Query Suggestions as Summarization in Exploratory Search

Medlar, Alan

ACM
2021

Medlar , A , Li , J & Glowacka , D 2021 , Query Suggestions as Summarization in Exploratory Search . in Proceedings of the 2021 Conference on Human Information þÿ Interaction and Retrieval . ACM , pp. 119 128 , ACM SIGIR Conference Information Interaction and Retrieval , Canberra , Australia , 14/03/2021 . https://doi.org/10.1145/3406522.3446020

http://hdl.handle.net/10138/352580
https://doi.org/10.1145/3406522.3446020

unspecified
acceptedVersion

Query Suggestions as Summarization in Exploratory Search

Alan Medlar
University of Helsinki
alan.j.medlar@helsinki.fi

Jing Li
University of Helsinki
jing.li@helsinki.fi

Dorota Głowacka
University of Helsinki
dorota.glowacka@helsinki.fi

ABSTRACT

Query suggestions have been shown to benefit users performing information retrieval tasks. In exploratory search, however, users may lack the necessary domain knowledge to assess the relevance of query suggestions with respect to their information needs. In this article, we investigate the use of *alternative queries* in exploratory search. Alternative queries are queries that would retrieve similar search results to those currently visible on-screen. They are independent of the original search query and can, therefore, be updated dynamically as users scroll through search results. In addition to being follow-on queries, alternative queries serve as keyword summaries of the current search results page to help users assess whether results are inline with their search intents.

We investigated the use of alternative queries in scientific literature search and their impact on user behavior and perception. In a user study, participants inspected half as many documents per query when alternative queries were present, but were exposed to over 40% more search results overall. Despite using them extensively as follow-on queries, user feedback focused on the summarization properties offered by alternative queries; finding it reassuring that documents were relevant to their search goals.

CCS CONCEPTS

• **Information systems** → **Search interfaces**; **Query suggestion**; **Presentation of retrieval results**.

KEYWORDS

exploratory search; query suggestions; summarization; scientific literature search; alternative queries

ACM Reference Format:
Alan Medlar, Jing Li, and Dorota Głowacka. 2021. Query Suggestions as Summarization in Exploratory Search. In *Proceedings of the 2021 ACM SIGIR Conference on Human Information Interaction and Retrieval (CHIIR '21), March 14–19, 2021, Canberra, ACT, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3406522.3446020

1 INTRODUCTION

Search can be divided into two broad categories: lookup and exploratory search [32]. Lookup search is where users have precise search goals, such as question-answering and fact-finding. These search tasks are well-understood in terms of user behavior and information retrieval system design. Exploratory search, however, is characterised by user uncertainty with respect to search domain and information seeking goals. Exploratory search is, therefore, considered challenging [49] and requires search systems to provide additional support to help users satisfy their information needs. Query suggestions are one of many techniques that have been shown to benefit users performing exploratory search [49]. Query suggestions are usually generated from query logs, using historical data to identify common follow-on queries and popular destinations for users issuing similar search queries [48]. In the absence of query logs, query suggestions have been generated directly from the corpus [6], external data sources [40] or using pseudo-relevance feedback [24]. In the case of pseudo-relevance feedback, query suggestions are effectively extractive keyword summaries of search results; identifying phrases that are over-represented in search results compared to the corpus as a whole [5].

In this article, we focus on how to use query suggestions in scientific literature search, where users are searching for the purpose of knowledge acquisition. In previous research, query suggestions were investigated in the context of exploratory search tasks, such as purchasing a VoIP telephone [48] or identifying topically relevant newspaper articles [24]. In these kinds of task, users already have sufficient knowledge to evaluate whether suggested queries are better than their own search query. For example, when searching for information about VoIP telephones, users can leverage their existing knowledge of cellular and landline telephones to assess the quality of the provided suggestions. This is not necessarily the case for exploratory search of scientific literature, however, where novice users can be highly uncertain about document relevance [34] and scroll through significantly more search results compared to lookup search [2, 3]. This can lead to frustration, either because of time wasted inspecting lower ranked search results or because the accumulation of low quality feedback leads to query drift [38].

Our approach to query suggestions for exploratory search focuses on *summarization*, in addition to providing follow-on queries. We summarize search results by generating *alternative queries*, i.e. queries that would retrieve similar search results to those visible on-screen. We generate alternative queries using a nearest neighbor-based approach with a novel SERP embedding model based on a sequence-to-sequence autoencoder. Alternative queries provide succinct keyword-based summaries that, as they are independent of the original search query, can be updated dynamically as users scroll down the search results page. We take advantage of being able to dynamically update query suggestions by integrating them into an interface with infinite scroll; fetching additional search results on-demand as users scroll down the page. Infinite scroll serves two functions: first, removing pagination allows for fine-grained information to be displayed at all positions of the ranking and, second, provides an opportunity to animate the ordering of query suggestions, clearly highlighting their changing importance

CHIIR '21, March 14–19, 2021, Canberra, ACT, Australia

ACM ISBN 978-1-4503-8055-3/21/03…$15.00
https://doi.org/10.1145/3406522.3446020

to users as they scroll. This interface allows searchers to quickly assess whether the currently displayed documents are relevant to their search intent.

In this article, we make the following contributions: (i) a novel query suggestion method that uses a sequence-to-sequence autoencoder to identify alternative queries, (ii) a search interface based on infinite scroll that dynamically updates query suggestions as users scroll through results, (iii) an expert assessment of the summarization properties of query suggestions, contrasting our approach to methods based on pseudo-relevance feedback and (iv) a user study demonstrating that users appreciated the availability of those summaries during exploratory scientific literature search.

