Reranking Search Engine Results Using a User Profile

Report for Group Project in DD2476: "Search Engines and Information Retrieval", KTH.

Steven Gerick, gerick@kth.se
Gudjon Ragnar Brynjarsson, grbr@kth.se
Zehua Chen, zehuac@kth.se

KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science

18.05.2018

Contents

Abstract

In this paper, we specify and build a user history database for the purpose of reranking the results returned by a search engine. We describe two proposed methods of re-ranking search results using these records, as well as a combined method, for several queries and evaluate the results using Discounted Cumulative Gain. Our experiments are performed on the corpus of Simple English Wikipedia pages. We find that all methods lead to a variable increase in score, the effects of which are especially pronounced for ambiguous queries.

Introduction

Modern search engines have become quite good at presenting relevant results. However, they often fail when presented with homographs (words having the same spelling but different meanings) and ambiguous names. This is a problem inherent to the structure of the query: we can't always assume that the query contains the full information needed to answer the user's request. One way to present more relevant results is to guess the user's intentions. This can be a difficult task and requires an understand about the user, usually constructed from user data. This info can be quite diverse, including information as nuanced as a complete browsing and search history along with a network traversal record, but in practice most of this data is irrelevant for any given query. In practice, only user data regarding the issued query is relevant to the search engine. In this short report we exhibit a search engine that tracks user page visits and the categories of those pages, and uses a user profile built from this history to re-rank the top results after initial ranking by the search engine Elasticsearch. We test two methods, one based on category hitcounts (our guess as to the user's interests) and one based on search history and click-through. The hope is that the method based on interests will make ranking across queries better while for the second method we would like to see that for repeatedly queried searches, the previously relevant documents get a higher rank. The actually performance improvement can be measured with Discounted Cumulative Gain, or DCG.

Related work

Use of user information to improve search capabilities has been researched quite a bit, since the "query-only" model of user desire modelling fails in quite a few different scenarios. Much of the current research is on how to work with user search logs.

One such method[3] proposes the use of query extensions and adjacent queries in the user's history. Since most people often formulate their searches a few times to try for better results by searching for something more specific or altering their search altogether, we can often infer what information users want based on which queries are associated with their current query.

Other methods use not only the search logs but also other things like clickthrough rates related to queries[1]. One method [1] is to use similar queries to the one posed to the search engines to boost relevant documents, either by building a query index over similar queries – requiring a training set of queries and clickthroughs to build similarity – or by using a synonym dictionary to expand the results. Recent years have also seen a surge in the use of machine learning techniques to do user-based re-ranking. [5] provides a good summary of some recent work of this nature.

Ranking methods

When performing a query, we retrieve initial ranked results using *Elasticsearch* , which ranks documents using a Lucene scoring function. In this paper we will perform re-ranking of the top results using two methods, as well as a combination of the two. We will then compare the results using the discounted cumulative gain model, where documents add more score to performance both by being rated more relevant and by being included higher in the list of results. In this section we will look at the two methods we used.

3.1 Data & datastructures

For building user profiles we need to either determine user interests by tracking their behavior or by building an *a priori* user model. Our engine performs the first option, inferring interests of the user from the categories associated with their visited pages and their search history. The database queried for information is the text corpus of the Simple English Wikipedia. This database is identical in format to Wikipedia, but is written in English at a non-technical level. As an encyclopedia, it contains highly relevant information in a very small number of pages for a large number of academic topics. This means that the number of relevant documents for most reasonable queries is very small, usually not exceeding three to five. The September 2017 snapshot of the database we used contains only roughly 182 thousand main-namespace articles totalling only about 350 MB of text, which means it is much more feasible to work with than – for example – the English Wikipedia with 60 GB of text spread across about 5.5 million articles. Each page has been stripped down to only its title, id number, text (including tables and templates) and list of categories.

