greater than 0 in case the pattern resembles skimming, and values smaller than 0 in case of reading. Since *class* reflects the distance of a point from the classification boundary its absolute value of also reflects, to some extent, the certainty of the classification result. The closer the value is to 0 the more likely it could fall into the other class. Another very practical advantage of the linear classification is that the space of $v_{vss}$ becomes, in contrast to SVMs, predictably partitioned, which allows for an adjustment of the speed axis for the purpose of individualization. While on sufficiently large horizontal text a vertical pattern is most certainly an indicator against reading, the specific threshold when to classify a horizontal saccade length still as reading is dependent on the context and task for which this classification is being performed[7]. With this in mind we introduce an additional parameter $\beta$, which is being used to adjust the speed values prior to their classification, generating a derived speed:

$$p' = \beta\, p$$

The parameter can best be interpreted as a dilatation factor on the forwards-speed axis, effectively shifting data points with low angularity towards one side of the classification boundary. With it the influence of the horizontal saccades on the classification result can be adjusted, while at the same time the points with a high angularity remain mostly unaffected.

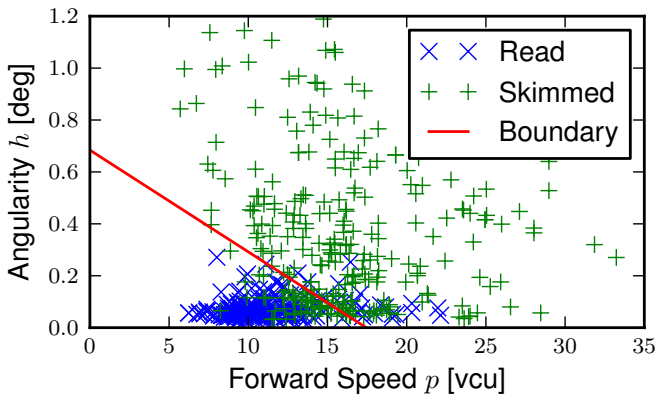


**Figure 6:** *Visualization of our ground truth when expressed as angularity and forward speed for the window size of $\delta = 3$ and the trained linear classification boundary. Adjusting $\beta$ will shift the points to the left or right and the less angularity there is involved the more impact the parameter has.*

## 5 Conclusion

Starting from the assumption that gaze patterns alone should contain most information needed to recognize and discriminate between reading and skimming patterns we first performed a data acquisition experiment. We invited a number of participants to read and comprehend a number of passages, embedded in mostly unrelated text, to elicit various forms of reading-for-comprehension

[7] In addition to other factor like for example the rate of non-newline regressions.

and searching patterns. We selected saccades as the base unit of consideration and two eye tracking experts labeled approximately 1400 of them independently into the classes *reading* and *skimming*. This resulted in 1055 saccades that we considered as ground truth for the construction, training and testing of an automated classifier. We investigated the expressiveness of three different feature types, a character-offset based feature type, somewhat in line to currently established classification methods and psychological knowledge; a raw vector feature type, only considering angles and normalized lengths; and a feature set we call contextual saccade shape, which condenses the overall shape of a saccade and its adjacencies into the two parameters *forward speed* and *angularity*. For all three feature sets we performed SVM grid searches to find the best parameters and window sizes and eventually presented a linear classifier that delivers a comparable performance.

Based on our findings there are a number of conclusions to draw. Comparing the three investigated vector representations, the contextual saccade shape appears to be most robust in terms of generalization and accuracy. On the training set it gives significantly better results than both other methods, even on small window sizes. On the testing set this observation also holds.

While the character based approach comes close on testing, its higher volatility over different window sizes, and also compared between both sets, are an indicator of a latent instability. However, it is unclear if this instability emerges from a principal problem of overlapping classes in the feature space, the variations in a-priori class distributions or an insufficient number of training examples for the given feature size. At a window size of $\delta = 3$ the vector $v_{b1}$ consists of 6 attributes while $v_{css}$ of only 3.

The impact of the window size on the overall accuracy is noticeable for all classifiers and in our study a context of three saccade into both direction gives the best results. Smaller windows are most likely to cut off parts of the actual perusal process (e.g., when considering only two saccades a newline regression might be hard to distinguish from a scanning pattern), while larger windows are likely to introduce too much unrelated noise. In case of both baselines an increased window size inherently leads to bigger feature vectors, which could probably be compensated by an simultaneous increase of training data, while in case of the overall saccade shape the expressiveness of $h$ and $p$ becomes diluted. All in all however, a window size of 3, thus 6 saccades or approximately 2 seconds of gaze data appears to be a reasonable size to distinguish reading from skimming.

