Eye-Tracking Assisted Search Engine

Pablo FOCKE
Université Grenoble Alpes
Grenoble, France
pablofocke@gmail.com

Supervised by: Francis JAMBON & Philippe MULHEM

I understand what plagiarism entails and I declare that this report is my own, original work.
Pablo Focke, 26/04/2021

Abstract

This work focuses on the usage of eye-tracking devices to improve query results of a search engine. We extend an existing software platform dedicated to such problem by integrating a new eye-tracker, the Tobii 4C, and a new information search system, Terrier v5. We then compared the results of our experiments to previous work [Vaynee Sungeelee2020] by repeating the same experiments with our new upgraded machinery.

1 Introduction

Web information retrieval is present in our day to day life, whenever we search for something on Google we are unconsciously asking the browser to find all the documents related to the few words we put on the query and to top it all of we expect to get the most relevant results on top. Google alone has more than 130 trillion indexed pages and handles 3.8 million searches per minute on average across the globe. Nonetheless, from the moment you press enter to the moment you get your results usually no more than a second passes. This work is done by the information retrieval system.

After an initial query asked by a user, classic Web search engines usually return a list of results composed of extracts (called snippets) of supposedly relevant documents. Usually the user then chooses the document that seems most interesting by clicking on the corresponding snippet, and then consults the document. If the document is not relevant, the user can return to the results page, and choose another document, etc.

Eye-tracking can study how the user reads each snippet of the results page, and determine if some words are more particularly watched by the user. It is thus possible to extend the initial query of the user with these words, to send a new more precise query, and thus to automatically improve the search results without explicit intervention by the user.

Our work is based on an existing mockup that gave interesting results [Vaynee Sungeelee2020]. Our main objective is to improve this mockup with the new eye-tracking device (Tobii 4C with Pro SDK). This device being more precise and allowing the user slight movements of their head while remaining accurate is better adapted for the job than the previous Eyetribe Eye tracker used.

In this paper we also upgraded the information retrieval system from terrier v4.0 to terrier v5.4, the latter has an integrated snippet generator that we will use to simulate a search engine opposed to the previous version where we had to generate the snippets ourselves.

Contents

2 Context

2.1 Information Retrieval

In information retrieval, we have a large set of documents and a system that is in charge of organising them by their content and be able to retrieve them quickly when the user wants. A query is a set of words given by the user as an input to this engine it gives the engine clues of what the user is looking for.

Search engines, based on classical information retrieval systems, take a query (i.e. a list of keywords) as input and

use it to filter the set of indexed documents. Only relevant documents to the query are taken into account and the list of documents is ordered by decreasing relevance. However a keyword can span over multiple topics (e.g. "earth" can either refer to the "planet earth" or the "earth material"), so some users might give ambiguous queries.

As a consequence, a large set of methods were developed to improve the relevance of the documents returned. Among these methods, "query expansion" is a common technique that consists in selecting and adding terms to the query to improve the results of the search. Most of query expansions methods rely on explicit actions from the user (e.g. document selection). Although this technique could drastically improve the results, asking directly for explicit feedback form the user can be a burden to the user.

2.2 Eye Tracking

An eye tracking device, as its name suggests, tracks the eyes of the user in order to get information such as the point in the screen where he's looking at, the size of his pupils, etc. A lot of interesting information can be obtained from the device [Navalpakkam and Churchill2018, Holmqvist2017]. In our work, we only focus on "fixations" which correspond to "pauses" within eyes movements, where the eyes keep looking at a target for some time (opposed to saccades which are short and fast movements between fixations). These fixations represent signs of cognitive processing of information and are noticeably useful for determining if a user is reading a specific word of a text on screen.

2.3 Eye Tracking Assisted Information Retrieval

Assuming that the eye tracker has a prior knowledge of the positions of the words in the search engine result page, it is able to analyze eye movements and so to determine which words are read by the user (i.e. fixations). By keeping track of the read words, methods to improve the query via query expansion [Buscher *et al.*2009, Eickhoff *et al.*2015] could be proposed. For example if the query is "earth" and the user is looking at words such as "Jupiter" and "Mars", adding any of those therms to the query will most likely exclude all the results non-related to planets. Obviously, the more accurate the eye tracker is, the better the expansion will be.

