Introducing the User-over-Ranking Hypothesis

Benno Stein and Matthias Hagen

Bauhaus-Universität Weimar
<first name>.<last name>@uni-weimar.de

Abstract The User-over-Ranking hypothesis states that rather the user herself than a web search engine's ranking algorithm can help to improve retrieval performance. The means are longer queries that provide additional keywords.
Readers who take this hypothesis for granted should recall the fact that virtually no user and none of the search index providers consider its implications. For readers who feel insecure about the claim, our paper gives empirical evidence.

1 Introduction

Web search is a standard information retrieval use case: information needs are satisfied with an indexed document collection by submitting keyword queries to a search engine and getting back ranked result lists. Typically, the user is neither given arbitrary access to the index nor has she knowledge about the engine's underlying retrieval model, its implementation details, and the like: the search engine appears as a black box.

For many easy queries like `google` or `ebay` a user does not have any effort besides typing the query—current web search engines work so well that the desired item will pop up in the top results. However, the scenario we are considering here is that of an experienced user who has a more intricate information need. We assume that the user knows a whole bunch of keywords that, in her opinion, all tell something about her information need. However, if this entire set is submitted as a single query, it is likely that the returned result list contains very few or even no elements. On the other hand, if single-word queries are submitted, the obtained result list lengths will be in index size order of magnitude—a fact termed as the "million or none problem" [11].

We argue that very promising queries are the ones that return just a "reasonable" number of results. This is inspired by the observation that the number of results a user will consider from the entire result list is constrained by her processing capacity k, determined by the user's reading time, her patience in browsing result lists, the available processing time, etc. *Underspecific* queries entail over-length result lists from which only a fraction—typically the top-ranked results—are processed at user site, whereas *overspecific* queries with only a handful of results usually do not help either. We argue that the probability to satisfy a user's information need by exploring the top-k results becomes maximum if the result list length is in the order of magnitude of the user's processing capacity. The user then is still able to check all the results and can avoid any search engine ranking issues that she cannot influence. We term this argument *the-user-knows-better hypothesis* or, more formally, *User-over-Ranking hypothesis*. See Figure 1 for an idealized illustration of the outlined connections. Empirical justification for both parts of Figure 1 is provided in Sections 2 and 3. Potential application areas for the User-over-Ranking hypothesis and related work is presented in Section 4.

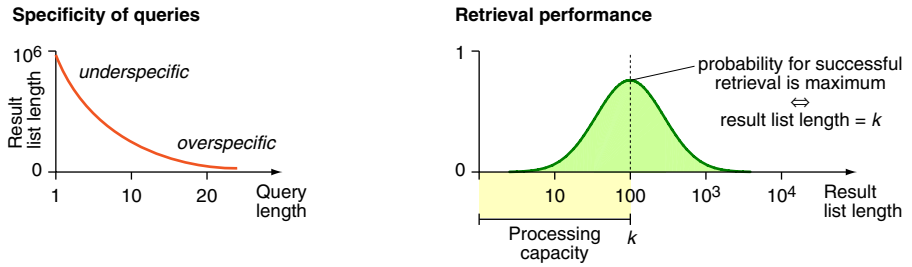

Figure 1. The User-over-Ranking hypothesis consists of two parts: specificity and retrieval performance. The specificity part (left) takes up the folklore assumption that short (underspecific) queries return exponentially more results than long (overspecific) queries. The retrieval performance part (right) assumes that a user's processing capacity k constraints the number of documents she will consider (we assume $k \leq 100$). For longer result lists, the user typically just scans the top-k documents. Shorter result lists often contain no relevant document at all. Under the User-over-Ranking hypothesis a result list length of k maximizes the probability to satisfy a given information need.

2 Specificity of Queries

In this section we empirically examine the specificity part of the User-over-Ranking hypothesis with real users' web queries from a large search engine query log.

Experimental Setup We use the AOL query log [9] that contains about 21 million web queries collected from about 650 000 AOL users over three months in 2006. A preprocessing removed the few users that could be considered as automatic bots (i.e., that submitted very many queries within very short time periods) as well as queries that just contain a URL. Such URL queries were probably submitted by users confusing the search box and the address bar of their browser. Finally, we eliminated query duplicates, and we restricted the query length to 22 keywords: there are too few queries with 23 or more keywords to draw reasonable conclusions. It remained 4 424 198 unique queries with an average length of 3.53 keywords. The distribution is as follows: about 300 000 single keyword queries, about 1 million each with 2-4 keywords, 0.5 million with 5 keywords, and then a steady decrease to about 100 queries with 22 keywords. All these queries were submitted to the Bing API during May 21-25, 2010, and Bing's reported estimations on the number of results were stored for each query.