The structure of the article items indexed into *Elasticsearch* is straightforward:

```
{
    "title": articleTitle,
    "id": number,
    "text": articleBody,
    "categories": [cat1, cat2,...]
}
```

When searching we will be searching in the *text* field only. Additionally we created a user index to hold the user info:

```
{
    "id": userID,
    "preferences": {
        cat1: NumOfVisits,
        cat2: NumOfVisits,...
    },
    "search_history": {
        query1: [visited documents],
        query2: [visited documents],...
    }
}
```

3.2 Re-ranking using categories

The dataset we use is a collection of articles from the simple English Wikipedia as previously mentioned. These articles have assigned categories. Our thought was that using these categories we could boost the score of articles that are assigned to categories that the user is interested in, and rank higher the articles that may be more relevant to the user.

To do this we make a note of every category that a viewed article is assigned to. We then increase the number of visits to these categories, in the user profile of the current user, by one. This will mean that over time the categories that the user is most interested in will get to have a high score in their user profile. To make use of these scores we present the following method.

Given all documents returned by the search engine we pick the top N scoring documents D. We start by collecting all the categories C_D that documents in D are assigned to. Using these categories we then construct category vectors for all of the articles. These article category vectors will be binary vectors of length $|C_D|$ with ones in places for categories that the document is assigned to e.g. a document $d \in D$ with categories $k \in C_d$ will have a vector cv_d s.t. $cv_{d,i} = 1/|C_d|$ if $i \in C_d$ but otherwise 0. Similarly we create a category vector for the user where we use the category visits previously mentioned by picking out the visitation values for the categories in C_D putting them in the correct places in cv_u and normalizing. Using these article category vectors and user category vector we calculate the cosine similarity of the user category vector and each article category vector. Finally we re-rank the results with the following formula:

$$\text{Score}_{rerank}(d) = \alpha \cdot \text{Score}_{original}(d) + (1 - \alpha) \cdot \text{CosineSim}_{cv_u, cv_d}$$

The parameter α can be tuned to place more weight on the original score (higher α) or more weight on the cosine similarity in the categories (lowerα). This calculation exploits the sparsity of the page category markers and the user's category list, meaning we do not need to find an enormous dot product including mostly zero terms for all possible categories in the database. If a user has a lot of visits to certain categories they will have a high score in the user category vector, and if the category is present in the document we will have a higher cosine similarity, leading to a larger increase in score for these categories.

3.3 Clickthrough-boosted baseline

Our second method is to include clickthrough rates for queries in a simple fashion to boost the score output of the search engine. Basically, we want boost articles that the user visited in previous sessions with the same query [1]. This model works well when the result previews are informative enough to assume that the user only clicks on relevant documents. We also want to boost those articles that are visited more frequently by the user. To do this we used the Clickthrough-boosted baseline ranking proposed in (Aktolga and Allen) [1]. The formula for the re-rank of document d given query q is:

$$P_{BoosLuc}(d|q) = \gamma\frac{c_{CT}(q,d)}{c_{CT}(q)} + (1-\gamma)P_{Luc}(d|q)$$

where

$$P_{Luc}(d|q) = \frac{score_{Luc}(q|d)}{\sum_i score_{Luc}(q,d_i)}$$

$$\gamma = \frac{c_{CT}(q)}{c_{CT}(q)+\rho}$$

In the formula above, $score_{Luc}(q|d)$ is the original tf-idf score which we can get directly from the Elasticsearch, $c_{CT}(q,d)$ is the clickthrough rate of document d from query q and $c_{CT}(q)$ is the clickthrough rate of query q. $P_{Luc}(d|q)$ is the normalized Lucene score of the returned documents. ρ is an empirical parameter which can be determined by experiments on the training data.

3.4 Combination

We also tried combining the two scoring methods. In the combination method, we first rerank the original score using category reranking, and then feed this score as the original score to the Lucene clickthrough-boosting method for a second rerank.

Comparing methods

Now we try to do some experiments on both the algorithms and the combined version of them. We issued some specific queries and manually score each search results on a scale from 0 to 5 with increasing relevance to the query issued.

