User profiling

Group 3: Anton Lu, Björn Lindqvist, Chris Lin, Sergio Liberman

Background and related methods

Modern commercial search engines like Google compute search results to a given query not just based on measures like tf-idf and PageRank, but also on the user's past behavior. In particular, what links users have selected in preceding searches will heavily influence the ranking of the results of new searches.

In this project we have explored and implemented methods to manipulate search rankings based on past search history of individual people.

Methods proposed by others include:

Personalised PageRank

The method adjusts the random surfer model so the surfer is biased toward some pages, such as those found in the users history and bookmarks. In this way, each user has their own set of unique PageRank scores. [1]
Another method is to assign each page a categorial PageRank score, and use the score from a specific category for ranking, depending on the topics related to the user's query. [2]

Session-based user profiling

Shen, Tan and Zhai [3] proposed categorizing user preferences as either ephermal or long-lasting. Focusing on ephermal preferences, they developed a Bayesian model in which each query result would be affected by the previous ones in the same session. In this way, given a predecessor query *cgi programming*, the query *java* is more likely to be about programming

User embeddings

Representing user preferences as vectors and having their similarities with document vectors influence search rankings have been rigorously explored.
User representations have previously been explored with both standard tf-idf embeddings [4,5], as well as representing users and documents with topic vectors, trained using deep learning methods. [6]

Problem statement

In this project we assumed that we were able to record the queries individuals input into a search engine, as well as which pages the user clicks from the search results. We also presumed that the user had a social profile, representing the user's personal preferences.

Using this information, the goal was construct an user profile, and to over time influence the user's search results, so that results more aligned with the user's immediate profile, as well as previous searches were more highly ranked than others.

Methods

I. User and document representation

Each user profile is represented with three document profiles in tf-idf-space:

$\boldsymbol{u_t}$: *Terms of document titles clicked on*
$\boldsymbol{u_c}$: *Terms of document categories clicked on*
$\boldsymbol{u_x}$: *Terms of document content clicked on*

Final user and document vectors $\boldsymbol{u}$ and $\boldsymbol{d_i}$ are calculated as

$$\boldsymbol{u} = \frac{1}{Z_u}[w_t\boldsymbol{u_t} + w_c\boldsymbol{u_c} + w_x\boldsymbol{u_x}]$$
$$\boldsymbol{d_i} = \frac{1}{Z_d}[w_t\boldsymbol{t_i} + w_c\boldsymbol{c_i} + w_x\boldsymbol{x_i}]$$

Where the Z are normalization factors.

II. Query expansion

Using the idea of the Rocchio algorithm we enhance queries input to the system, $\boldsymbol{q}$, using the modified user profile vector $\tilde{\boldsymbol{u}}$ as

$$\boldsymbol{q}_{new} = \boldsymbol{q} + \gamma\tilde{\boldsymbol{u}}$$

Where the user profile vector $\boldsymbol{u}$ has been pruned so that terms with small weighting are removed to improve system performance. Terms that already exist in $\boldsymbol{q}$ were also removed from $\boldsymbol{u}$ to prevent overemphasizing certain terms. In this case, $\gamma \in [0,1]$.

III. Re-ranking strategy

Re-ranking of search results is done by fetching documents from the search engine using ranked search for the active query, and then re-ranking them using the user profile. For each document $\boldsymbol{d_i}$ and original score s_i, its final score is calculated as

$$r_i = \alpha s_i + (1-\alpha)\boldsymbol{u}^T \boldsymbol{d_i}$$

Where $\alpha \in [0,1]$ controls the influence of the user profile on the final ranking.

IV. Adaptive profile updating

To emulate concentration shift and forgetting, we employed an exponential decaying scheme. The user profile vectors are then functions of time, defined as

$$\boldsymbol{u_k}(t) = \boldsymbol{u_k}(t')e^{-\lambda(t-t')}$$

Where k in this cases represents title, category, content, as presented in (I). The exponential decay constant λ is set to be different for different kinds of profile vectors For instance, a real-world user is more likely to remember the title of an article rather than the text content, so the decay factor of the title vector is smaller than that of content. When the user clicks on a new page on the results page, the old profile vector is decayed, and then added to the new document vector of the page clicked.