Evaluation Note that the Bing API's estimations are not unbounded but the largest result list length is 2^{31} (the maximum `long`-value). As $10^9 < 2^{31} < 10^{10}$, we decided to use $\log_{10}$-discretization in order to have 10 bins for queries returning between 10^{n-1} and 10^n results for $n = 1, \ldots, 10$. In Figure 2 we show the resulting distribution of query length and result list length. For every query length q we depict the relative portion of queries with length q that fall in the same $\log_{10}$-scaled result list bin.

One can observe a "ridge" in Figure 2 whose characteristic implies that on average queries with more keywords have an exponential decrease in the estimated result count. We further analyze this decrease by computing, for every query length q, the median result list length of all queries of length q. The resulting medians clearly show an exponential decrease for larger q (see the AOL log plot in the left part of Figure 3 for a visualization). Using the Eureqa tool[1] and excluding the "outlier" for single keyword

[1] http://ccsl.mae.cornell.edu/eureqa

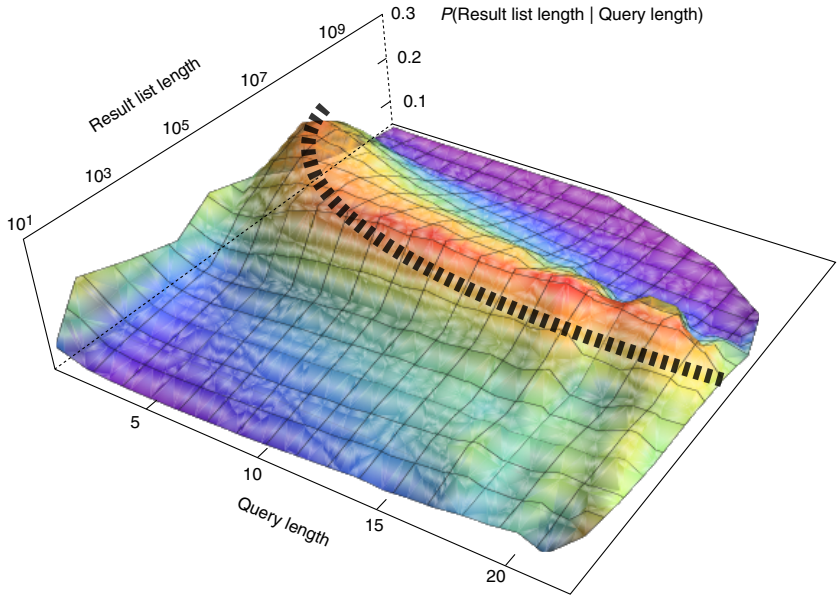

Figure 2. 3D distribution of query length (in keywords) over result list length ($\log_{10}$-scaled) in the AOL query log sample. For each query length $q = 1, \ldots, 22$ the plot shows the relative portion of queries having length q that return a specific result list length ($\log_{10}$-scaled). The dashed line further highlights the location of the plot's "ridge".

queries we obtain $f(q) = 8\,730\,000 \cdot q^{-2.29}$ as the corresponding decrease function. This gives experimental evidence for the specificity part of the User-over-ranking hypothesis and justifies the folklore assumption that short (underspecific) queries return exponentially longer result lists than long (overspecific) queries.

Finally, we briefly address the none-case of the million or none problem. For a closer consideration we treated all queries returning at most 10 results as queries representing the none-case. The reason is that some AOL log mirror pages exist on the web that just contain all the AOL log queries. Hence, hardly any query from the AOL log returns no results at all and result counts below 10 can be supposed to have had an empty result list during AOL log recording in 2006.

The portion of "no result"-queries we observe is a little surprising. Although the portion increases for longer queries, it stays below 10% up to 21-keywords queries; increasing to 20% for 22-keywords queries. This observation is also supported by the median result list length that stays above 2 000 for queries of at most 21 keywords and drops to about 600 for 22 keywords. One reason for the surprising behavior is that often the longer queries are verbose parts of song lyrics that still return a significant number of results.