Comparing our approach to existing work we outline a straightforward way to obtain and evaluate reading and skimming data on text. Presented to and labeled by two independent experts we generate ground truth based on fixations and saccades. The data serves as the foundation for an comparable assessment of the precision and recall of different algorithms. With the design of our baselines and implementation of $v_{ld*}$ we also tried provide at least a basic level of comparability to previous approaches. There are, however, certain, limits. Of all the papers addressing an automated reading detection we are aware of, only few target explicitly the discrimination of reading-for-meaning from skimming (such as fast perusal while still maintaining a mostly line-wise pattern) and can thus be compared directly.

The usage of character based offsets, presented in [Buscher et al. 2008], inspired our design of baseline$_1$. This work, however, expressly targets a line-based non-realtime detection and was not evaluated against labeled ground truth.

The *length-blink-duration* approach presented by [Kollmorgen and Holmqvist 2007] focusses on detecting reading vs. non-reading, where the latter consisted of activities such as *"editing periods [where a writer's] gaze may rest on text ROIs, although she is not reading"*. It was trained and evaluated on a set of apparently one hour of labeled gaze data. The authors report an accuracy of 0.86 (precision / recall 0.8) for realtime classification on their validation set. They also employed a neural network for classification and training which achieved precision and recall values of 0.86 for the reading class with reported training times of several hours. As mentioned earlier we implemented their feature set $Q'$ as $v_{ld4}$ and $v_{ld3}$, with the exception of not taking into account blink events. On our reading and skimming data this set performed with an overall accuracy of 0.74 and precision / recall values for reading of 0.69 / 0.74 respectively at a window size of 4.

Comparing these results one might be tempted to draw another conclusion: The main difference between $v_{b2}$ and $v_{ld*}$ is the move from saccade angles to fixation durations. Hence, it appears that to distinguish reading from skimming the saccade angles hold more predictive power than the fixation duration. While in principle we consider this statement to hold, one should keep in mind that for the generation of ground truth the eye tracking experts did not consider the fixation duration. This might, in turn, favor a feature set similar to those observed by the experts.

A very comprehensive report on the classification into three categories was published by [Simola et al. 2008]. In an experiment the users had to perform word search, question answering and a search / reading for the most interesting items in lists. While the word search was similar to skimming, the latter two categories apparently required intense amounts of reading for meaning. For their most predictive classifier the authors used a feature vector combination of fixation duration, saccade length and direction, as well as a binary regression flag. On their testing data a trained HMM was able to achieve an accuracy of 0.6, however on a much harder three-class problem with two similar categories. Treating their class $W$ as skimming and $A + I$ as reading instead (see Table 2 in [Simola et al. 2008]) we can compute an overall accuracy of 0.8 based on their reported data.

The reading detector described in [Holmqvist et al. 2003] and [Holmqvist et al. 2011] appears to consider saccade windows with a context of size 1 and is calibrated with samples from each recording to classify. It aims at detecting reading in contrast to non-reading, i.e., mostly scanning in their case. However, no classification results on labeled ground truth are being reported.

The approach described by [Campbell and Maglio 2001] aims at distinguishing reading from icon search. In contrast to all other approaches it does not use fixation and saccade data, but aggregated data over $100ms$ segments for both x- and y-directions. The relative movements are then evaluated according to some length-criterion and rated. Reading is considered once a certain threshold of aggregated points are reached. In an experiment where four participants either read a text carefully or performed a search for the Windows Excel document icon achieved approximately 0.90 accuracy. The reported recognition speed of their algorithm ranged *"from 200 to 3000ms, with an average of just over one second (1106ms)"*.

Considering our achieved accuracy levels, the experts' agreement and surveying the reported publications we got the impression that we came close to a realistic peak classification accuracy on our labeled data, similar to [Kollmorgen and Holmqvist 2007]. While an overall accuracy in the 'upper nineties' would, for ideas such a QuickSkim, surely be advantageous, we have to consider that both experts also only agreed to 83% on unproblematic data. In light of a limited saccade context and *finite* classification resources some patterns are apparently just too ambiguous. On the other hand this finding is hardly surprising. The transition from slow skimming to fast reading is a soft one. Both classes do not exist as well defined, crisp entities, but instead as regions around a zone of confluence. Hence in any classification scenario one should not only rely on the computed classification output, but also on the classifiers' confidence—in our case the distance from the classification boundary.

In overall we present a classification method that can be trained to

distinguish both patterns with high a precision, recall and overall accuracy. The low variance of the the feature vector we propose, in contrast to both baselines, is a good indicator that the algorithm should be reasonably robust, even considering new users. By varying the window size it allows for a certain tradeoff between latency and overall accuracy. It is furthermore straightforward to implement and requires only a few lines of code, making it a suitable candidate for small scale devices or embedded hardware. The clear notions of its features simplifies its understanding. With the introduction of $\beta$ it can also be adjusted in its sensitivity regarding its assignment of *inconclusive* gaze data to 'fast reading' and 'slow skimming'.