2 BACKGROUND

In this section, we review prior work on query suggestions in interactive information retrieval and exploratory search.

2.1 Query Suggestions

Query suggestions are queries displayed alongside search results that are generally intended to be used as follow-on queries or reformulations of the present search query. Methods for query suggestion tend to use information from query logs to infer relatedness, either on the basis of click-throughs [8, 48] or query co-occurrence, i.e. session-based methods [17]. Recent advances have improved the context-awareness of query suggestions within the same search session [47] and developed personalized query suggestions [46, 52]. In the absence of query logs, query-phrase correlations [6] and pseudo-relevance feedback [24] have been used to make query suggestions. While not a method for query suggestion, query substitution identifies typical alternatives to queries issued by web users to transparently improve recall in sponsored search [21]. Our approach similarly generates alternative queries, but they are presented to the user to summarize the current search results page.

Prior to the availability of query logs, query expansion methods suggested terms and phrases using, for example, relevance feedback [26], concept hierarchies [44] and over-representation statistics [9]. More recently, query expansion methods have used word embeddings [37] to suggest search terms that are semantically similar to the search query. For this purpose, embedding models have been trained on the entire corpus [27], search results [12] and on the basis of document relevance, rather than term proximity [50]. Our approach has much in common with these semantic methods, however, our model is based on document representations from search results and is independent of the original search query.

2.2 Query Suggestions in Exploratory Search

Query suggestions in exploratory search have mostly been investigated in the context of web search, product search and using TREC test collections. In early work on interactive information retrieval, for example, relevance feedback was used to suggest additional search terms for users to expand their search queries [26]. More recently, Kelly et al. used pseudo-relevance feedback and clustering to generate query suggestions based on the top-100 documents returned by a given query [24]. In both instances, users were tasked with finding topically relevant news articles from a TREC test collection.

In the context of web search, query logs or other external data sets are generally used to generate query suggestions. White et al. annotated web search results with popular destinations for users issuing similar search queries [48]. QAque used data from Yahoo! answers, a question answering community, for faceted query expansion [40]. Finally, knowledge graphs based on search logs have been shown to produce query suggestions that are beneficial to users engaged in exploratory search [29, 31].

There are also a few systems that focus on query suggestions in exploratory scientific literature search. PULP used a dynamic visualization of topic models built for each year of the ArXiv repository to assist users with query formulation [35]. Recently, faceted query suggestions derived from pre-defined article keywords from ACM Digital Library were shown to improve user performance, although the analysis was based solely on user behavior and task performance with no user satisfaction data (questionnaires or interviews) collected [43]. These systems were designed to assist with initial and follow-on queries, but not to help users understand their current search results.

2.3 Summarization in Exploratory Search

This article focuses on the search result summarization properties of query suggestions during scientific literature search. Summarization is a broad topic that can touch upon many different aspects of exploratory search system design. Summarization has been used, for example, in summary visualization to enable faster relevance feedback [22, 23, 33], interactive exploration of faceted data sets [1, 51], topic model-based visualizations of large corpora [16, 30], query formulation through global visualization of the search space [13, 15, 35] and interactive interfaces to help the user comprehend the information space [10, 39]. These interfaces can greatly aid users conducting exploratory search for knowledge acquisition by visualizing the global search space and its relationship to the user's search intent. Our approach is more local: we summarize the information that is currently on-screen in order to help searchers avoid wasting time on less relevant search results. Studies of exploratory search behavior have highlighted its dynamic nature with narrowing and broadening of search queries as well as changes in click behavior throughout a search session [4, 11, 36]. This suggests that exploratory search should be supported by more fine-grained and dynamic visualisation or summarizarion methods. Our approach has specifically this goal in mind: helping users to understand search results and decide when to terminate the current search and reformulate the search query.

2.4 Summary and Contributions

Although query suggestions are known to complement exploratory search, this has tended to be investigated with less cognitively demanding search tasks [24, 48]. In this paper, we distinguish our contribution on query suggestions in exploratory search in the following ways: (i) we are focused on scientific literature search, (ii) our goal is to provide summarization, as well as follow-on search queries, (iii) our interface dynamically updates suggestions as users scroll, providing fine-grained context intended to aid in knowledge acquisition, and (iv) our approach does not require query logs and can therefore be used in specialist retrieval systems.

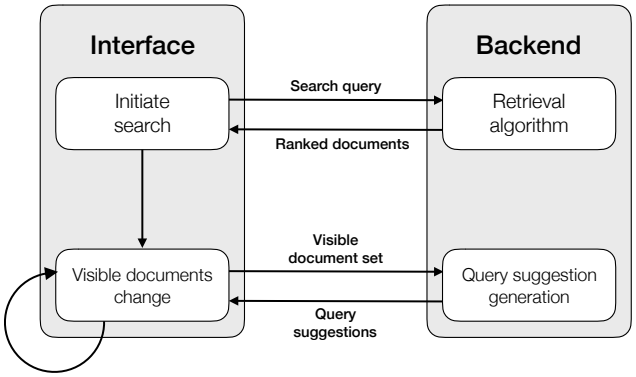

Figure 1: System overview: users initiate a search with a search query. A retrieval algorithm is used to rank search results. Whenever the set of visible documents changes (when a new search is initiated or due to scrolling), the document set is sent to the query suggestion module, which returns a list of suggestions. Whenever the user scrolls near the bottom of the SERP, the retrieval algorithm is called again, implementing infinite scroll.