3 Retrieval Performance

In this section we empirically examine how a query's result list length influences retrieval performance. For this purpose we conduct a TREC style experiment. Our results show that queries with short but not too short result lists have a better retrieval per-

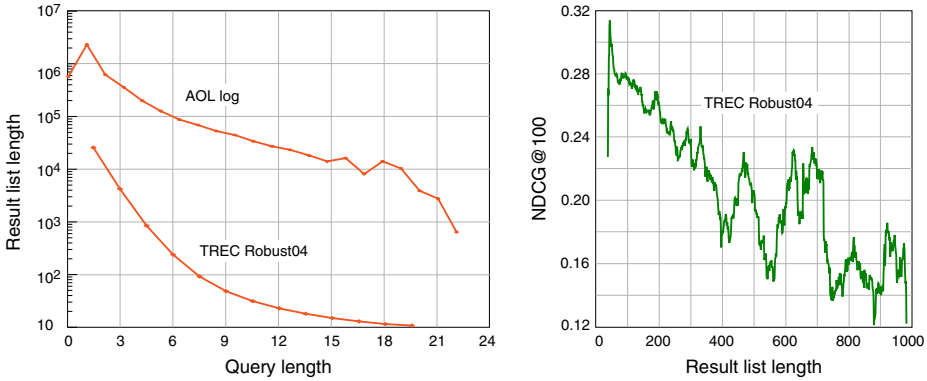

Figure 3. Left: The median result list length over query length (in keywords) in the AOL log (cf. Section 2) and the average result list length over query length (in key*phrases*) for the constructed phrase queries against TREC Robust04 (cf. Section 3). **Right:** Retrieval performance over result list length for the constructed phrase queries against TREC Robust04 (cf. Section 3). The NDCG@100-values are averaged over all queries that return a specific number of results. For smoothing purposes each data point is averaged with its 32 next neighbors to the left and to the right.

formance than queries with longer result lists. This gives empirical evidence for the retrieval performance part of the User-over-Ranking hypothesis: the idea that a user should strive for queries that return about as many results as the user's capacity is.

Experimental Setup For our experiments we choose the TREC Robust04 corpus as it is used in several studies [3, 7, 8] evaluating long query reduction—an interesting potential application area for the User-over-Ranking hypothesis (cf. Section 4). The Robust04 corpus contains 528 155 newswire documents from Federal Register, LA Times, Foreign Broadcast Information Service, and Financial Times. There are the TREC topics 301–450 and 601–700 associated with this corpus. On average these 250 topics contain 2.66 words in the title field, 14.5 words in the description field, and 38.3 words in the narrative field; and on average have 1 246 associated Qrels (documents from the corpus with relevance judgments).

For our experiments we use the BM25 retrieval model and querying should be possible with complete phrases and not just words. Therefore, we indexed the Robust04 corpus with Terrier 3.0 using block-indexing with a block size of 1. To produce queries for each topic we segmented the topic titles into phrases using the naïve query segmentation method from [5]. From the description and narrative fields we extracted keyphrases using an implementation of the head noun extractor described in [2]. Especially for the narrative part we performed some manual cleaning after the automatic keyphrase extraction in order to remove some non-relevant phrases like `"relevant document"` etc. For each topic we introduced an ordering of the phrases according to their first appearance in the original topic (i.e., the first phrase from the title, the second phrase from the title, ..., the first phrase from the description, ..., and finally the phrases from the narrative part). From the phrase ordering we only use the first 15 phrases per topic when available. Using the described technique we obtain on average 9.5 phrases per topic. For example we get `"whale watching california"` `"californian site"` `"whales"` `"habitat"` `"guide services"` for topic 660.

The queries are built as follows. Phrases are combined to all possible sequences with respect to the phrase ordering. Hence, there is no sequence where the i-th phrase comes before a phrase j for $j < i$, but phrases could be left out (e.g., a sequence with the first and third phrase but not the second is possible). Each sequence is a distinct query such that there are $2^{15} - 1$ possible queries for a topic with 15 phrases. All these queries were submitted to Terrier with highlighted phrases. For each query we stored the number of returned results and the first 1 000 documents when available. Queries returning at most 10 results were removed.

Evaluation Our first observation supports the findings of the AOL experiments (cf. Section 2), namely, that there is an exponential decrease in the result list length for longer queries. The TREC Robust04 plot in the left part of Figure 3 shows the average result list length for our constructed phrase queries at each query length level.