V. Static profile vector

To complement the dynamically updated user profile vector $\boldsymbol{u_d}$ as seen in (I), one can also introduce a static profile vector $\boldsymbol{u_s}$ representing long-lasting user preferences, to simulate social profiles. The user profile vector used in (II), (III) is then instead defined as

$$\boldsymbol{u} = \beta \boldsymbol{u_d} + (1-\beta)\boldsymbol{u_s}$$

Which means that the static preferences do not decay over time as in (IV).

Implementation and system setup

For this task we used a 4 vCPU Google Cloud VM with ElasticSearch 6.3.1, and indexed the whole Swedish Wikipedia dump dated 2019-04-29 as search corpus. The dump contained 11,548,339 documents.

To present search results to users, a website frontend was built using Materialize and Vue.js with a Python Flask backend.

Individual user social profiles were simulated by letting the user input age, sex, location, and selected keywords into the web interface, which were stored in the persistent user database as a static user profile.

Figure 2: overview of the structure of our implementation.

When an user clicks a search result, the dynamic user profile is updated according to (IV) using the document vectors of the page and (I), and in the case when a static profile exists, it is also used to produce the final user vector.

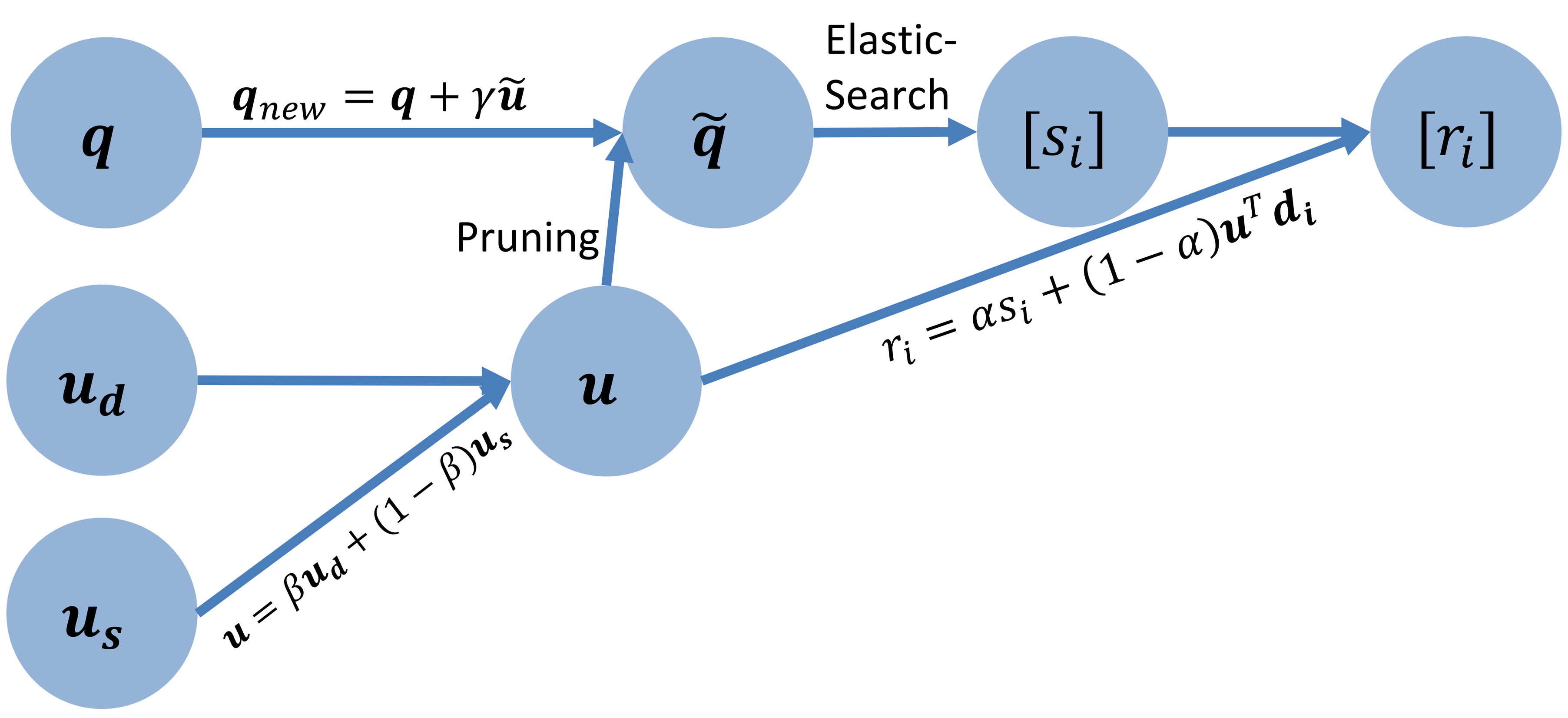


Figure 1: Simplified computational graph showing the process from query input $\boldsymbol{q}$ to final ranking scores r_i.