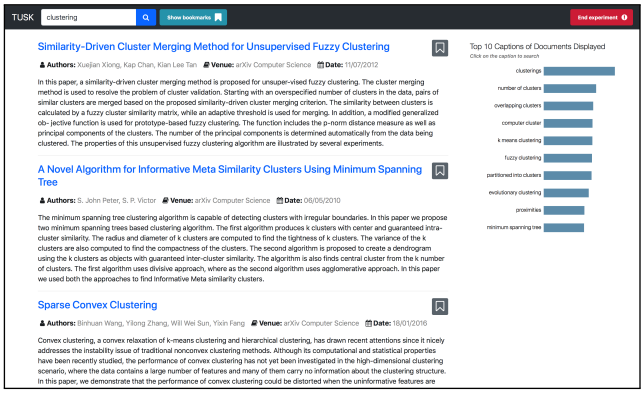

Figure 2: The search interface incorporating query suggestions. The baseline interface in the user study was identical with the exception that the query suggestion component (right-hand margin) was absent. Query suggestions are updated dynamically to reflect the current search results currently displayed on-screen.

3 APPROACH

Our system has three components: the user interface, the query suggestion method and the ranking algorithm. The current version of the system operates on ~170, 000 Computer Science articles from the arXiv repository (www.arxiv.org, downloaded November 2018). An overview of the system is presented in Figure 1.

3.1 Interface Design

Figure 2 shows the user interface of the system. A search is initiated by the user typing a search query into the search box at the top of the page. This results in a list of documents appearing on-screen, ranked by the Okapi BM25 algorithm [20]. The interface implements infinite scroll, fetching additional search results on-demand just prior to the user scrolling to the bottom of the page. A list of query suggestions generated from the set of visible documents is anchored to the right-hand margin. Query suggestions are ranked by their confidence level, which is also visualized as a horizontal bar graph (longer bars indicate greater confidence). As users scroll down the page, the query suggestions are updated to reflect the documents currently visible on-screen, i.e. the rank/bar length of the currently displayed queries change or are swapped out for new ones. These transitions are animated, giving the user a visual cue in their peripheral vision when there is a change.

3.2 Alternative Query Generation

Our approach to query suggestions uses alternative queries (queries that retrieve similar documents to those visible on-screen) to summarize the content of search results. To generate alternative queries we used an autoencoder, a type of neural network, to generate a learned representation for search engine results pages (SERPs). We refer to the autoencoder as the *SERP embedding model*. To train the model, we do not rely on query logs, but simulate queries and their associated search results directly from the document corpus. Once the model is trained, we generate SERP embeddings for each simulated query and store the results in a lookup table to allow us to identify which query is associated with each SERP embedding.

We generate query suggestions when users either issue a new search query or when the visible set of documents changes as a result of scrolling. The system uses the top-10 search results starting from the first visible document and gives them to the SERP embedding model to generate an new embedding, $\vec{v}$. We rank all SERP embeddings in the lookup table by their cosine similarity with $\vec{v}$ to identify its nearest neighbors. Finally, we return the simulated queries associated with the SERP embeddings at the top of the ranking as query suggestions.

In the following sections, we expand on these details, covering the architecture of the SERP embedding model, data generation and augmentation for model training, and query suggestion post-processing.

3.3 SERP Embedding Model

Figure 3 shows the SERP embedding model, a sequence-to-sequence autoencoder that learns an unsupervised representation of ranked search results. Autoencoders are composed of an encoder network (Figure 3, left) and a decoder network (Figure 3, right) [18]. Both encoder and decoder networks were Long Short-Term Memory (LSTM) networks [19]. Given a sequence of documents, $\mathbf{x} = (x_1, x_2, x_3, \ldots x_T)$, the encoder network reads each document representation, x_t, sequentially and updates the hidden state in the LSTM network, h_t. After the last document in the SERP, x_T, the hidden state, h_T, can be viewed as a learned representation of the SERP, that we refer to as a *SERP embedding*. The decoder network takes the SERP embedding, h_T, as input and attempts to regenerate the original input sequence as $\mathbf{x}'$. Both encoder and decoder networks are trained jointly by minimizing the reconstruction error: $\mathcal{L}(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||^2$. The goal is that the SERP embedding will be a meaningful representation of the SERP because the input sequence

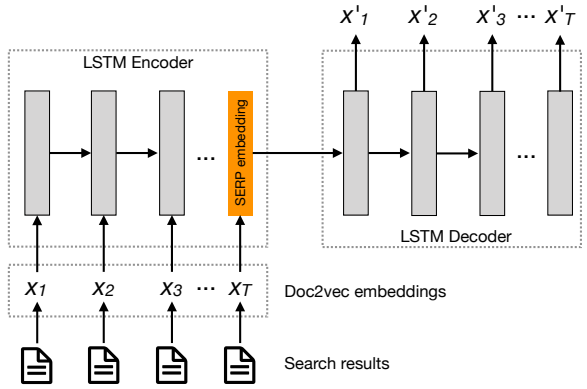

Figure 3: The SERP embedding model is a sequence-to-sequence autoencoder with two LSTM networks: a encoder network (left) and a decoder network (right). The encoder network accepts a sequence of Doc2vec embeddings and outputs a learned representation that we term the SERP embedding. The decoder is trained jointly with the encoder using reconstruction loss.