We measured the performance of the search engine in several metrics, including precision, recall and the discounted cumulative gain metric.

$$DGG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2(i+1)}$$

4.1 Query: Fencing

Here we compare the ranking for the query ”Fencing” hoping to find the page called Fencing. We use the following user-profile:

```
{
  "id":5,
  "preferences": {"Fencing":7, "Agriculture":1, "Swords":3}
  ,
  "search_history": {"Fencing": ["28037", "28037"]}
}
```

The user has some relevant preferences like ”Fencing” and ”Swords” as well as ”Agriculture” which is not relevant. We put ”Swords” as a category since it is closely linked to the primary category of interest and thus makes it more realistic. The ID in the search_history clickthrough for the ”Fencing” query is the ID for the wanted article.

We can see in tables 4.1 to 4.4 the difference in ranking between the methods. The articles given relevance 1 are articles for famous fencers so they have some relevance but quite low still. When using the preferences of the user to re-rank (table 4.2) we see much increase because of the prevalence of ”Fencing” and ”Sword” categories. Furthermore we see that ”Agricultural fencing” outranks ”Imre Gedvari”. This could be considered an adverse affect of the fact that the articles about fencers do not have any of the specified categories and that one of the user preferences is ”Agriculture”. This affect could be made smaller by changing the value of *alpha* in the re-ranking algorithm. That will however also affect the boost given to articles of interest.

Looking at table 4.3 we see that by using the clickthrough of the articles by search terms we can boost the ”Fencing” article up to first place. The difference being that the ”Sabre” and ”Swordsmanship” articles are not boosted. If we add the ID for ”Lyudmila Shishova” to the list of clickthroughs it shoots to the top because of the difference in the original ranking. However by adding one more click to the ”Fencing” article it will again be on top. This seems quite volatile but can

Article	Relevance
Imre Gedvri	1
Lyudmila Shishova	1
Agricultural fencing	0
Victor Gaureanu	1
Hyun Hee	1
DCG-score	2

Table 4.1: Default

Article	Relevance
Fencing	5
Sabre	4
Swordsmanship	3
Agricultural fencing	0
Imre Gedvri	1
DCG-score	10

Table 4.2: Category reranking

Article	Relevance
Fencing	5
Imre Gedvri	1
Lyudmila Shishova	1
Agricultural fencing	0
Victor Gaureanu	1
DCG-score	6

Table 4.3: Lucene Boosting

Article	Relevance
Fencing	5
Sabre	4
Swordsmanship	3
Agricultural fencing	0
Imre Gedvri	1
DCG-score	10

Table 4.4: Both methods

be smoothed by increasing the ρ parameter. That would also mean that fewer clicks do not increase the rank of visited articles as dramatically, meaning that only pages viewed many times will increase the rank substantially which could be though as adverse. Furthermore the mixing of the two methods seems to give no more benefits than the category-only based ranking. This is because of the small size of this example. In bigger examples we could expect a bigger impact.

4.2 Query: Fields

For the second example, we will be searching for information on the Fields Medal, a medal in mathematics often referred to as the ”Nobel Prize in Mathematics”. The query itself is the single word ”fields” and the user has the following history:

```
{
  "id":5,
  "preferences": {"Physics":10, "Mathematics":8, "Algebra":
   3, "Living people":10, "Linear Algebra":6, "Computer
   Science":2, "Agriculture":1, "Wine":1},
  "search_history": {"fields":["38008"]}
}
```

Some of the category hits here (Living People, Agriculture, Wine) were selected because they are reasonable categories to appear as noise in the user profile (Most people will have significant visits to pages for living people, but this query does not concern a living person). The results are as follows:

As in the previous example, we see that category reranking alone dramatically boosts the DCG score. In this case, both relevant articles (possibly the only relevant articles in the entire database) appear at the top of the list, although not in the

Article	Relevance
London Fields railway station	0
Gracie Fields	0
Field (physics)	0
Galois field	0
Rich Fields	0
DCG-score	0