3 Previous Work

Our work mostly rely on the [Vaynee Sungeelee2020] work. I this section, we mainly present the architecture of the software and the main experimental results. More detailed information and a deeper explanation of the goals of assisted information retrieval can be found in the original paper.

3.1 Software Architecture

The software architecture of the proof of concept proposed by [Vaynee Sungeelee2020] is presented in figure 1. This architecture is globally preserved in our work, with changes that do not play an important role on the bigger picture.

Eye Tracking system

The eye tracking system is composed of the eye tracker device and a program used to retrieve its information, this system will send information corresponding to the point the user is looking at in screen for later analysis. This system also sends display information such as screen width and height.

Eye Tracking Analysis

The points sent by the eye tracking system are then processed and refined into fixations (when the user maintains eye contact with a point for a certain time). This analyser also re-

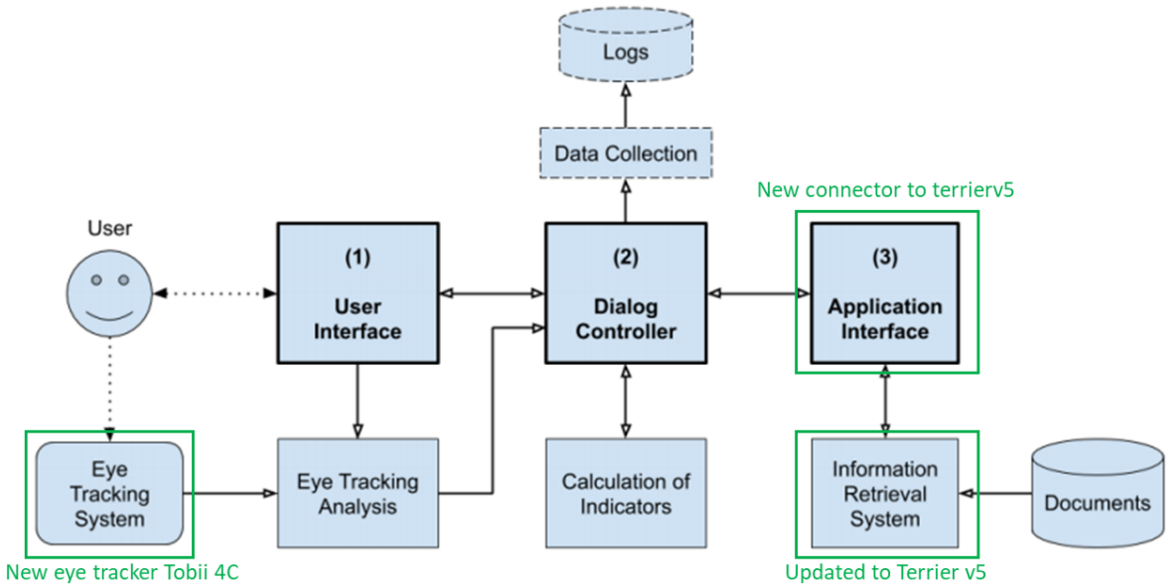

Figure 1: Architecture of the application (by [Vaynee Sungeelee2020])
In green the changes made for this paper

ceives the position of each word displayed on screen by the User interface, each word correspond to a rectangle zone on screen containing the value of the word to identify it (e.g. ”the”, ”planet”, ”earth”). The figure 2 presents a visual example of what zones and fixations look like. The zones corresponds to the yellow rectangles, the user gazes points are represented by grey points, and the fixations are shown as blue circles (the radius is proportional to how much the eye moved around the area), the blue lines are just the movement of the eye from one fixation to the other. Every time the user fixates his gaze on a zone/word the analyser sends this information to the Dialog Controller.

Figure 2: Eye tracking data visualization with Zones (yellow rectangles), Gaze Positions (grey points), and Fixations (blue circles)

Information Retrieval System

In a Web search engine, when you ask for a query, you wait for a couple of seconds, then you get a list of results related to the keywords you put in it. During that time, the search engine filters documents from its database based on their computed relevance to the query and presents them on a decreasing order of relevance. This is done by the information retrieval system (IRS for short). Since this process is not the main topic of our study, we opted to use Terrier V5.0, “an open source search engine, readily deployable on large-scale collections of document” [TerJune 2021].