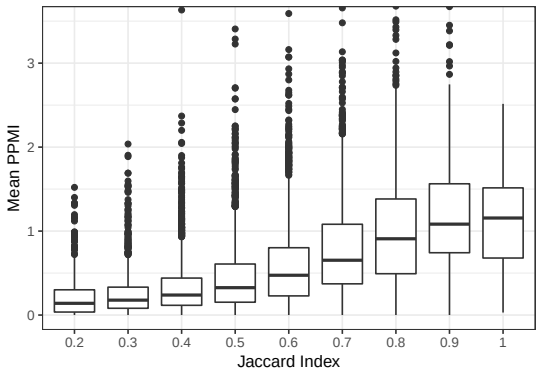

Figure 4: Average query suggestion PPMI correlates with Jaccard Index. The SERP embedding model's performance tends to be higher when query phrases have higher average PPMI, i.e. when search results have a more coherent theme.

can be reconstructed from the SERP embedding by the decoder network.

Our training data consisted of a collection of 299,451 unique search results pages associated with simulated search queries (data generation procedure described below). We limited the input sequence to 10 documents per SERP to simplify training. Each document was represented using Doc2vec [28], which was trained using the PV-DBOW (distributed bag-of-words version of paragraph vectors) method, ignoring terms with a frequency < 10 to infer document vectors of length 300. To train the sequence-to-sequence autoencoder, we used 95% of the data for training and the remaining 5% for validation to avoid overfitting. We used Adam [25] with a learning rate of 0.001 and trained for 10 epochs, after which the validation loss ceased to improve. We performed very little hyperparameter optimization because we had no objective quality metric to judge the summarization properties of our queries. Instead, we trained several models with different embedding lengths from which we selected the model with length 300 by manual inspection of the predicted queries. As such, our results should be considered a lower bound for performance using this approach.

3.4 Data Generation and Augmentation

As we do not have query logs, in order to gather the data necessary to train the SERP embedding model, we generated search queries from the corpus of documents. We used these queries to search the same corpus to simulate SERPs. For the corpus, we used all ~170, 000 Computer Science articles from the arXiv repository. Each document was represented by concatenating the article title and abstract text. To generate search queries, we preprocessed each document (removed punctuation, lowercased and filtered out tokens composed of numbers) and extracted bigrams, which were allowed to skip over stop words. We performed bigram detection twice, extracting up to 4-grams. From this collection of automatically-derived queries, we filtered out the most common (appearing $>$ 5, 000 times), rare ($<$ 10 times) and short ($<$ 3 characters) search queries. This procedure extracted 70,551 queries, including very specific (e.g. "deep recurrent neural networks") and very broad queries (e.g. "machine learning").

As the number of query phrases was relatively limited, we performed data augmentation to increase the diversity of the training set. Using our collection of queries, we retrieved up to the top-50 search results from the corpus using Okapi BM25 [20]. Next, we used different ranges and stride lengths to extract multiple "top-10" rankings per search query. For example, if there were 30 search results, then we used ranges 1–10, 11–20 and 21–30, as well as every third document with offsets 0, 1 and 2. This procedure yielded 299,451 unique document rankings associated with search queries that were used to train the SERP embedding model.

3.5 Query Suggestion Post-processing

The SERP embedding model is trained using search results pages generated with individual queries, however, it could also be presented with documents from a mixture of topics. Ideally, when the SERP embedding model is presented with documents from two separate search queries, the resulting SERP embedding should approximate the mid-point between the SERP embeddings for those two queries. To test whether this was the case we randomly selected two search queries, q_a and q_b, without replacement, from the training set and interpolated the mid-point between them by adding their respective embeddings: $\vec{v}_a + \vec{v}_b = \vec{v}_{ab}$. We then generated a second vector with our SERP embedding model using half of the search results associated with q_a and half from q_b: $encoder(d_a[1..5] + d_b[1..5]) = \vec{v}'_{ab}$. The output of the SERP embedding model, $\vec{v}'_{ab}$, is an approximation of $\vec{v}_{ab}$ because we only used half of the documents from q_a and q_b. Were $\vec{v}_{ab} \equiv \vec{v}'_{ab}$, then this would imply there is no information in the unused documents associated with q_a and q_b as they would have contributed nothing to the result. Finally, we calculated the Jaccard index, which is the proportion of identical queries in the 10 nearest neighbors between $\vec{v}_{ab}$ and $\vec{v}'_{ab}$ after filtering out duplicates. We repeated this procedure 10,000 times. Figure 4 shows how the Jaccard index varies with

the average positive pointwise mutual information (PPMI) between all pairs of query suggestions. Performance is worse when average PPMI is lower, making PPMI a predictor of embedding quality.

Taking these findings into consideration, we created a logistic regression model based on the mean PPMI between all pairs of query suggestions. We defined the response variable as whether the Jaccard index between interpolated and SERP embedding model-derived queries passed a threshold of 0.6 (threshold chosen empirically, data not shown). We use the probability output by the logistic regression model to determine the proportion of query suggestions to use from the SERP embedding model, with the remainder made up of phrases with the highest Okapi BM25 term weights (see Section 4.1 for details). This takes advantage of the SERP embedding model in scenarios where it performs well, but falls back to a simpler pseudo-relevance feedback method when we detect no coherent theme to the search results on the basis of PPMI.

4 EXPERT ASSESSMENT

We conducted an expert assessment of the summarization properties of alternative queries, focusing on relevance and specificity. We focused on search results covering multiple broadly-defined topics (e.g. "computer vision" and "autonomous driving") because the SERP embedding model was trained using search results from single topics and we wanted to understand how well different approaches generalized to more complex searches.