Table 4.5: Default

Article	Relevance
Distribution (Mathematics)	3
Fields Medal	5
Galois field	0
Rich Fields	0
Flurbereinigung	0
DCG-score	6.15

Table 4.6: Category reranking

Article	Relevance
Fields Medal	5
London Fields railway station	0
Gracie Fields	0
Field (physics)	0
Galois field	0
DCG-score	5

Table 4.7: Lucene Boosting

Article	Relevance
Fields Medal	5
Distribution (Mathematics)	3
Galois field	0
Rich Fields	0
Flurbereinigung	0
DCG-score	6.89

Table 4.8: Both methods

preferred order. This is because although both articles are tagged with the ”mathematics” category which has heavy weight in the user profile, the Fields Medal has an additional ”Science awards” category which reduces the cosine similarity score to the User’s profile. Lucene boosting shows the same effects as before, where the results are identical except for the visited pages being dramatically boosted to the top of the list. The mixed method also shows a similar effect to the Lucene boosting only; it is identical to the category only reranking but with the visited pages boosted to the top of the list.

4.3 Query: Churchill

For our final example, we will attempt to find information on Winston Churchill using only the query ”Churchill”, which would be very ambiguous if we only have access to the query text itself. The user profile is now:

```
{
  "id":5,
  "preferences": {"Disease-related deaths in England":3, "
   Living people":10, "English Dukes":6, "Time People of
   the Year":7, "World War II political leaders":5, "
   British people of World War II":6},
  "search_history": {"churchill": ["8471"]}
}
```

As before, the profile is heavily biased in favor of categories related to the desired information (in this case, the categories of British World War II people and political leaders during World War II), and categories present in the top 10 articles of the default search results have been used to construct noise that such a user might have in their profile ("Living People" as expected for most users, and in this case the particularly challenging noise of british nobility and deaths in England)

Article	Relevance
John Spencer-Churchill, 11th...	0
Duke of Marlborough	0
Churchill County School District	0
Clarissa Eden	1
Mary Soames, Baroness Soames	1
DCG-score	0.817

Table 4.9: Default

Article	Relevance
Duke of Marlborough	0
Clarissa Eden	1
Dan Ito	0
John Churchill, 1st Duke...	0
Winston Churchill	5
DCG-score	2.57

Table 4.10: Category reranking

Article	Relevance
Winston Churchill	5
John Spencer-Churchill, 11th...	0
Duke of Marlborough	0
Churchill County School District	0
Clarissa Eden	1
DCG-score	5.39

Table 4.11: Lucene Boosting

Article	Relevance
Winston Churchill	5
Duke of Marlborough	0
Clarissa Eden	1
Dan Ito	0
John Churchill, 1st Duke...	0
DCG-score	5.5

Table 4.12: Both methods

Again, we see that the desired main article (Winston Churchill) is always displayed in first place using a method involving lucene boosting of the query history. The category reranking, however, does a much poorer job for this query, boosting Winston Churchill only from a default rank of 8 to rank 5. This is because the Winston Churchill page is tagged with a large number of categories, most of which have no hits in the user's profile, resulting in a smaller boost to score than otherwise expected.

The last example draws attention to flaws in our use of cosine similarity to rerank using user categories. When a page has a large number of tagged categories, it is effectively penalized for every category the user has not visited. A different metric, such as a cosine similarity only considering categories which have nonzero hitcount in the user's profile, would create more desirable behavior for this situation, but possibly reduce performance elsewhere.