In an Information Retrieval System (IRS), each document is *indexed* to describe its content as a set of ”attributes” based on their words. During the retrieval process, a query (composed of words) issued by a user is first analysed according to the same set of content attributes to describe its content. Then, the query content is matched efficiently to the documents content in a way to compute their relevance value to the query. Then the system displays the documents retrieved in decreasing order of relevance. A deep exploration can be found in these books: [Christopher D. Manning2008] (chapters ”Boolean retrieval” & “Relevance feedback and query expansion”) and [van Rijsbergen1979] (chapter 1). One key problem is for the user to be able to express a query that reflects his ideas. For this project we’ll not go into details and just scratch the surface of IR (see [Mulhem2021] for a simpler explanation).

For the IRS to work, it needs to have a corpus of documents from where to get the information. We fed Terrier with the TIME collection [TimJune 2021] collection, which consists of articles from the Time magazine.

Application interface

The information retrieval system can change very often, for example if we want to retrieve the results from Google or Bing, the structure is quite different (or even just by changing the version the structure can change). For this reason, we need an intermediary program to get the raw data from the IRS then filter it to only get the useful data in our format (which consists of the title, URL, document Id and description). The application interface is hence in charge to communicate with the Information Retrieval system, it can ask for the results of a query then process them to filter only the useful information. It can then transmit this data to the Dialog Controller. The application interface is also called ”Connector” since it serves as an intermediate for other programs to ask for a query and get the results in a pre-established format.

User Interface

A user interacts with the system via the user interface (see figure 3). It is displayed like a classical search engine result, with the search bar on top and the results in form of snippets with a title, an URL and a description text. The only main difference being that there is a “refine” button on the right of each snippet that the user can click at any point to refine the search via query expansion. This interface is connected to the Dialog controller to ask for queries and get their results, it then transforms the text results into graphic snippets. By doing so, it has knowledge of all the positions of each word, then sends these Zones to the Eye Tracker Analyser. Whenever these zones are updated (scrolling down, asking for new query, change window size etc.), the User Interface will re transmit them to the Eye Tracker Analyser.

Dialog Controller

The central piece of the program is the Dialog Controller. It is the mediator between the User Interface and the Application Interface. After a query is sent and treated, we retrieve the results in form of snippet text. The dialog controller can then proceed to communicate with the User Interface to update it with the new results. The Dialog Controller also receives all the important information from the Eye Tracker Analyser such as the fixations in zones (words on screen), this will be later used for the calculation of indicators.

Calculation of Indicators

Once a user decides to refine a query by clicking on the ”refine” button on the User Interface, the program will look at the words that the user looked at the most in the current snippet (the snippet where the user clicked refine), and adds the one with the highest score (based on the duration of the fixation, the last word looked at etc.). This added term should in theory improve the results of the original search.

Usybus Communication

In this project we call the communication handler ”Usybus” (Usability data Bus). We use Ivy software bus for communication: ”Ivy is a simple protocol and a set of open-source libraries and programs that allows applications to broadcast