Evaluation

Queries were issued to the search engine, and after n number of searches, the quality of the profile-based search result were compared to that of a blank profile search.

The metric used for evaluation was the Normalized Discounted Cumulative Gain (nDCG) at position p, defined as

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

Where

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

$$IDCG_p = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

and $|REL|$ represents a list of the p top documents, and rel_i is a relevancy score from 0 to 3.

We also used precision as metric for search result goodness.

Experiment 1

In the first evaluation, we ran test queries for 5 blank user profiles, and computed their $nDCG_{20}$. Then the profiles were being updated by clicking 10 relevant and 10 irrelevant documents, and observing what effect they had on the search results. The performance after each click was measured using $nDCG_{20}$. The documents were re-ranked with $\alpha = 0.7$, and the top 100 terms in the user profile were used to expand the search query. The static profile was disabled in this experiment, and experiment 2.

Experiment 2

To test sensitivity to query expansion, another experiment was performed by setting $\alpha = 0$ (no query expansion), and using the top 500 terms in the user profile to re-rank search results.

Experiment 3

A second experiment was run to investigate the impact for weightings between the static and dynamic user profile. We assigned an user a static profiles with somewhat unrelated keywords, and ran several queries related to “Japan”, and evaluated the performance of the search engine to help the user for example find information about sightseeing points in Japan.