4.1 Baseline Methods

We compared our approach to several pseudo-relevance feedback-based methods for generating query suggestions. In this context, query suggestions are equivalent to extractive keyword summaries over search results [5]. The baselines were Rocchio's weights [42], using TF-IDF and Okapi BM25 [20] as weighting functions, χ^2 [14] and KL divergence [9]. Following [45], we set $k_1 = 1.2$ and $b = 0.75$ for Okapi BM25. All methods had access to the same document representation as the SERP embedding model (i.e. n-grams containing 1–4 terms) and were used to output the top-10 scoring phrases as query suggestions.

We tested four variants of our approach: (i) our approach, utilising both SERP embedding model and the post-processing procedure, (ii) our approach, but constrained to only output phrases present on the SERP, (iii) the SERP embedding model alone and (iv) the SERP embedding model, but constrained to only output phrases present on the SERP. Variants (ii) and (iv) will help us to understand the impact of using the entire vocabulary, whereas (iii) and (iv) will help us understand the effect of our post-processing procedure.

4.2 Study Design

We designed an experiment around scientific literature search. We assumed users were searching for literature related to two broadly defined topics, e.g. "autonomous driving" and "computer vision", and considered two different scenarios: (i) all search results contain both topics, that we refer to as *conjoint results* and (ii) the topics are mutually exclusive, with each document containing only one of the two topics, which we refer to as *disjoint results*. In terms of set operations, conjoint results are found at the intersection of two topics and disjoint results in the symmetric difference.

SERP Simulation. We identified pairs of topics related to machine learning by manually enumerating phrases from the Scikit-learn user guide section headings (https://scikit-learn.org/stable/user_guide.html) and identified phrases that co-occurred in our corpus. We manually inspected this list of topic pairs and removed subjectively broad (e.g. "machine learning") and narrow topics (as these would be lookup searches). We generated conjoint results by identifying all documents that contained both topics and randomly sampled 10 documents without replacement. Similarly, we generated disjoint results by identifying documents that contained one of the two topics at least 3 times, but not the other. We required topics to be present multiple times as documents where they appeared only once or twice tended to not be specifically related to that topic. Finally, we randomly sampled 5 documents without replacement for each topic and merged them together.

Expert Assessment. We randomly sampled 125 SERPs for conjoint and disjoint search results and for each SERP generated 10 query suggestions using all eight methods. The expert, a professor in machine learning, classified queries on the basis of whether they were relevant to a majority of documents in the SERP (i.e. summarized the document content appropriately). Assessment was carried out double-blind: SERPs were selected randomly, the queries generated by all methods were pooled, duplicates removed and the presentation order randomized. Relevant queries were further subdivided by their specificity into: *a*) narrow, *b*) in-between and *c*) broad. Similarly, not relevant query suggestions were further subdivided into: *a*) too generic and *b*) unrelated. Queries classified as "relevant/narrow" were more specialized terms, e.g. names of specific methods. Queries classified as "relevant/broad" were the opposite, focusing on broader topics or classes of method, e.g. "clustering". "Relevant/in-between" were relevant queries that fit neither narrow nor broad categories. For not relevant queries, "too generic" occur frequently in Computer Science, such as "model" and "algorithm". "Unrelated" included terms such as "state of the art" that are uninformative. In Section 4.4, we review each assessment category in terms of IDF to show that these assessments were highly consistent.

4.3 Results

We present the results from the expert assessment of conjoint and disjoint SERPs in Tables 1 and 2, respectively. In addition to counts for each assessment category, we include precision@10 (average proportion of relevant queries) and precision@5. We also calculated the number of excess queries as the mean absolute difference from equality, i.e. $|5 - n|$, where n is the number of queries more strongly associated with one of the two topics by PPMI in the document corpus. Excess queries can range from 0 (5 queries per topic) to 5 (10 queries for one topic and 0 for the other).

Conjoint Results. Our approach had 5% higher precision@10 than the best performing SERP embedding model and 26% more than Okapi BM25, the best performing pseudo-relevance feedback method. This lead dropped to 3% and 11% for precision@5 compared to the SERP embedding model and Okapi BM25, respectively. Our approach had the lowest number of excess queries, treating both topics more fairly than all other methods. Post-processing was essential to the success of our approach: 74% of query suggestions

	Excess queries	Relevant/narrow	Relevant/in-between	Relevant/broad	Not relevant/too generic	Not relevant/unrelated	P@10	P@5
TF-IDF	2.4	308	88	375	301	178	0.62	0.77
χ^2	3.9	412	114	254	176	294	0.62	0.74
KL Divergence	2.4	301	97	403	311	138	0.64	0.76
Okapi BM25	2.5	301	86	441	284	138	0.66	0.81
SERP emb. (const.)	2.4	476	25	457	159	132	0.77	0.87
SERP emb.	2.0	553	45	391	118	143	0.79	0.85
Our method (const.)	2.4	468	28	488	144	121	0.79	0.89
Our method	**1.8**	505	55	478	117	95	**0.83**	**0.90**

Table 1: Expert assessment of query suggestions for conjoint results. Our method performs best in precision@10, precision@5 and divides queries most evenly between topics. "Const." refers to queries being constrained to terms appearing in search results as in pseudo-relevance feedback.

	Excess queries	Relevant/narrow	Relevant/in-between	Relevant/broad	Not relevant/too generic	Not relevant/unrelated	P@10	P@5
TF-IDF	**0.9**	299	100	325	240	286	0.58	0.71
χ^2	1.4	344	133	269	120	305	0.60	0.73
KL Divergence	1.2	254	101	340	255	300	0.56	0.71
Okapi BM25	1.5	226	118	379	238	289	0.58	**0.74**
SERP emb. (const.)	1.7	223	37	418	190	382	0.54	0.63
SERP emb.	1.6	231	21	401	145	452	0.52	0.59
Our method (const.)	1.5	233	47	461	194	315	0.59	0.73
Our method	1.5	245	60	504	156	285	**0.65**	0.72

Table 2: Expert assessment of query suggestions for disjoint results. Our method has the highest precision@10, but lags behind Okapi BM25 for precision@5. "Const." refers to queries being constrained to terms appearing in search results, as in pseudo-relevance feedback.

came from the SERP embedding model and the remainder came from Okapi BM25 weighted phrases.

