An Online Evaluation Platform for Proactive Information Retrieval Task

Li Yao*
*School of Science and Technology, Aalto University
Espoo, Finland
lyao@cis.hut.fi

Antti Ajanki†
†School of Science and Technology, Aalto University
Espoo, Finland
antti.ajanki@tkk.fi

Abstract

The last decade has seen a great progress on the research and applications of information retrieval. The major improvement has been made to combine traditional keyword-based search with implicit inputs such as eye movements or fixations, mouse clicks and voice commands, thus gradually forming a new branch under the name of proactive information retrieval. This paper focuses on the the study of eye movement tracking in the document retrieval by constructing a universal online research platform that merges all the research steps - collecting data, feature selection, model selection and testing into a flexible and extendible cross-platform software system.

1 Introduction

Proactive information retrieval (PIR) is a relatively new research field that incorporates explicit user inputs such as search keywords from keyboard and clicks from a mouse with additional implicit inputs such as eye movement, speech, blood pressure and even facial expression. Buscher and Dengel (2009) shows that personalized information, if available, can be combined into PIR and the retrieval shows high efficiency and accuracy. The most commonly used personalization approach is to include a user's personal information such as age, nationality, sex, educational background and career. However, those kind of information are always too personal to be safely collected. Thus, latest research has been focused on collecting the observation of eye movement that is directly related to a user's personal interest or attention. Miller and Agne (2005) and many other research discussed the feasibility of the attention-based information retrieval using eye tracker data.

So far, many research has been devoted to studying the implicit feedbacks as a complement of traditional explicit inputs. As a relatively thorough report, He Zhang and Laaksonen (2008) presents a literature survey conducted to review the current state of the art in research concerning the use of eye movement measurements and other non-conventional and implicit relevance feedback modalities in content-based image and information retrieval. In the work of Viitaniemi and Laaksonen (2008), the author presents the results of a series of experiments where knowledge of the most relevant part of images is given as additional information to a content-based image retrieval system using mouse clicks. Furthermore, Arto Klami and Kaski (2008) presents important results on inferring the relevance of images based on implicit feedback about users' attention, measured using an eye tracking device.

Specifically in terms of PIR using eye movement, according to Campbell and Maglio (2001), the eye movement data is utilized to two very different types of interfaces: command and non-command. Command-based interfaces use gaze location to directly issue commands to the system while non-command interfaces use gaze information to indirectly tune the system to the user's needs. Command-based interfaces is most conveniently used to control the system such as clicking by fixation. In the field of information retrieval, the non-command interfaces, combination of traditional application with gaze, are used to adapt the computer to return more accurate information. However, gaze data usually comprises large amount of noise due to the flexibility of human's eyes. Different person may have very different reading behavior and habit that make the universal modeling a non-trivial task. Hardoon and Pasupa (2010) explores the idea of implicitly incorporating eye movement features in an image ranking task by combining image features together with implicit feedback from users' eye movements in a tensor ranking Support Vector Machine and shows that it is possible to extract the individual source-specific features. Zakria Hussain and Shawe-Taylor (2010) demonstrates that by

using a greedy Nystrom algorithm on the eye movement features of different users, we can find a suitable low-dimensional feature space for learning the individualized behavior.

Based on the principles briefly discussed above, many applications of PIR have been developed for both practical and academic purposes. One of the earliest and most representative system can be found in Maglio et al. (2000) where an attentive information system called "SUITOR" was developed in IBM. The system monitors a user's behavior of operating a computer such as web browsing, word processing and provides suggestions and helps regarding to a user's current activity. The main source of information comes from the tracking of the eye movements. In Laaksonen et al. (1999), a PIR system named PicSOM, is introduced for content-based information browsing and retrieval system based on the Self-Organizing Map (SOM). In Laaksonen (2008) and Viitaniemi and Laaksonen (2008), the authors defined and implemented communication principles and data formats for transferring enriched relevance feedback to the PicSOM content-based image retrieval system used in the PinView[1]. The goal of PinView is a proactive personal information navigator that allows retrieval of multimedia - such as still images, text and video - from unannotated databases. The modalities of enriched relevance feedback in PinView include recorded eye movements, pointers and keyboard events and audio including speech. Another delicate interactive PIR system can be found in Lszl Kozma and Kaski (2009) which introduces GaZIR, a gaze-based interface for browsing and searching for images. The system computes on-line predictions of relevance of images based on implicit feedback, and when the user zooms in, the images that are predicted to be the most relevant are brought out.