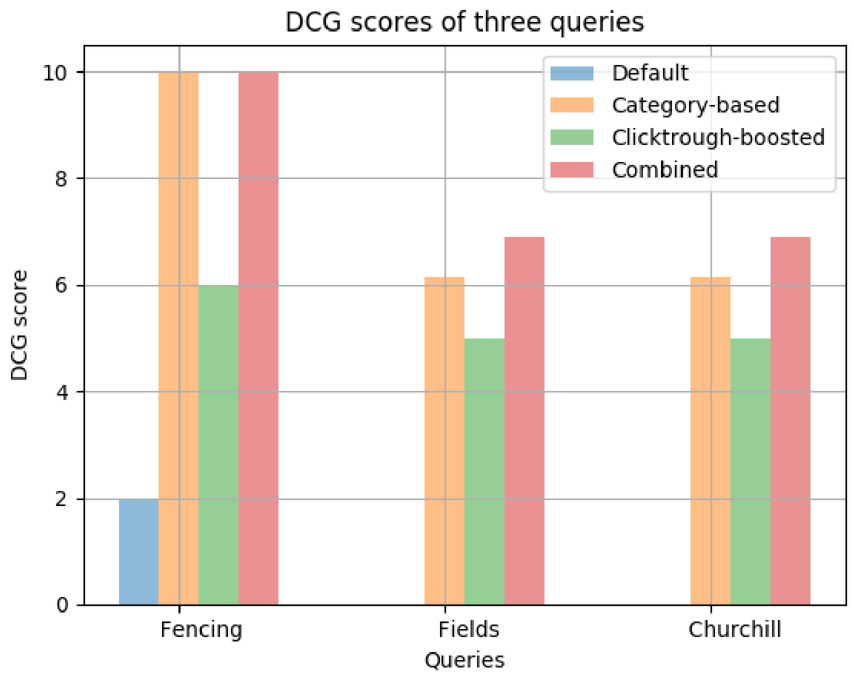

Figure 4.1: DCG scores

Conclusions

In this project we implemented two re-ranking algorithms as well as an algorithm using both. Our hope was that by tracking categories associated with page visits, we could get some idea of what the user is interested in and thus being able to boost articles that the user may find relevant in other queries. This approach seems to work, particularly when the query is ambiguous, although we find that often the categories are not descriptive enough and some articles that are related or relevant are not marked with the proper categories. This of course leads to the algorithm not boosting all articles that the user might feel relevant or interesting. Results when categories are consistently and descriptively applied are promising.

Users that are interested in some subject often visit articles around that topic, which hopefully builds up a user profile containing a spread of categories instead of a small number of precise categories. After a long period of tracking, this translates into better results for documents directly relevant to the query as well as non-relevant but related documents. This last effect improves search performance if we assume the user's desired information reflects a combination of both their own interests and information presented in a page. One of our tested queries 4.1 shows an inkling of this effect where the user has looked at articles with the related, "Fencing" and "Sword" categories and indeed they influence the results of the query positively.

More investigation is needed into methods of fine-tuning the parameters used in the model, as well as the calculation of category similarity scoring. In particular, the performance of a particular similarity metric is sensitive to the problem definition. If we for example wish to penalize topics containing more information than what the user asked for and is interested in, a cosine similarity measure will accomplish this. In the case of our encyclopedia database, however, we found that penalizing ”extra” information often leads to topics that are relevant in multiple fields being penalized for their versatility, as well as to penalties whenever a page has a rather exhaustive list of associated fringe categories.

Bibliography

[1] Aktolga, Elif, and James Allan. ”Reranking search results for sparse queries.” *Proceedings of the 20th ACM international conference on Information and knowledge management.*ACM, 2011.

[2] Speretta, Mirco, and Susan Gauch. ”Personalized search based on user search histories.” *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on.* IEEE, 2005.

[3] Zhuang, Ziming, and Silviu Cucerzan. ”Re-ranking search results using query logs.” *Proceedings of the 15th ACM international conference on Information and knowledge management.* ACM, 2006.

[4] Jrvelin, Kalervo, and Jaana Keklinen. ”IR evaluation methods for retrieving highly relevant documents.” *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 2000.

[5] Singhal, Ayush, and Sinha, Pradeep, and Pant, Rakesh. ”Use of Deep Learning in Modern Recommendation System: A Summary of Recent Works.” *International Journal of Computer Applications 180(7):17-22* 2017