The pseudo-relevance feedback methods had similar overall performance to one another in terms of precision, but skewed to either more narrow (χ^2) or broad (TF-IDF, Okapi BM25 and KL divergence) queries. All pseudo-relevance feedback methods were outperformed by both SERP embedding models in terms of precision@10 and precision@5. The embedding approaches excel at finding narrowly relevant queries. Constraining methods to only predict queries present in documents on the SERP had a minor effect on performance, suggesting that there is little, if any, benefit from drawing queries from the entire vocabulary. Removing this constraint, however, improved fairness (i.e. excess queries was lower).

Disjoint Results. Our approach had 20% higher precision@10 than the best performing SERP embedding model and 8% more relevant queries than χ^2, the best performing pseudo-relevance feedback method. For precision@5, our approach was consistently better than the embedding models by a margin of at least 22%, but 1-2% worse than Okapi BM25 and χ^2. All methods performed worse in terms of precision compared to the conjoint experiment, but better in terms of excess queries. While the pseudo-relevance feedback methods had a moderate drop in performance, the semantic methods' performance dropped dramatically. In this scenario, our approach automatically compensated by using the SERP embedding model to only predict 49% of query suggestions. In this experiment, constraining the SERP embedding model to queries on the SERP actually improved performance, preventing many irrelevant queries from being output.

4.4 IDF Analysis

We used inverse document frequency (IDF) to measure the predicted queries' specificity, with higher IDF indicating their presence in fewer documents in the corpus. Figure 5 shows that all methods predict queries with a wide range of specificities. Okapi BM25, KL divergence and TF-IDF tended to predict more common queries than other methods. TF-IDF, despite clear similarities with Okapi BM25, had a much wider spread of IDF values, which was reflected in terms of performance. χ^2 appears to focus on rarer terms, which explains why it predicted more narrowly relevant queries (see Table 1). Unfortunately, this can lead to error, χ^2 suffered from many irrelevant queries, which is concordant with its IDF distribution being most similar to random (the term frequency distribution has a long tail, so random sampling will include mostly rare terms). In terms of median IDF, the methods using the SERP embedding model sat between χ^2 and the other pseudo-relevance feedback methods.

We additionally used IDF to investigate the properties of the blinded expert assessments. In general, the expert assessments were concordant with our expectations. The median IDF for relevant/narrow was higher than relevant/broad, which was higher than not relevant/too generic. The median IDF of relevant/in-between was the same as relevant/narrow, but it's third quartile was lower. The IDF distribution for not relevant/unrelated queries was higher than the relevant/broad and not relevant/too generic categories because it contains rarer, noisy terms.

5 USER STUDY

From the expert assessment we know that alternative queries provide better summaries compared to pseudo-relevance feedback methods, but we also want to know whether they help users performing exploratory scientific literature search. We conducted a within-subject user study. We were particularly interested in users' perceptions of query suggestions; whether they were perceived as appropriate follow-on queries and whether they were actually used for summarization purposes. We compared two systems: one incorporating query suggestions and a baseline that did not. Thus,

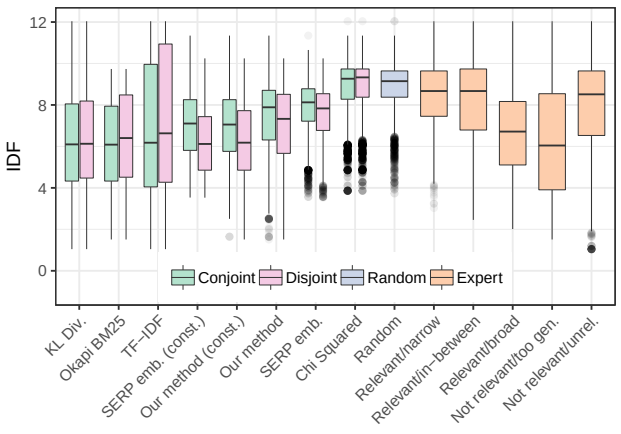

Figure 5: Boxplots of inverse document frequencies (IDF) of suggested queries. Higher IDF indicates greater specificity. We show IDF for each method, for both conjoint and disjoint experiments, and for each expert assessment category.

any differences in the observed user behavior can be attributed to the presence of query suggestions.

5.1 Participants

Prior to the experiment, potential participants were pre-screened in the following manner. We asked users to rate their familiarity on a 4-point scale (where 1 = "no knowledge" and 4 = "highly familiar (I worked on project/thesis on this topic)") with the following topics: robotic surgery, political bias online, sports analytics, gender recognition, cancer diagnosis, autonomous driving evaluation, gender bias in natural language processing and music recommendation. The topics were selected beforehand to ensure appropriate coverage in the corpus. To ensure that users would engage in exploratory search, only topics with low levels of familiarity for a given user were considered, i.e. topics marked as 1 or 2. If a user indicated only two topics they are unfamiliar with, then those two topics were randomly assigned to the two systems. If users indicated low familiarity with more than two topics, then they were asked to select their two preferred topics, which were then randomly assigned to the two systems. Users who did not indicate low familiarity with at least two topics were excluded from participation in the study.