Experiment 1 results

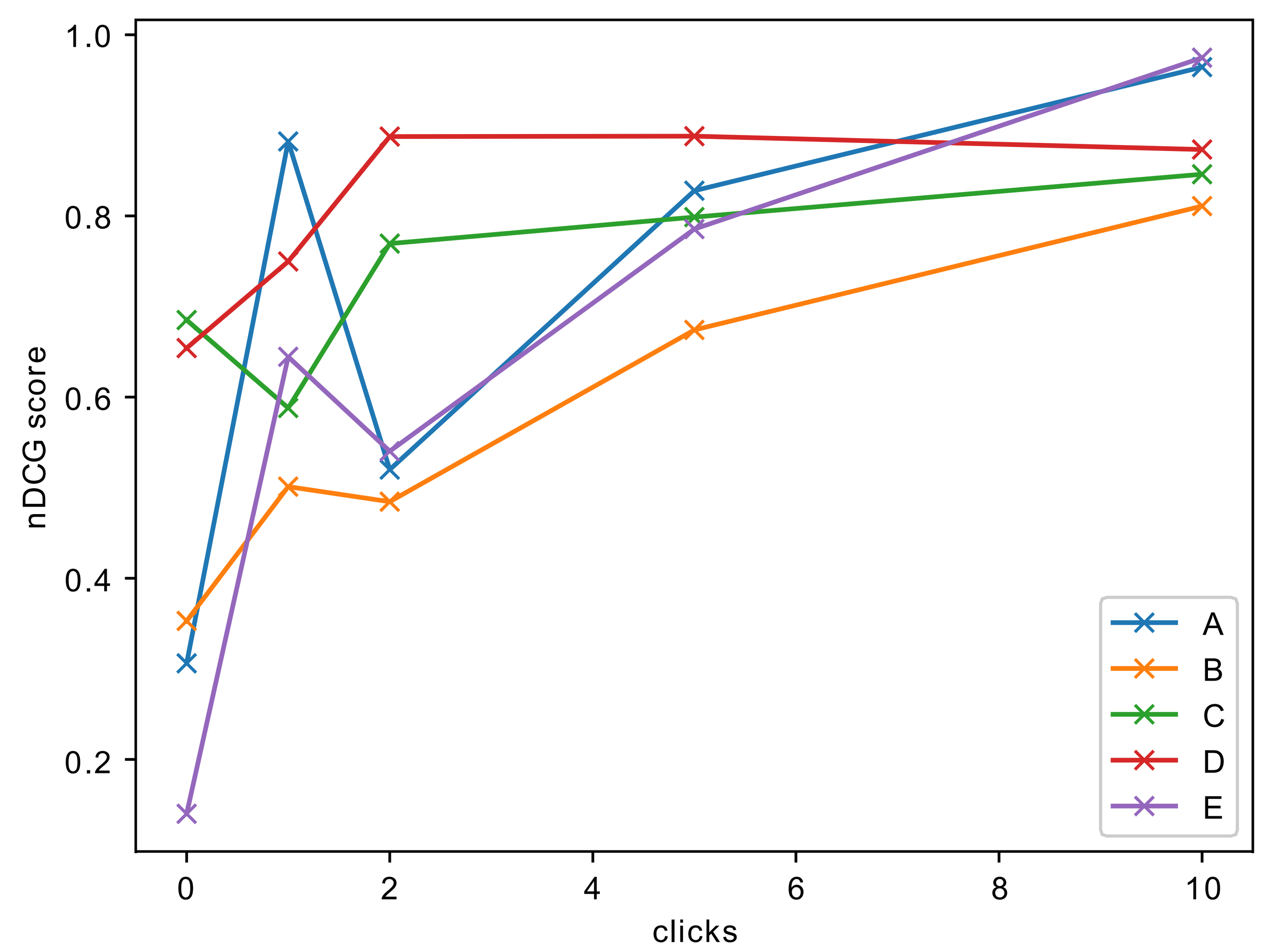


Figure 3: Experiment 1, $nDCG$ scores for 5 different users as function of number of clicks (implicit feedback). New queries were posed between clicks.