Technically, all the PIR systems discussed above rely on a collection of features which represent the most essential part of information collected from a mouse, a keyboard, a microphone and an eye tracker. The collection of features, on one hand, differentiate individuals from each other and on the other hand, provide enough information for the computer to model the exact behavior or intention of a specific user. Among various of implicit inputs, eye movement tracking is used in different tasks such as document retrieval, image retrieval, and some subtasks such as reading/skimming detection. The selected set of features is coupled with a specific task and a model that takes the selected features and generates performance measurements indicating the acceptability of the given feature-model pair. However, in eye movement tracking, the number of collectable eye movement features can be enormous as a result of exquisite structure of human eyes. Furthermore, the number of different models one can utilize is even larger — logistic regression, support vector machine, artificial neural network and their variations — just to name a few. It remains a confusing question that which feature-model-task combination gives the best performance.

The traditional methods to find the best combination are limited in offline mode containing mainly two stages. In the training stage, a task is firstly defined, then data is collected, features are abstracted from the collected data and models are trained using the features. The theoretical performance of the tuned task-feature-model in this stage can be illustrated by using an isolated test set. This is called theoretical performance for the reason that it is entirely based on the data collected from a group of subjects who only represent a small portion of general population. In the testing stage with the fixed task-feature-model tuple, the practical performance test of a task-feature-model can finally be carried out by involving more subjects in the experiment. The actual practical performance is only available by using separate analysis after the whole experiment is finished, which is why this 2-stage process is named after the word "offline". One may realize that the whole process is time-consuming and involves large amount of work such as constructing an experiment platform for different tasks to collect training data, building and testing different combination of task-feature-model to get theoretical performance measurement and finally analyzing the practical performance. Furthermore, different researchers usually have their own software platform to perform specific tasks and their own models trained in different platforms such Matlab, R or even Python on different operation systems such as Windows and Linux. Thus, a new universal software platform is needed to simplify the whole research process and to provide flexibility and reliability to PIR research. In this paper, we focus on developing such a system having the following promising features:

- It is a proactive document retrieval system using eye movement data.
- Researchers can define different task-feature-model components in their favorite programming languages (e.g. Python and Matlab) and

[1]PinView is an EU FP7 funded Collaborative Project 216529. For more information, go to the project webste: www.pinview.eu

components can be easily plugged into the system.

- It incorporates stage one of training and stage two of practical testing into a unified software system that is connected with the backstage research environment.
- It provides online evaluation feedback in the practical testing stage so that the practical performance of a trained task-feature-model tuple is shown in real time during the testing stage while the subjects are performing the experiment.
- Even though the system is made to evaluate task-feature-model combination in an efficient way in the research, the online evaluation feedback can be used in many other purposes such as giving suggestions to a user or providing guidance.
- The system can be easily extended to proactive image retrieval system using eye movement data.

In what follows, section 2 overviews the system and demonstrates the usage of the system. Section 3 goes deeper into the implementation of the system. Section 4 performs the experiment: Reading/skimming detection in order to show exactly how the system works with different task-feature-model components in the PIR research. Note that this work is not dedicated to any specific research task but the emphasis of use of the system in various of tasks and how it simplifies the experiment and data analysis and meanwhile guarantees accurate results.

2 System Overview

2.1 High level architecture of system

Figure 1 shows the top level architecture of the system. The system consists of three seperate components located in two different operating systems. The Firefox extension is installed in Firefox browser in the Windows computer in which the eye fixations are collected by Tobbi 1750 eye tracker driver and preprocessed to XML-formatted data. The preprocessed XML fixation data is then relayed to the Apache HTTP server.