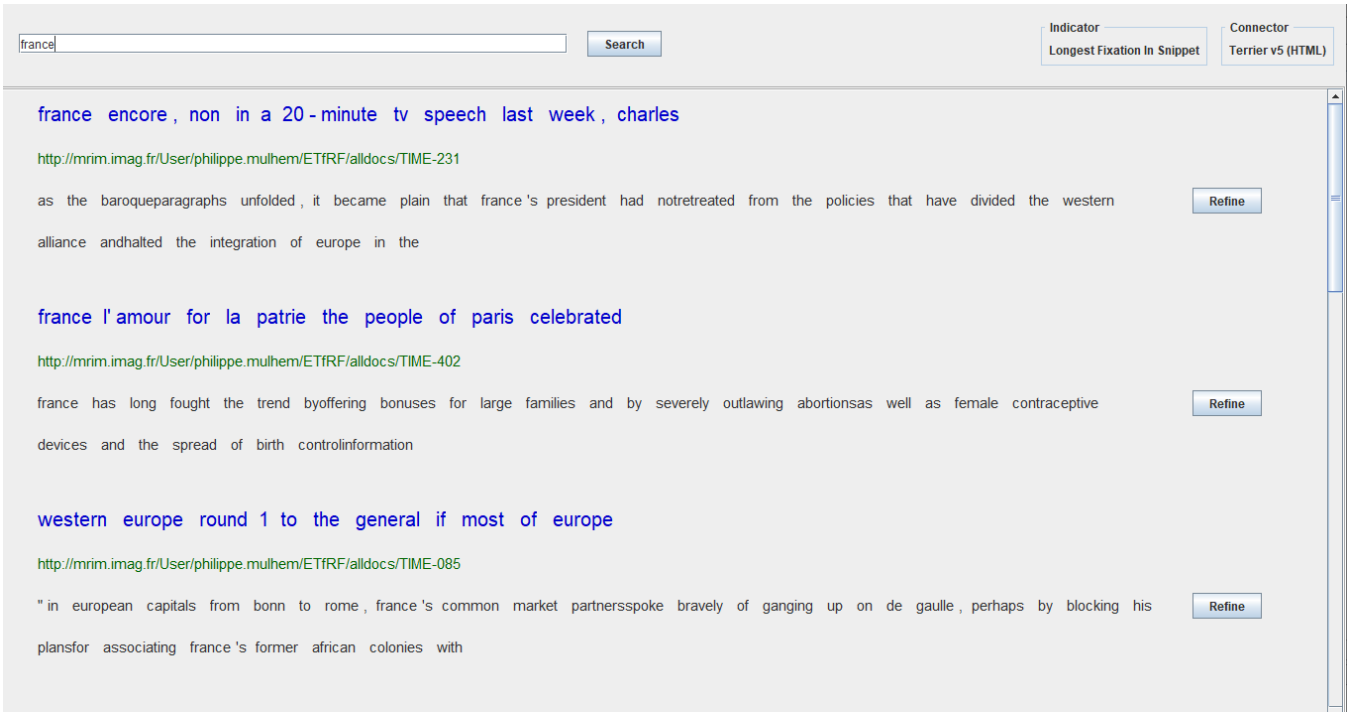

Figure 3: User interface

information through text messages, with a subscription mechanism based on regular expressions" [IvyJune 2021]. It is available in multiple libraries such as C++, Java and Python which is the reason why it was chosen, to provide the tools needed to communicate between the different languages used in this project (Python and Java mainly). The main idea of Usybus is to make an oversimplified communication system using high level functions (connect, disconnect, sendMsg, bindType, etc.) that take care of the low level Ivy formalities. It allows our various modules to be written in different languages as long as they use Usybus to communicate.

3.2 Experiments end Results

For the [Vaynee Sungeelee2020] experiment, the Eye Tribe ET1000 [TheJune 2021] eye tracker was used, its sampling frequency was 30 Hz with an average accuracy between 0.5 to 1 ° of visual angle. The eye tracker was placed on the bottom of the display (similar to figure 5 but with a different eye tracker) and the users were asked to maintain a distance of about 60 cm away from the screen for better gaze detection. Nine participants conducted the experiment; each of them was asked to choose 2 queries out of 3 options, and for each query, search for the snippet they consider the most relevant then click on its refine button (just like one would click on a result in a web search). Among all the results only 7 of them were retained due to calibration issues.

The results obtained in [Vaynee Sungeelee2020] are detailed in Table 2. The scores of each query before expansion are given for P@5, P@10 and Reciprocal Rank followed by their corresponding scores after expansion with the corresponding word (go to 5.3 to see how they're calculated). A (+) sign next to a score means that the expansion has improved its score, whereas a (-) indicates the opposite. A (=) means that the score hasn't changed. The last column tells us if at least one score has improved after expansion.

Looking the results of table2 we can see that 4 out of 7 queries improved after expansion. However not all queries seem to be affected unanimously, the first and third queries have only positive results while the second one only negative. This hints us that not all queries can benefit as much of the query expansion.

4 Implementation

It is important to mention that the eye tracker system used in [Vaynee Sungeelee2020] is now outdated as newer and better eye tracking devices are now available in the market. In the previous work they used "Eye Tribe ET1000" eye tracking device and the new device used for this study is the "Tobii 4C with Pro SDK".

The other main difference is the older version of Terrier v4.0 that is now outdated by v5.4. The old snippet generator used in the previous study was "homemade" . It can be replaced by Terrier v5 embedded snippet generator, easing the reproducibility of our experiences, and so allowing comparison with the literature.