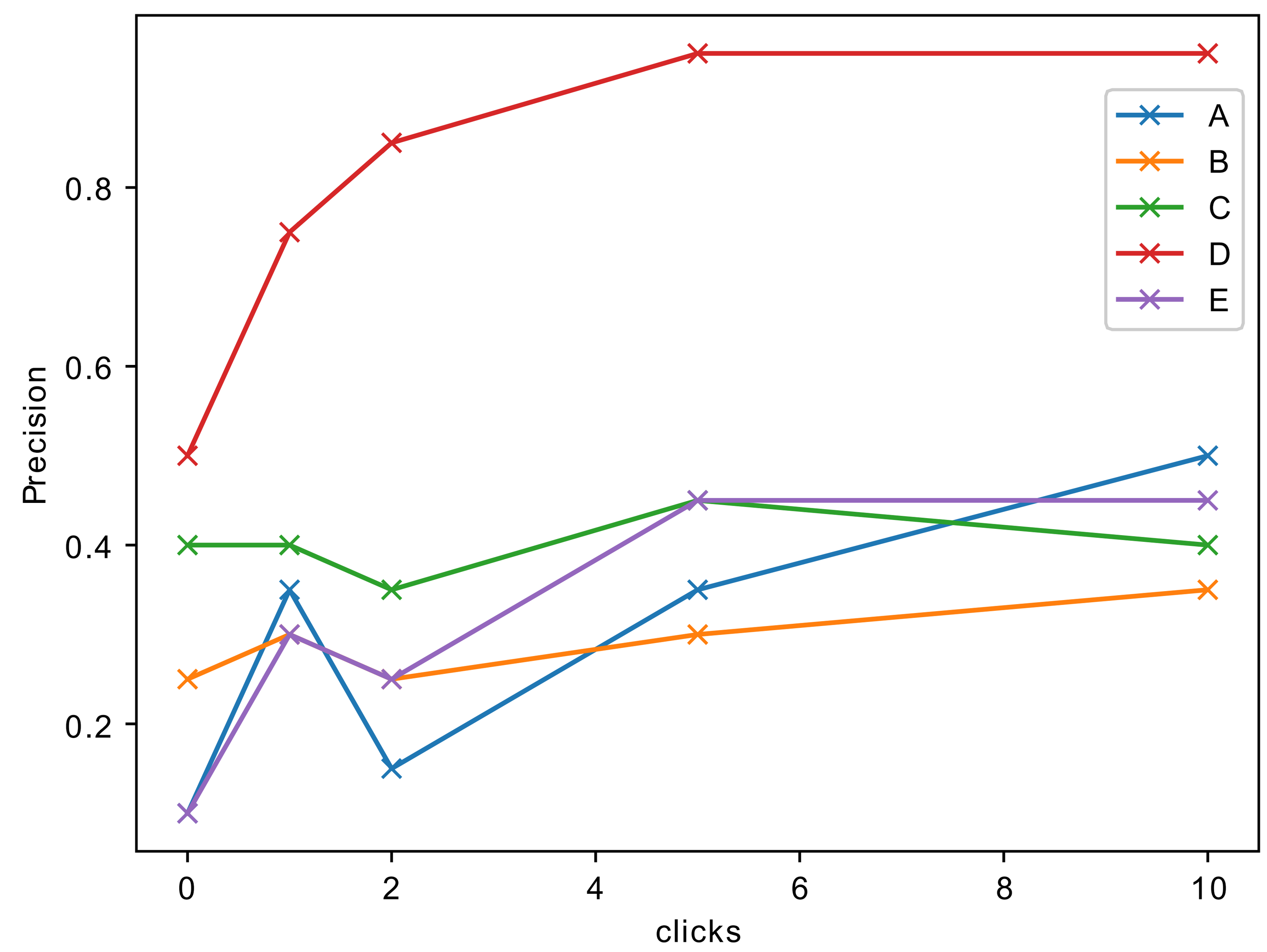


Figure 4: Experiment 1, Precision scores for 5 different users as function of number of clicks (implicit feedback). New queries were posed between clicks.

Figure 5: Experiment 1, Word cloud displaying the user profile vector for user A after 10 clicks.