The Apache HTTP server is installed on the Linux machine where fixation data from Firefox extension is received via standard Common Gateway Interface (CGI) and further processed to generate series of commands sent to the experimental module via socket client/server communication.

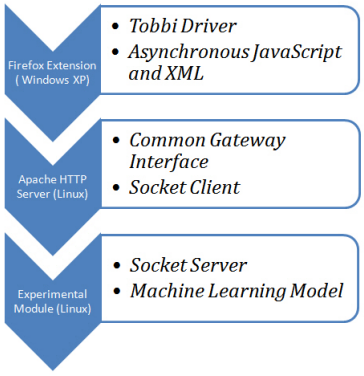

Figure 1: System architecture.

The experimental module is designed to run on any Linux machine inside a normal computer network, which is achieved by setting a socket server inside the experimental module and a socket client inside HTTP server. The commands from HTTP server are designed to be executed on specific experimental platforms, which in the current system, are Matlab and Python modules. Take Matlab as an example. The socket server implemented inside Matlab keeps waiting for the message from socket client. The received messages from the socket client are executable Matlab commands containing fixation data. The Matlab commands call the models implemented in Matlab to analyze the fixation data.

Data travels from Firefox to experimental module and then back to Firefox. It has two parts: evaluation feedbacks showing the response and performance of current machine learning algorithm, and the content of the next document. Due to the intensive real-time data communication, a stable network connection and bandwidth with at least 500KB/s is necessary.

2.2 Functional scenarios

Two basic functional scenarios are “collecting training data” and “testing model”. Firefox browser plugin shows a series of documents by using and collects the information of eye fixation data. The fixation data is then used for a research purpose, for instance, modeling a subject’s behavior. After a model is trained, the system is further used to test the performance of the trained model.

2.2.1 Scenario one: collecting training data

A subject selects an interesting topic in the interface shown in Figure 2. Then a series of random documents on that topic are shown to the subject. The

subject reads given documents one by one as in Figure 3. While reading, the subject's eye movements are collected by the system. The collected data is then used in the research. During this stage, the experimental module does not load any algorithm to analyze the fixations, so the system gives no evaluation feedbacks.

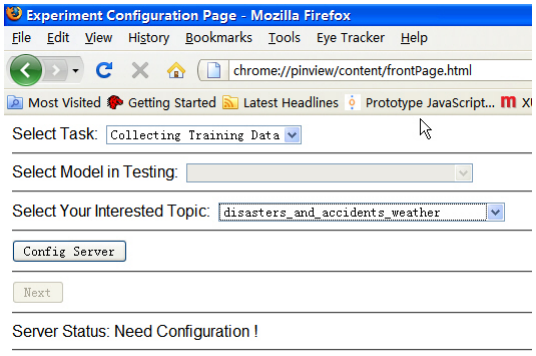

Figure 2: Select task: collecting training data. Documents on a selected topic are loaded. No model is involved.

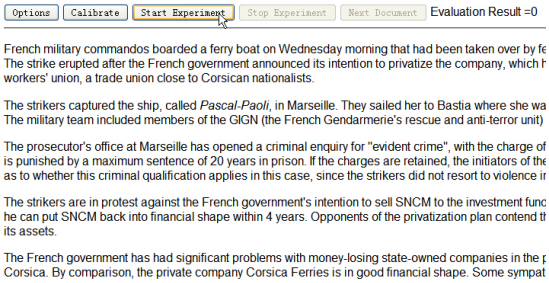

Figure 3: Subjects read through documents while training data is collected at the backstage. The performance evaluation of task-feature-model is not provided since no model is loaded behind the scene.