However, the main focus is on the retrieval performance. As stated before, our assumption is that a typical user has a processing capacity k that constraints the number of results she will consider from the whole result list. Typically, these considered results will be the top ranked documents. We fix $k = 100$, i.e., assuming that a user will check not more than the first 100 results. As for evaluating retrieval performance we use NDCG to account for the ranking of the relevant documents. Hence, we measure NDCG@100 for the obtained Terrier result lists with the notion that for lists longer than 100 results the user will not skim further. The right part of Figure 3 shows the averaged NDCG@100 for all queries having a specific result list length for all topics together. Note that the right part of Figure 3 does not completely reflect the idealized retrieval performance plot of Figure 1. Nevertheless, it clearly supports the retrieval performance part of the User-over-Ranking hypothesis: the best performing queries are the ones that return about as many results as the assumed user's capacity of checking at most 100 results.

4 Applicability and Related Work

One field where the User-over-Ranking hypothesis can be applied is that of long query reduction, i.e., handling verbose text queries, similar to the description parts of TREC topics or queries to medical search engines. Research on long queries is on the rise [1, 3, 6, 7, 8], as a typical web query nowadays is becoming longer and longer, or, "more verbose." Existing approaches reduce the long user query to a shorter keyword query by applying sophisticated strategies that try to only focus on promising candidates of reduced queries. The User-over-Ranking hypothesis opens an interesting perspective from the user's site: promising candidates are the ones that return about as many results as the user can consider.

Using the User-over-Ranking hypothesis to identify promising queries cannot only be useful in reducing long queries but also for helping users that enter short and underspecific queries. Stein and Hagen [10] describe an automatic approach for helping users during search sessions. Their system combines a user's keywords from the short (underspecific) queries of a search session to a longer and more specific query. The User-over-Ranking hypothesis explains why the approach can improve user experience.

Also fully automatic query formulation systems can benefit from the hypothesis' insights. For example, the plagiarism detection system of [4] constructs web queries

against commercial search engines from keywords extracted from a suspicious document. The aim is to retrieve web documents with similar content from which the suspicious document's author might have plagiarized. The User-over-Ranking hypothesis states that the constructed queries should be enlarged with additional keywords till the result list length is short enough for the detection system to handle it completely.

5 Conclusion and Outlook

In this paper we postulate the User-over-Ranking hypothesis and give empirical evidence for it. The hypothesis is split into two parts. The specificity part picks up the folklore assumption that longer (and thus more specific) queries return exponentially fewer results. However, the actual crux of the User-over-Ranking hypothesis is the second part on retrieval performance. It states that queries that return about as many results as the user can consider will maximize the probability of satisfying the user's information need. The straightforward conclusion from both parts then is that sufficiently long and specific queries are the best choice to query a search engine. Hence, the User-over-Ranking hypothesis provides an explanation for the plausible fact that users can help search engines by adding additional keywords to underspecific queries in order to obtain fewer and better results. Of course, the hypothesis is not meant to be applied for "easy" queries but rather as a model to explain the success of refined querying strategies in more elaborate information need scenarios.

A potential application area for the User-over-Ranking hypothesis is long query reduction. The conclusion from the User-over-Ranking hypothesis for a bad performing long verbose query is to reduce it to a keyword query that returns a reasonable number of hits. Developing a corresponding long query reduction approach is an interesting task for future work.

Bibliography

[1] N. Balasubramanian, G. Kumaran, and V. R. Carvalho. Exploring reductions for long web queries. In *Proc. of SIGIR 2010*, pages 571–578.
[2] K. Barker and N. Cornacchia. Using noun phrase heads to extract document keyphrases. In *Proc. of AI 2000*, pages 40–52.
[3] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *Proc. of SIGIR 2008*, pages 491–498.
[4] M. Hagen and B. Stein. Capacity-constrained query formulation. In *Proc. of ECDL 2010*, pages 384–388.
[5] M. Hagen, M. Potthast, B. Stein, and C. Bräutigam. The Power of naïve query segmentation. In *Proc. of SIGIR 2010*, pages 797–798.
[6] S. Huston and W. B. Croft. Evaluating verbose query processing techniques. In *Proc. of SIGIR 2010*, pages 291–298.
[7] G. Kumaran and J. Allan. Adapting information retrieval systems to user queries. *Information Processing and Management*, 44(6):1838–1862, 2008.
[8] M. Lease, J. Allan, and W. B. Croft. Regression rank: Learning to meet the opportunity of descriptive queries. In *Proc. of ECIR 2009*, pages 90–101.
[9] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proc. of Infoscale 2006*, article number: 1.
[10] B. Stein and M. Hagen. Making the most of a web search session. In *Proc. of WI-IAT 2010*, pages 90–97.
[11] D. Tunkelang. *Faceted Search*. Morgan & Claypool Publishers, 2009.