These changes are visualised in green in the architecture (see figure1).

4.1 Tobii 4C Pro SDK Controller

The new Tobii 4C eye tracker has a completely different implementation from the previous device. As the Tobii Pro SDK (software development kit) is not available in Java (the main

language used for this PoC), we had then to use the Python SDK instead. The Python code uses a Tobii module in order to retrieve gaze information from the device.

Python Usybus Implementation

Usybus needs to be implemented in all the programming languages used in order for them to communicate. A Java code for Usybus was already implemented, however the Python implementation was missing. Usybus for Python was then implemented in order for the Controller to communicate with the rest of the Java code. With this we were able to communicate the Eye Tracker gaze data via Usybus.

User Interface

The next step was to create a user interface in order to make communication and transmission of messages easier (as well as connection, disconection, calibration, etc.). To generate the user interface, we used the QtDesigner program as "Qt Designer is the Qt tool for designing and building graphical user interfaces (GUIs) with Qt Widgets" [QtDJune 2021]. With this program, we can create intuitively ".ui" files, then we can use the PyQt5 module for Python to generate the GUI with code and associate each button to an action. The final result can be seen in figure 4

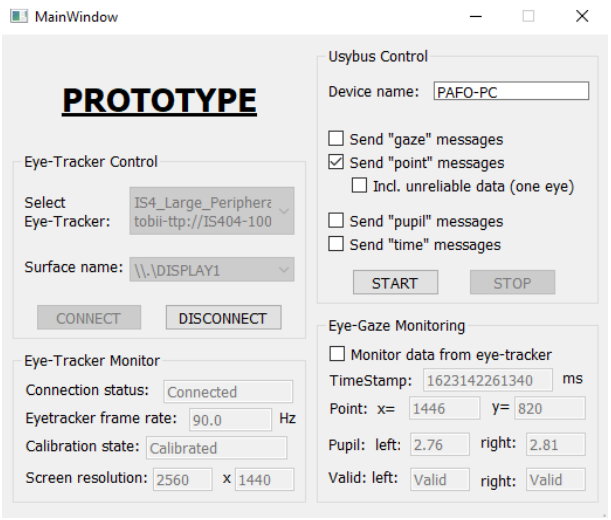

Figure 4: Tobii 4C controller user interface

As can be seen from the image this user interface allows to connect and disconnect a selected eye tracker and also select the screen to which it is attached to (in case of multiple displays). This can be done in the eye tracker control box.

Once the eye tracker is connected we can get useful data such as the connection status, the sample rate, the calibration status, and the screen resolution of the screen.

In the eye gaze monitoring box, we can get the displayed info sent by the eye tracker. By checking the "Monitor data from eye-tracker" case, we can get real time information about the gaze data : timestamp, gaze positions, pupils diameters and data validity flags.

And finally the Usybus control box is the one in charge to communicate the checked info via Usybus (here it will only send "Point" information), it transmits the data with the same frequency as the eye tracker when the button start is pressed, and stops sending data once the stop button is clicked.

4.2 Terrier v5 Connector

In the previous work [Vaynee Sungeelee2020], Terrier v4.0 was used for the search engine. However this needed to be upgraded to Terrier v5.4 newer version. A new code was implemented in order to transform the results given by Terrier v5 to the pre-established Search Engine Research Page structure (title, document Id, URL and description).

Terrier v5 can be used on a web tab just like a normal search engine browser, our code then extracts the useful information from the web source code to retrieve the results of the query. Since the Terrier tab source code is in HTTP language this connector is technically an HTTP source code parser.

5 Experiments and Results

For this new experiment, we tried to be as close as possible to the previous proof of concept [Vaynee Sungeelee2020] with the only changes being the new eye Tracker device and the new information retrieval system.

5.1 Experimental setup

Before each user experiment, the eye tracker first needs to be calibrated via its driver and a direct interaction with the user (this is to take into account the distance to the screen, local illumination etc.), then it is ready to use. Similar to [Vaynee Sungeelee2020], the setup occurred in front of a screen with the Eye Tracker attached in the bottom (figure 5).