2.2.2 Scenario two: practical test of model performance

After training a model using data from scenario one and plugging it into the system, it is time to run the practical test with new test subjects. In this scenario, the trained model is loaded into the experimental module as in Figure 4. The system collects the fixation data and evaluates the fixations sequentially by using the preloaded algorithm and gives evaluation feedbacks to the Firefox extension. As in Figure 5, the evaluation result is shown at the top of the document. Another alternative way is to play a music note in accordance to the value of evaluation result.

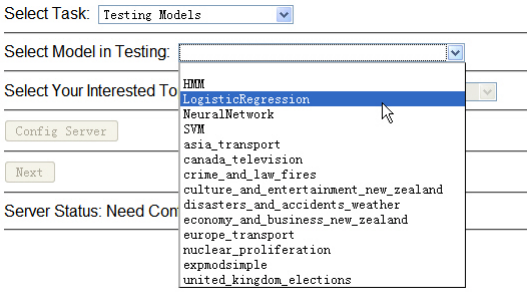

Figure 4: Preparing stage for the practical test after task-feature-model has been tuned with training data. Selecting a topic equals to selecting a model at the backstage.

Figure 5: Subjects read through the documents with model loaded and performance evaluation returned in real-time. Evaluation is normalized between 0 and 1.

3 System Implementation

3.1 Implementation of Firefox extension

Firefox, developped by Mozilla organization, provides the standard of constructing the entension to Firefox browser. In general, every extension consists of two parts, XPCOM and a user interface. XPCOM, which builds a communication channel between eye tracker and Firefox is coded by C++. The implementation of the user interface requires several techniques including Javascript, XML user interface language (XUL) and HTML. XUL and HTML construct the user interface controls such as buttons and labels. Javascript collects raw fixation via XPCOM, and maps each fixation to a paragraph in a document

Table 1: Functionality of CGI scripts

CGI scripts	Functionality
configure	Load experimental module, load model list, configure tasks, (collecting data or testing model)
collect	Receive fixations, communicate with Python/Matlab, response feedback to Firefox
format	Generate training data, format all communications to XML file

and transforms the mapping into standard XML format. The XML-formatted mappings are then delivered to HTTP server via Javacripts' embeded object XMLHttpRequest, which is also known as AJAX.

3.2 Implementation of Common Gateway Interface in Apache HTTP server

The Common Gateway Interface (CGI) is a standard protocol that defines how webserver software can delegate the generation of webpages to a console application. Such applications are known as CGI scripts that in principle can be written in any programming language. Apache HTTP Server provides a container to which the Firefox extension sends data by standard HTTP requests. The CGI scripts wait in the HTTP container for the incoming requests, analyze the requests and compose HTTP responses back to Firefox extension. The system includes three CGI scripts written in C, dealing with 3 different tasks in Table 1. Those three scripts explicitly use GNU Cgicc library to traverse XML requests and compose XML response that is then fed back from HTTP server to Firefox extension.

More specifically, *configure* corresponds to the configuration phase shown in Figure 2 where it dynamically loads the topics or models into the dropdown lists and notifies the server with a subject's selection after clicking "*config server*" button.

collect corresponds to the phase shown in Figure 3 and Figure 5. Evaluation result is shown as "Evaluation Result". When a subject clicks *next document* button, the request of a new document is inserted into the XML request so that in the corresponding response, not only the evaluation but also the content of new document are provided to Firefox.

format is responsible to convert all isolated XML requests into one well-formatted XML file so that training data are easily collected for the future use.

3.3 Implementation of experimental module

This part gives details of experimental modules. The whole system consists of two experimental modules, Matlab and Python. Each module further consists of several models each of which is in fact a specific machine learning algorithm handling feature selection and classification. Besides, each module has its own implementation of a socket server.

3.3.1 Implementation by Matlab

Matlab implementation of the experimental module makes use of the power of embedded Java network communication package "java.net.SocketServer", I/O package "java.io" and other utility packages such as "java.lang.String".