Experiment 2 results

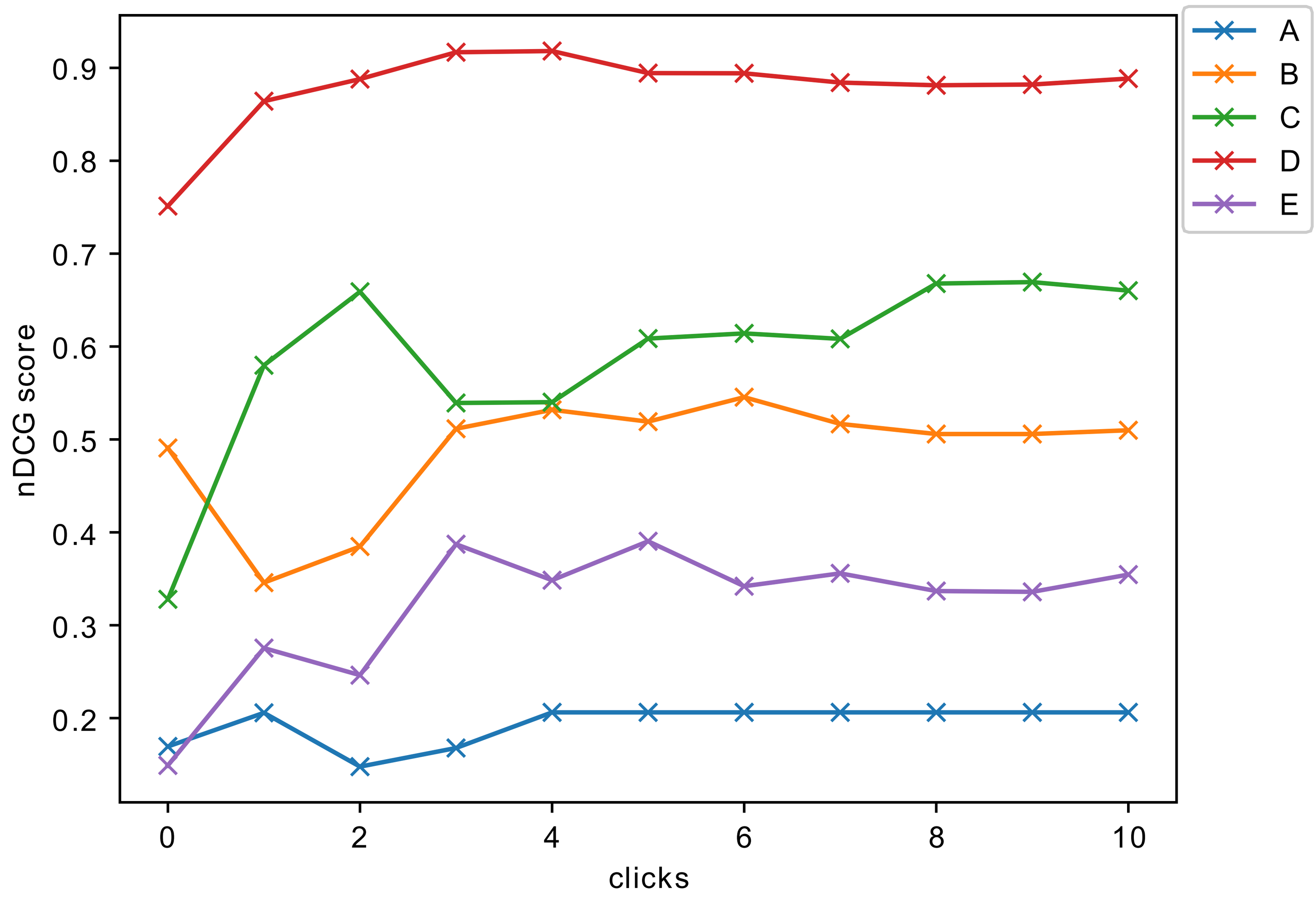


Figure 6: Experiment 2, $nDCG$ scores for 5 different users as function of number of clicks (implicit feedback). New queries were posed between clicks.

For most personas in both variants experiments 1 | 2 the relevancy of the search results increased a lot. Even when not using query expansion, search performance still improved, albeit not as much as in experiment 1. Relying only on re-ranking can still be useful as it does not impact the ranking of non-related queries as much.