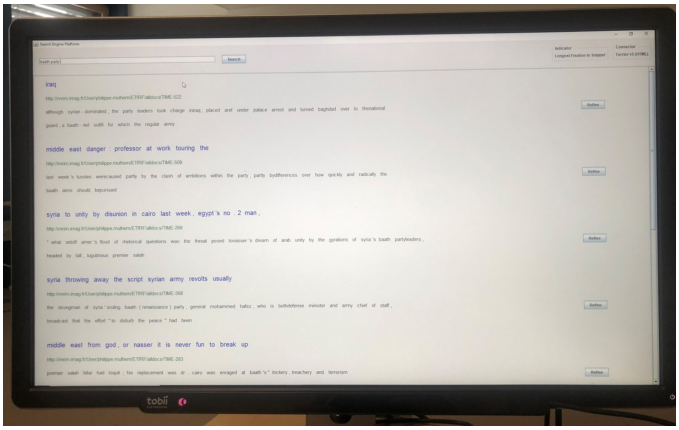

Figure 5: Experimental setup

The Tobii 4C eye tracker gives us information about the validity (e.g. closed or open eye), the point on screen we're looking at and the pupil size for each eye. Given the slightly different information structure given by the new eye tracker compared to the previous one, we only consider the user is looking at a point in screen when both eyes are valid and the "look point" is set to the average of both eyes.

As explained before, the eye tracker used is the Tobii 4C. It has a sampling frequency of 90 Hz, the participants were

about 60 cm away from the screen for optimal detection, two of them wearing glasses (this is important to mention since it's a common problem that eye trackers have a hard time dealing with glasses) and it was done in a closed room illuminated only with artificial light (for better detection).

5.2 Experiment

It is important to mention that due to the global pandemic only three users did the experiment (including myself), and these three people were part of the project so they are biased as they already know the goal of this experiment and how it works.

The experiment was the following; the participant were asked to search three given queries, for each of them, they look at the results until they find one they consider relevant then click on the refine button on the snippet they liked the most. Query expansion is then applied and we can now see how much the query improved dompared to the initial result.

5.3 Results

We present here preliminary experiments conducted on three participants. They were asked to search for three queries, and their gaze was analysed to perform query expansion as described before. The results are present in table 1. In this table, we study three classical Information Retrieval Evaluation measures (P@5, P@10 and reciprocal rank), that respectively reflect the quality of the top-5, top-10 and the position of the first relevant document in the result list.

$$P@k = \frac{number\ of\ relevant\ documents\ among\ the\ first\ k}{k}$$

$$reciprocal\ rank = \frac{1}{rank\ of\ first\ relevant\ document}$$

From table 1, we see that none of our experiment was able to outperform the initial (non expanded) queries, on any of the three evaluation measures considered. These results are then worse than the ones reported in [Vaynee Sungeelee2020] presented in table 2.

We propose some hypotheses to explain these poor results. First, the eye tracker may send low quality data, but this hypothesis is not credible since the device specifications are significantly better than the previous one. Second, the Terrier v5 snippet generator might be worse than the "homemade" snippet generator. This hypothesis is credible and need further investigation. Our experiments do not help us to give an answer to this question yet. Third, another explanation is that the very low number of participants (only 3), and the fact that 2 of them had glasses may alter the efficiency of the words identifications.

Another point worth mentioning is that, as presented in table 2, we do not always obtain similar evaluations than [Vaynee Sungeelee2020] for non-expended queries (scores are different in the "$\emptyset$" row): what changes between the two experiments is the version of the Terrier system, v5.4 in our experiments, and V4.0 in [Vaynee Sungeelee2020]. Further tests have to be achieved to find out the reasons for these differences.

6 Conclusion

In conclusion, the goal of the work described here was to:

- Acquire knowledge on the research topics of Information Retrieval and of Eye Tracking;
- Build a Terrier v5 Connector for the current architecture, which helped the main code to communicate with the new version of Terrier;
- Develop an Usybus module for general communication purposes between any Python program and the rest of the project ;
- Develop a graphic user interface in order to easily and quickly connect a Tobii Pro SDK eye tracker to the rest of the system and retrieve the selected data;
- Run a first experiment using the tools developed.

All these steps have been completed, and even if the preliminary experiments are given bad results (worst than previous work), we are confident about the fact that further tunings will outperform existing results.

So, a goal for short term would be to do more experiments in order to have more results and data to analyse, this would be the most logical next step. As for longer therm another interesting idea would be to explore other search engines such as Google, Qwant, Bing, etc., but with such search engines additional efforts will be needed to tackle the problem of manual assessment of relevant documents.