Java receives fixation requests from CGI and judges the type of command from CGI. The "select_task" containing parameters "model" and "task" is always the first command that is relayed from Java and executed by Matlab. It sets up the variable "task" inside Matlab module and loads the model. After that, if the command is either "handle_fixation" or "next_document", Matlab searches the .m file with the same name as the command and executes the file. If the command is something else other than the previous three, Java will by default recognize that command as the stop server signal and thus stops the server. Basically, any unrecognized commands will stops the Matlab server. For example, "telnet localhost 6666" in Linux console will do. The Matlab server cannot be stopped inside Matlab command line for some safety reasons of the operating system.

From the other way around, when a task is set to "testing", "handle_fixation" returns evaluation result (otherwise, "NoEvalution") to Java. Java will send back the result to CGI.

Another practical concern is the change of workspace in Matlab. Since in Matlab experimental module, socket server and models are saved under different directories, the change of workspace needs to be explicitly added into the Matlab scripts. The one-server-multiple-models structure relies on accurate change of workspace.

3.3.2 Implementation by Python

The implementation of Python module uses the same one-server-mulitple-models structure as Matlab. The difference is that Python is sufficient to handle both

socket communication and model implementation. The command sets are the same as those used in Matlab. Python socket server can be stopped by using keyboard combination "CTRL+C" in Linux console used to start the Python server.

4 Experiment on the system

The importance of the system is that it can be used as an architecture which may also be enhanced and extended in various research tasks. One example is demonstrated in this section. Other experiments may be designed based on the different goals.

4.1 Reading and skimming detection

The last hundred years has seen a lot of research focusing on the behavior of the human's eyes when reading. The most important results can be found in K.Rayner (1998): When reading silently the eye shows a very characteristic behavior composed of fixations and saccades. A fixation is a time interval of about 200-250 ms on average during which the eye is steadily gazing at one point. A saccade is a rapid eye movement from one fixation to the next. The mean left-to-right saccade size during reading is 7-9 letter spaces. This depends on the font size and is relatively invariant concerning the distance between the eyes and the text. Approximately 10-15% of the eye movements during reading are regressions. Reading detection provides an accurate way of defining the level of a user's interest. For example, reading shows one's consistent interests of the content while skimming shows the opposite. As in Georg Buscher and van Elst (2008), by monitoring the distance and direction in letter spaces, features such as read forward, skim forward, long skim jump, short regression and unrelated move are abstracted and then used to generate both reading and skimming weights for each fixation. Both reading and skimming weights are then added together as a score to judge whether a user is reading or not. A threshhold for reading is predefined. If the score exceeds the threshhold, system draws the conclusion that a user is reading. If not, a user is skimming. The similar approach is also dopted in Campbell and Maglio (2001).

The differentiation between the human behavior of reading and skimming shows its importance in proactive information retrieval task. After the discussion of the suitable features, the basic classification algorithms are briefly explained followed by the results.

4.1.1 Dataset and feature abstraction

By following the Figure 2 and Figure 3, the data was collected from 2 subjects who was told before the experiment to look for the words starting with capital letter, which mimicks skimming, or to understand the story of the given short passage, which mimicks reading. The whole data set includes 17 documents aimed for skimming and 33 documents aimed for reading. Although with limited size, the data is enough to show the significant use of our system.

In order to simplify the reading/skimming detection task, only document level features that summarize the fixations across the whole document are taken into consideration. The raw fixation directly collected from Firefox extension is of more than 10 dimensions including docID, fixation sequence, 3 pairs of (x,y) coordinates, fixation duration and other trivial information. From the raw fixation, we define new feature tuple (N, d_1, d_2, D) that is considered most useful and informative to this specific reading/skimming detection task:

- $\mathbf{N}$: Number of fixations per document
- $\mathbf{d}_1$: Average jumping distance of x coordinates between each pair of successive fixations
- $\mathbf{d}_2$: Average jumping distance of y coordinates between each pair of successive fixations
- $\mathbf{D}$: Average fixation duration per document

The dataset includes 50 rows and 5 columns. Each row $\mathbf{x}_i, \mathbf{x} \in \Re^n$ represents one document. In our following experiment, $n = 5$ is used for simplicity, however, one can come up with more complex features to better represent a document for different purposes. First 4 columns (N, d_1, d_2, D) show the feature tuple collected from the raw data. The last column shows the classification label c_i in which $c_i = 1$ represents reading and $c_i = -1$ represents skimming.