Experiment 3 results

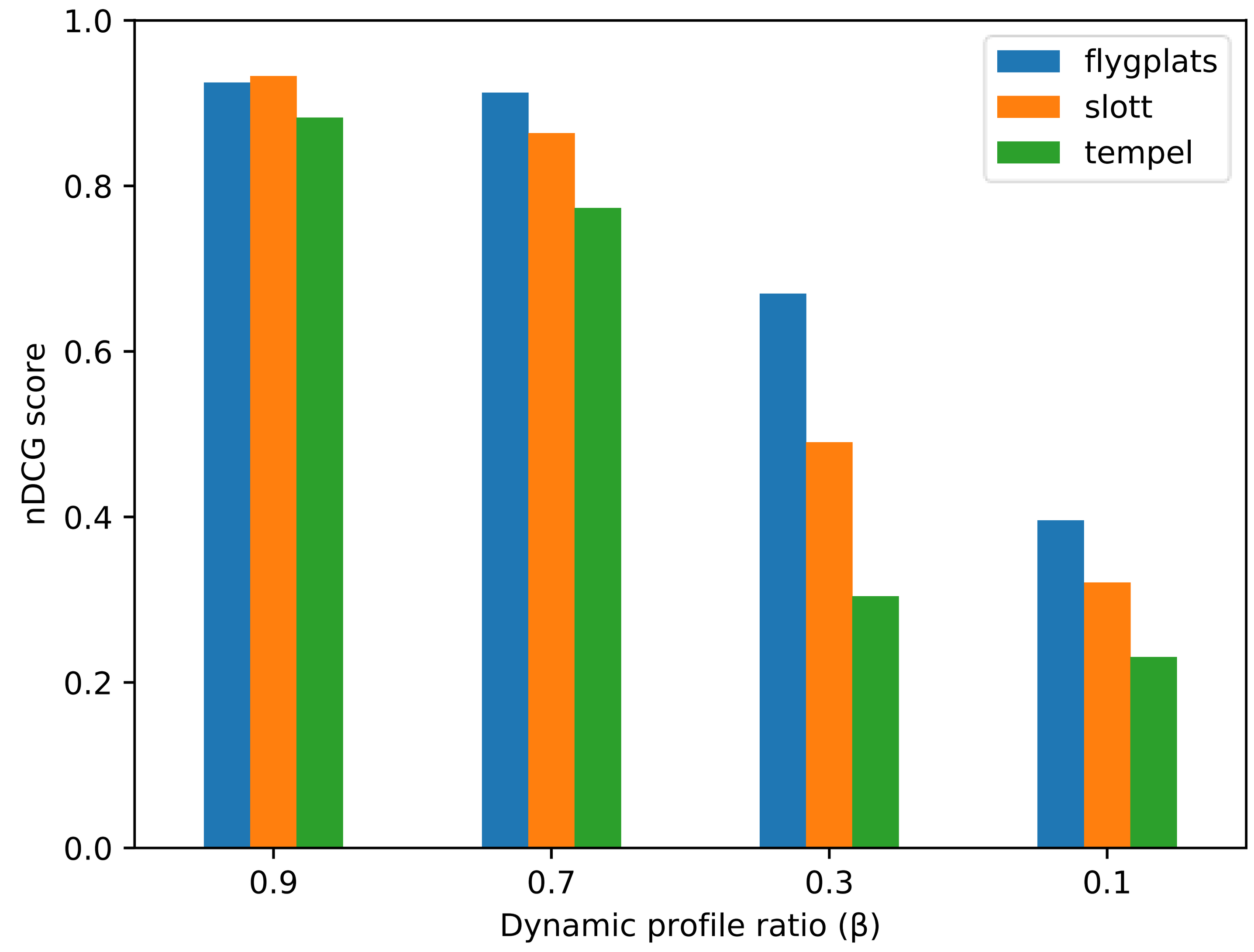


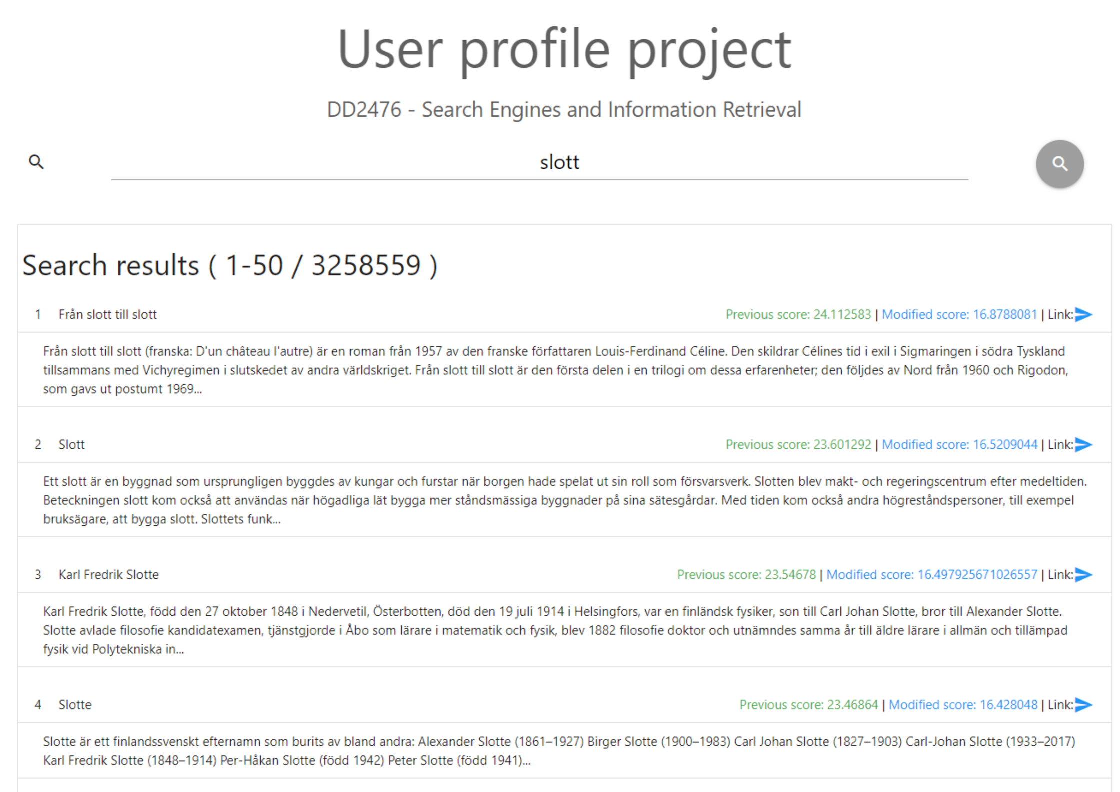
Figure 7: Experiment 3, $nDCG$ scores for different values of β (Method V), issuing the same queries for the same 4 experiments.