References

[Buscher *et al.*, 2009] Georg Buscher, Ludger Van Elst, and Andreas Dengel. Segment-level display time as implicit feedback: a comparison to eye tracking. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 67–74. ACM, 2009.

[Christopher D. Manning, 2008] Prabhakar Raghavan Hinrich Schütze Christopher D. Manning. *Introduction to Information Retrieval*. 2008.

[Eickhoff *et al.*, 2015] Carsten Eickhoff, Sebastian Dungs, and Vu Tran. An eye-tracking study of query reformulation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 13–22. ACM, 2015.

[Holmqvist, 2017] Kenneth Holmqvist. *Eye tracking: A comprehensive guide to methods, paradigms, and measures*. Lund Eye-Tracking Research Institute, December 2017.

[Ivy, June 2021] Ivy. `https://www.eei.cena.fr/products/ivy/`, June 2021.

[Mulhem, 2021] Philippe Mulhem. Recherche d'information, fondements. `https://rimiashs.imag.fr/lib/exe/fetch.php?media=fondements_ri_2020-2021.pdf`, 2021.

[Navalpakkam and Churchill, 2018] Vidhya Navalpakkam and Elizabeth F. Churchill. Et-slides, 2018.

[QtD, June 2021] Qtdesigner. https://doc.qt.io/qt-5/qtdesigner-manual.html, June 2021.

[Ter, June 2021] Terrier v5. http://terrier.org/, June 2021.

[The, June 2021] Eyetribe. http://theeyetribe.com, June 2021.

[Tim, June 2021] Time collection. http://ir.dcs.gla.ac.uk/resources/test_collections/time/, June 2021.

[van Rijsbergen, 1979] Keith van Rijsbergen. *Information Retrieval*. 1979.

[Vaynee Sungeelee, 2020] Philippe Mulhem Vaynee Sungeelee, Francis Jambon. Proof of concept and evaluation of eye gaze enhanced relevance feedback in ecological context. October 2020.

Query	Term added	P@5	P@10	Recip_Rank	Overall Improvement
Baath party	∅	0.0000	0.0000	0.0164	-
	lugubrious	0.0000 (=)	0.0000 (=)	0.0164 (=)	No
	rethorical	0.0000 (=)	0.0000 (=)	0.0164 (=)	No
	over	0.0000 (=)	0.0000 (=)	0.0164 (=)	No
U.S. policy toward South Viet Nam	∅	0.0000	0.0000	0.0159	-
	optimism	0.0000 (=)	0.0000 (=)	0.0152 (-)	No
	saigon	0.0000 (=)	0.0000 (=)	0.0154 (-)	No
	concrete	0.0000 (=)	0.0000 (=)	0.0137 (-)	No
Ceremonial suicides of buddhists monks	∅	0.2000	0.1000	0.333	-
	protest (PM)	0.0000 (-)	0.1000 (=)	0.1667 (-)	No
	macabre	0.2000 (=)	0.1000 (=)	0.333 (=)	No

Table 1: Results with Precision@5 and Precision@10 scores, reciprocal rank before/after refinements for our work.

Query	Term added	P@5	P@10	Recip_Rank	Overall Improvement
Baath party	∅	0.0000	0.0000	0.0164	-
	settle	0.0000 (=)	0.0000 (=)	0.0244 (+)	Yes
	self-isolation	0.0000 (=)	0.0000 (=)	0.0227 (+)	Yes
U.S. policy toward South Viet Nam	∅	0.0000	0.1000	0.1429	-
	conference	0.0000 (=)	0.1000 (=)	0.1000 (-)	No
	Military	0.0000 (=)	0.0000 (-)	0.0714 (-)	No
	misinformed	0.0000 (=)	0.1000 (=)	0.1429 (=)	No
Ceremonial suicides of buddhists monks	∅	0.0000	0.0000	0.0227	-
	automobile	0.2000 (+)	0.1000 (+)	0.3333 (+)	Yes
	school	0.2000 (+)	0.1000 (+)	0.2000 (+)	Yes

Table 2: Results with Precision@5 and Precision@10 scores, reciprocal rank before/after refinements from [Vaynee Sungeelee2020]