4.1.2 Logistic regression

Logistic regression in Hilbe (2009) is considered as one of the most basic linear classifiers in machine learning algorithms. It tries to construct a hyperplane that separates the instances of two classes by minimizing the classification error. The logistic function $f(z) = \frac{e^z}{e^z+1} = \frac{1}{1+e^{-z}}$, where $z = \mathbf{w} \cdot \mathbf{x} = w_0 + x_1 w_1 + x_2 w_2 + \cdots + x_d w_d$, w_d is coefficients and vector $[x_1, x_2, \ldots, x_d]$ represents one observation, is used to decide which class the given instance belongs to. The best model comes from the minimization of the classification error which is defined

as $e = \frac{1}{n}\sum_{i=1}^{n}(c_i - \tilde{c_i})$ where c_i is the observation label and $\tilde{c_i}$ is the classification result equal to $f(z)$ and n is the total number of observations.

4.1.3 Soft margin support vector machine

Support vector machine (SVM) is another widely used family of classifiers. Linear SVM constructs the hyperplane that separates the given two classes by maximizing the margin between support vectors. Seeking for a plane to separate two classes of instances becomes infeasible when instances are by nature not linearly separable. Soft margin SVM, detailed in Alpaydin (2004), relaxes the separation to allow misclassification. In Figure 6, by minimizing $\frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi^i$ subject to $c_i(\mathbf{w}\cdot\mathbf{x} - b) \geq 1 - \xi^i$ where ξ^i stores the deviation of each instance i from the margin and C is the complexity trade-off, the optimal separation plan $\mathbf{w}\cdot\mathbf{x} - b = 0$ is computed out by using all the instances between the plane $\mathbf{w}\cdot\mathbf{x} - b = 1$ and $\mathbf{w}\cdot\mathbf{x} - b = -1$.

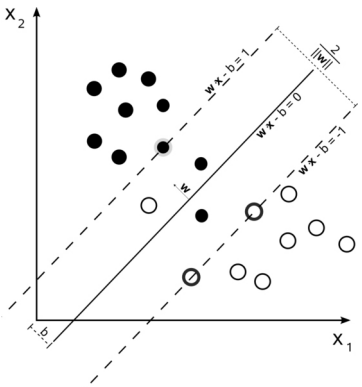

Figure 6: Soft margin support vector machine.

4.1.4 Classification result

The whole data has 50 feature vectors representing 50 documents either by reading or skimming. Their d_1 and d_2 are plot in Figure 7. According to the plot, we can generally get the sense that reading and skimming are not perfectly separable by straight line. Since the size of data available is relatively small and in order to minimize the prediction error, the expected prediction error is computed for leave-one-out cross-validation. Different combinations of features are also tried with both logistic regression and SVM and the classification result is shown in Table 2.

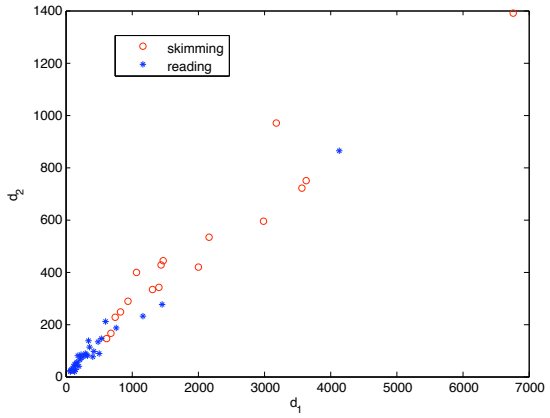

Figure 7: Plot of feature subset (d_1, d_2)