Nineteen participants (eight female), aged 23–48 (median age 30), passed the pre-screening and took part in the study. All participants were Computer Science students: 8 MSc and 11 PhD students in the early stages of their doctoral degree. Each participant was compensated with a book voucher worth 15 EUR for taking part in the study. Each participant performed two tasks – one with each interface. The order of the interfaces was balanced. Before performing the two tasks, participants were shown a video tutorial on how to use the two interfaces. This was followed by a practice session to allow the users to familiarize themselves with the systems at their own pace. In the practice session, the participants were instructed to perform a free search related to their own study interests. The interface with query suggestions was used in the practice session as it covers all the features present in both interfaces.

QS	B	p	Question
3.4	2.7	0.0049	1. I think that I would like to use this system frequently
1.7	1.9	0.107	2. I found the system unnecessarily complex
4.3	3.8	0.107	3. I thought the system was easy to use
1.6	**1.5**	0.726	4. I think that I would need the support of a technical person to be able to use this system
3.8	3.4	0.159	5. I found the various functions in this system were well integrated
1.9	1.9	0.705	6. I thought there was too much inconsistency in this system
4.3	4.2	1.0	7. I would imagine that most people would learn to use this system very quickly
1.9	2.1	0.129	8. I found the system very cumbersome to use
3.6	3.3	0.144	9. I felt very confident using the system
1.6	**1.5**	0.705	10. I needed to learn a lot of things before I could get going with this system

Table 3: SUS score averages for query suggestion interface (QS) and baseline (B) systems with p-values from Wilcoxon signed-rank test. Questions used a 5-point scale from 1 (strongly disagree) to 5 (strongly agree). Best score in each row is bold; higher is better for odd numbered questions and lower is better for even numbered questions.

5.2 Tasks and Procedure

In the search tasks, users were instructed to write a short essay draft on a given topic. The task descriptions followed the template:

> *"You are going to start a new research project on the following topic: X. You would like to learn as much information as possible about this topic, e.g. applications, problems, specific algorithms. Write your answer as an essay draft or in bullet points, and bookmark at least 10 relevant articles that you could use as a reference in writing the article."*

The search session with each system was limited to 30 minutes to ensure that the results obtained from both systems and all the users were not skewed by overly long search sessions. Participants were allowed to take a short break in between the two tasks. The average search session duration was 29.5 and 28.4 minutes for our system and the baseline, respectively. Participants were provided with pen and paper as well as a text editor for note taking and could select the method they felt most comfortable with. All participants opted for the text editor. After the search session was completed, participants could take additional time to finalize their draft essay.

After each task, the participants completed the SUS questionnaire [7] with 10 questions (Table 3) and a modified version of the ResQue questionnaire [41] with 12 questions (Table 4). During the experiments, we logged the queries issued by users, the query suggestions displayed, the documents presented and whether any of those documents were bookmarked. After both tasks were completed, the users completed a post-experiment questionnaire (Table 5) and we conducted a semi-structured interview. While the aim of the SUS and the ResQue questionnaires was to understand how our approach compares to the baseline, the focus of the post-experiment questionnaire and the interview was to better understand the utility of summarization in exploratory scientific literature search.

5.3 Results

We looked at the usability, task performance, user behavior and perception of query suggestions in scientific literature search.

5.3.1 Usability: Our system obtained higher scores in both SUS and ResQue questionnaires than the baseline. In SUS, the overall score was 76.8 for our system and 71.2 for the baseline. While the difference was not significant ($p = 0.136$, Wilcoxon signed-rank test), it shows that the usability of the system did not suffer from the added functionality provided by query suggestions. In ResQue, our system significantly outperformed the baseline averaging 83.2 versus 67.8 ($p = 0.001$, Wilcoxon signed-rank test), which shows that participants found the query suggestions useful.

5.3.2 Task Performance and User Behavior: Task performance was assessed by two experts based on participants' short essay drafts. The assessors were senior researchers with a background in machine learning and artificial intelligence with more than 10 years of post-PhD research experience who were not otherwise involved in the user study. Assessments were blinded. The ratings were done on 5-point scale from 1 (bad) to 5 (good). The inter-rater agreement between the two assessors was $\kappa = 0.82$ (weighted Cohen's Kappa test with linear weights). The average task performance was 2.95 with the baseline and 3.37 with our system ($p = 0.035$, Wilcoxon signed-rank test). To understand the difference in essay scores, we investigated how the presence of query suggestions affected how users interacted with the system. When using our system, users issued more queries per session (8.2 queries per session compared to 3.7 with the baseline ($p = 0.0006$, Wilcoxon signed-rank test)), they inspected fewer documents per query (7.8 documents per query compared to 18.6 with the baseline ($p = 0.004$, Wilcoxon signed-rank test)), but were exposed to more documents overall (55.3 documents compared to 38.7 with the baseline ($p = 0.02$, Wilcoxon signed-rank test)). There was no significant difference in the number of bookmarked documents, with 9.4 and 9.2 bookmarks for our system and the baseline, respectively. This is almost certainly due to the fact that users were instructed to bookmark at least 10 documents as part of the essay writing task. Finally, in the system with query suggestions, on average 49.3% of queries issued by the user were from query suggestions.