Eventually we would like to point out that our method requires little knowledge about the underlying text. Ideally it should be provided with information whether in the region was text after all[8] and the approximate character size. However, it can also be a suitable candidate in scenarios where such information is unavailable or expensive to generate. Possible examples could be mobile eye tracking applications, where a textual stimulus is read in the real world and only recorded by a scene camera. A *reading* result might then yield candidates where to perform an OCR analysis.

## 6 Outlook

In our scenario we are able to train a classifier on one set of users that is capable of generalizing to the reading patterns of other users. Being able to do so was not unexpected since the general pattern of *reading for meaning* on sufficiently large text in a given language is, to a large degree, influenced by the general layout and grammar. The main question was rather to what extent such a generalization would be possible, especially when disregarding most of the textual stimulus. However, when these preconditions are not given anymore, the overall patterns evoked will look significantly different, examples include small columns of text, bullet point lists or any other scenario where a mostly horizontal reading pattern has no space to evolve. Investigating a feature representation and classification method that can deal with or consider these layout details will expand the field of use of such a real time classifier.

Also we have to acknowledge that the our set of users was, with regard to the general public, obviously not representative, considering that the majority of our users were male university students. Therefore two interesting steps would be to analyze how our actually reported linear classifier performs on a completely different user group, and how a newly trained classifier including these or both groups would perform. First results from demo applications in which we integrated the presented linear classifier and used it on a set of different texts, font sizes (e.g., 11pt) and languages (English instead of German) indicate that even in these cases its generalization to different users does work.

Lastly we think that having a classifier capable of robustly distinguishing reading from skimming patterns on text in real time facilitates many new interactive and reading-context enriched end user application. They can use the knowledge about what has been read and what currently is being read to contextualize personalized information processing and change the interface according to the user's current state of mind.

## 7 Acknowledgements

We would like to thank Horst Bunke and Seiichi Uchida for their valueable remarks on SVM classification. This work was supported by the German Federal Ministry of Education, Science, Research and Technology (bmb+f), (project Perspecting, grant 01IW08002).

[8]For example, not to misinterpret a *skimming* result when the user just performed GUI tasks such as icon search instead. In fact for most desktop scenarios, when there was no text nearby, the algorithm does not even need to run.

## References

Biedert, R., Buscher, G., and Dengel, A. 2010. The eyeBook - Using eye tracking to enhance the reading experience. *Informatik-Spektrum 33*, 3 (June), 272–281.

Biedert, R., Buscher, G., Lottermann, T., Schwarz, S., Möller, M., and Dengel, A. 2010. The Text 2.0 Framework. *Workshop on Eye Gaze in Intelligent Human Machine Interaction*.

Biedert, R., Buscher, G., Schwarz, S., Hees, J., and Dengel, A. 2010. Text 2.0. *CHI EA '10: Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems* (Apr.), 4003–4008.

Buscher, G., Dengel, A., and Elst, L. v. 2008. Eye movements as implicit relevance feedback. *CHI '08 extended abstracts on Human factors in computing systems*, 2991–2996.

Buscher, G. 2010. *Attention-Based Information Retrieval*. PhD thesis, University Kaiserslautern, Kaiserslautern.

Campbell, C. S., and Maglio, P. P. 2001. A robust algorithm for reading detection. In *Proceedings of the 2001 workshop on Perceptive user interfaces*, New York, NY, USA, 1–7.

Chi, E. H., Hong, L., Gumbrecht, M., and Card, S. K. 2005. ScentHighlights: highlighting conceptually-related sentences during reading. *Proceedings of the 10th international conference on Intelligent user interfaces*, 274.

Holmqvist, K., Holsanova, J., and Barthelson, M. 2003. Reading or scanning? A study of newspaper and net paper reading. In *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*. Elsevier Science Ltd.

Holmqvist, K., Nystrom, M., Andersson, R., Dewhurst, R., Jarodzka, H., and van de Weijer, J. 2011. *Eye Tracking*. A Comprehensive Guide to Methods and Measures. Oxford Univ Pr, Nov.

Hyrskykari, A. 2006. Eyes in Attentive Interfaces: Experiences from Creating iDict, a Gaze-Aware Reading Aid. *acta.uta.fi.*

Just, M. A., and Carpenter, P. A. 1980. A theory of reading: From eye fixations to comprehension. *Psychological Review 87*, 329–354.

Kollmorgen, S., and Holmqvist, K. 2007. Automatically detecting reading in eye tracking data. *Lund University Cognitive Studies*.

Rayner, K. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin 124*, 3, 372–422.

Rayner, K. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*.

Reichle, E., Pollatsek, A., Fisher, D., and Rayner, K. 1998. Toward a model of eye movement control in reading. *PSYCHOLOGICAL REVIEW-NEW YORK-*.

Simola, J., Salojärvi, J., and Kojo, I. 2008. Using hidden Markov model to uncover processing states from eye movements in information search tasks. *Cognitive Systems Research 9*, 4 (Oct.), 237–251.

Underwood, N. 1985. Perceptual span for letter distinctions during reading. *Reading Research Quarterly*, 20, 153–162.