Table 2: Leave-one-out cross-validation

Feature sets	**Expected error rate of LG**	**Expected error rate of SVM**
N	0.328	0.325
d_1	0.195	0.225
d_2	0.124	0.135
D	0.242	0.235
N, d_1	0.147	0.190
N,d_2	0.142	0.130
d_1, d_2	0.125	0.162
N,d_1, d_2	0.125	0.095
N,d_1, d_2,D	0.035	0.085

5 Conclusions

In terms of the system, the current performance is acceptable with at least 3 fixations being handled per second both in Python and Matlab modules. This performance is considered as enough foundation to evaluate the advantage and drawback of any given algorithm in real-time. The evaluation feedback is expressed as the music note of different frequencies regarding to the value of the feedback. However, whether the use of music notes is sensible and informative enough to indicate the performance of the algorithm is still under discussion. The future work could consists of finding more reliable approaches.

In terms of the reading/skimming detection, data used in the experiment has its limitations. After all, only two persons are involved in collecting data. Two persons are insufficient to represent the mass population. Obviously, the system is adapted to model the behavior of only these two persons. Whether the feature is widely suitable for different persons needs more experiment to prove.

In terms of the availability of our system, unfortu-

nately, it is not going to be delivered into the public domain quite soon. But its variations have been internally distributed into the research and construction of the PinView system.

Acknowledgements

We would like to give thanks to all the researchers and developers involved in the PinView project. Their outstanding work provides an essential building block of our system.

References

Ethem Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2004. ISBN 0262012111.

Tefilo de Campos Arto Klami, Craig Saunders and Samuel Kaski. Can relevance of images be inferred from eye movements? In *MIR '08: Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval*, pages 134–140, 2008.

Georg Buscher and Andreas Dengel. Gaze-base filtering of relevant document segments. In *WWW*, 2009.

Christopher S. Campbell and Paul P. Maglio. A robust algorithm for reading detection. In *Proceedings of the 2001 workshop on Perceptive user interfaces*, 2001.

Andreas Dengel Georg Buscher and Ludger van Elst. Eye movements as implicit relevance feedback. In *CHI*, 2008.

David Hardoon and Kitsuchart Pasupa. Image ranking with implicit feedback from eye movements. In *In Proceedings of the 6th Biennial Symposium on Eye Tracking Research and Applications (ETRA'2010), Austin, USA*, 2010.

Markus Koskela He Zhang and Jorma Laaksonen. Report on forms of enriched relevance feedback. Technical report, Helsinki University of Technology, Department of Information and Computer Science, 2008.

Joseph M. Hilbe. *Logistic Regression Models*. Chapman and Hall/CRC Press, 2009.

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

J. T. Laaksonen, J. M. Koskela, and E. Oja. Picsom - a framework for content-based image database retrieval using self-organizing maps. In *In 11th Scandinavian Conference on Image Analysis*, pages 151–156, 1999.

Jorma Laaksonen. Definition of enriched relevance feedback in picsom. Technical report, Helsinki University of Technology, Department of Information and Computer Science, 2008.

Arto Klami Lszl Kozma and Samuel Kaski. Gazir: Gaze-based zooming interface for image retrieval. In *In Eleventh International Conference on Multimodal Interfaces (ICMI-MLMI),Cambridge, Massachusetts, USA*, 2009.

Paul P. Maglio, Rob Barrett, Christopher S. Campbell, and Ted Selker. Suitor: an attentive information system. In *IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces*, New York, NY, USA, 2000. ACM.

Tristan Miller and Stefan Agne. Attention-based information retrieval using eye tracker data. In *international conference on Knowledge Capture*, 2005.

Ville Viitaniemi and Jorma Laaksonen. Evaluation of pointer click relevance feedback in picsom. Technical report, Helsinki University of Technology, Department of Information and Computer Science, 2008.

Kitsuchart Pasupa Zakria Hussain and John Shawe-Taylor. Image ranking with implicit feedback from eye movements. In *In Proceedings of the 6th Biennial Symposium on Eye Tracking Research and Applications (ETRA'2010), Austin, USA*, 2010.