The result shows that the dynamic profile portion (large β) is excellent at capturing the user temporary profile. The more portion of the dynamic profile, the better the capture of ephemeral preferences.

Evolution of an user profile

Here we illustrate how an user profile evolves as an user uses the search engine

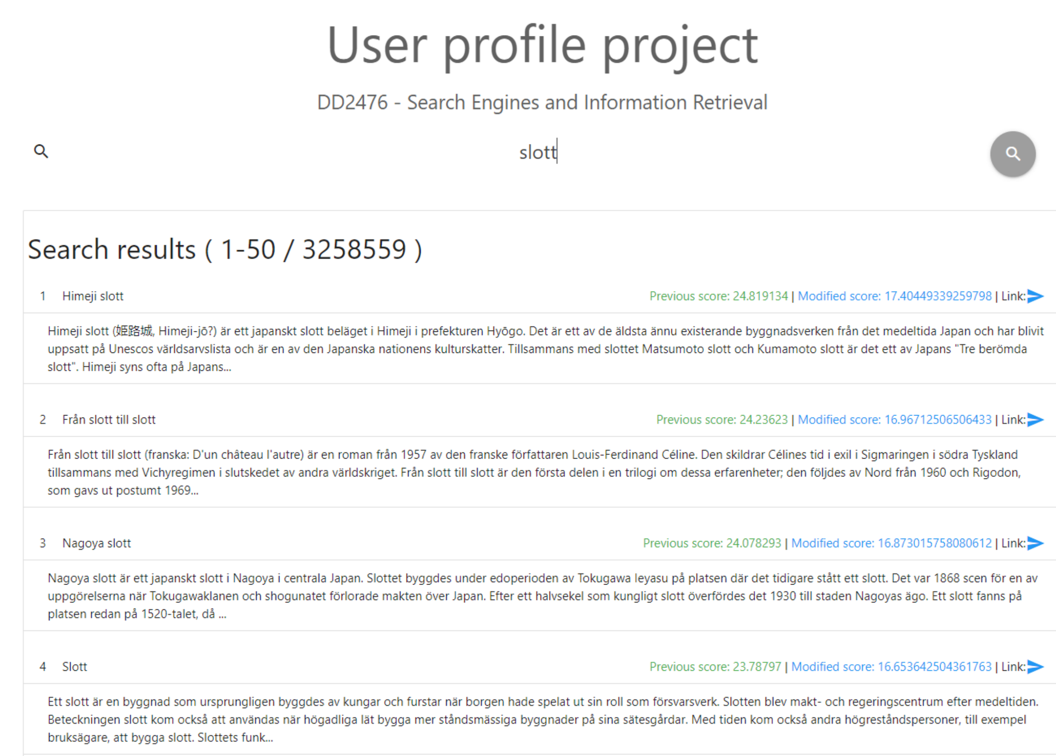
Blank profile with static keywords



Some intermediate queries

- Japanska
- Osaka
- Himeji

Updated profile



References

1. Larry Page et al. The PageRank Citation Ranking: Bringing Order to the Web. 1998
2. Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. “Adaptive Web Search Based on User Profile Constructed Without Any Effort from Users”. In: Proceedings of the 13th International Conference on World Wide Web. WWW ’04. New York, NY, USA: ACM, 2004, pp. 675–684.
3. Xuehua Shen, Bin Tan, and ChengXiang Zhai. “Context-sensitive Information Retrieval Using Implicit Feedback”. In: Proceedings of the 28th Annual International ACM SIGIR Con-ference on Research and Development in Infor-mation Retrieval. SIGIR ’05. Salvador, Brazil: ACM, 2005, pp. 43–50.
4. Nicolaas Matthijs and Filip Radlinski. “Personalizing Web Search Using Long Term Browsing History”. In: Proceedings of theFourth ACM International Conference on Web Search and Data Mining. WSDM ’11. HongKong, China: ACM, 2011, pp. 25–34.
5. Shengliang Xu et al. “Exploring Folksonomy for Personalized Search”. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR ’08. Singapore, Singapore: ACM, 2008, pp. 155–162.
6. Thanh Vu et al. “Search Personalization with Embeddings”. In: CoRRabs/1612.03597(2016). arXiv:1612.03597.