5.3.3 User Perception: After completing both search tasks, users completed a post-experiment questionnaire (Table 5) and a semi-structured interview. During the post-experiment questionnaire, users stated that they preferred our system to the baseline almost unanimously (18/19, $p = 7.6 \times 10^{-5}$, binomial test). A majority of users felt that the presence of query suggestions reassured them that search results were relevant to their search goals (16/19, $p = 0.004$). While we were concerned that the dynamic nature of the query suggestions would annoy users engaged in exploratory search, none of the participants thought they were distracting (0/19, $p = 3.8 \times 10^{-6}$), although a minority found the animation distracting (3/19, $p = 0.004$).

Similarly, during the semi-structured interview, a majority of users (16/19) reported that they preferred our system to the baseline, while two users stated that, although they liked the augmented interface, they thought it would only be useful for very specific applications. The most often mentioned benefits were: a concise summary of the search results (8 users), help with search context (2) and help to find new and interesting documents faster (3). Individual comments made about our system included: "...provide an instant analysis of the documents" [p4], "gives an idea of what the content is like in general" [p5], "makes it faster to see the information as a whole" [p9], and "giving you a very quick summary" [p16]. Users also provided some suggestions for future improvements, e.g. some of the suggestions provided by the system were too generic to form a basis of a good follow-up query and that we should indicate which papers are most correlated with which queries.

QS	B	p	Question
4.0	3.5	0.01	1. The documents recommended to me matched what I was searching for
3.7	3.1	0.0139	2. The system helped me discover new documents
3.6	2.8	0.1305	3. The documents recommended to me are diverse
3.6	2.8	0.0164	4. The system helped me find the ideal documents
4.2	4.1	0.7054	5. I became familiar with the system very quickly
3.8	2.8	0.0189	6. I found it easy to notice if the search results were not correct any more
3.9	3.2	0.011	7. I felt confident to modify my query
3.6	3.2	0.0522	8. Using the system to find what I like is easy
3.7	3.7	0.7192	9. I found it easy to re-find documents I had been recommended before
3.7	3.1	0.0079	10. The system gave me good suggestions
3.8	3.5	0.07	11. The system made me confident about the documents I bookmarked
3.6	3.2	0.0374	12. Overall, I am satisfied with the system

Table 4: ResQue score averages for query suggestions (QS) and baseline (B) systems with p-values from Wilcoxon signed-rank test. Questions used a 5-point scale from 1 (disagree) to 5 (agree). Best score in each row is boldface; higher is better.

6 DISCUSSION

Query suggestions have been shown to benefit users performing specific exploratory search tasks [24, 48], however, these tasks place very different cognitive demands on the user compared to scientific literature search. In this article, we were inspired by observations that, during exploratory scientific literature search, users scroll through more documents [2] and are more uncertain about document relevance [34] than in lookup search. This led us to consider the role that query suggestions could have in summarizing whether search results are inline with users' search intents.

In our expert assessment, we demonstrated that when search results contain multiple search topics (conjoint results), the semantic information provided by the SERP embedding model was sufficient to outperform pseudo-relevance feedback methods. Pseudo-relevance feedback produces noisier results (higher not relevant/too generic counts) because they were easily misled by terms that are common in a given topic (e.g. "model" in machine learning), but

Prop. agree	p-value	Question
0.947	7.6e-05	1. Which system did you prefer to use?
0.737	0.063	2. I found it easier to perform the search with query suggestions
0.737	0.063	3. I found it easier to write the essay draft with query suggestions
0.895	0.0007	4. The labels of the query suggestion interface are clear
0.632	0.359	5. The bars of the query suggestion interface are clear
0.474	1.0	6. The query suggestions should be an optional function
0.895	0.0007	7. The query suggestions enhanced my search session
0.895	0.0007	8. The query suggestions were related to my search results
0.842	0.004	9. The query suggestions reassured me that my search results were relevant to my search goals
0.737	0.063	10. The query suggestions provided a good summary of my search results
0.526	1.0	11. The query suggestions provided good followup queries
0.0	3.8e-06	12. The query suggestions were distracting
0.158	0.004	13. The query suggestion animations were distracting
0.579	0.647	14. The system's confidence in each query suggestion was clearly indicated
0.895	0.0007	15. The system was better with the query suggestions than without
0.0	3.8e-06	16. There were too many query suggestions

Table 5: Post-experiment questions with p-values. Question number 1 was binary. Questions 2–16 were scored on a 5-point scale from 1 (strongly disagree) to 5 (strongly agree). Prop. agree is the proportion of participants who preferred our system in Question 1 or who answered 4 (agree) or 5 (strongly agree) in Questions 2–16. P-values are from a one sample binomial test where the null hypothesis is a proportion of 0.5.

rare in the corpus as a whole. Next, we showed that when multiple search topics were mutually exclusive (disjoint results), our approach still had the best performance, though the performance of all methods were significantly degraded. We note that even when simulated topics were related to one another, there were always query suggestions that were either too generic or unrelated to search results. Indeed, even one of the study participants observed that some queries were too generic to be useful. This could have been due to us always displaying 10 query suggestions when there might not have been that many relevant queries for a given set of documents. Unfortunately, filtering out low quality queries is difficult, as even variables that correlate with relevance, such as IDF (Figure 5), only correlate weakly. To improve the quality of queries further, we would need to understand how users conceptualize their relevance in order to better calibrate the specificity of query suggestions to suit individual knowledge levels.

In the user study, we saw that the presence of query suggestions dramatically impacts user behavior: users issue more queries, investigate fewer documents per query, but are exposed to more documents overall. From a user behavior perspective, it appears that query suggestions aid users in the decision-making process of whether or not to terminate the current phase of their information seeking, i.e. whether to reformulate the search query or continue scrolling through the current search results. Athukorala et al. [2] showed that behavioral variables, such as the number of documents investigated, are key discriminative indicators of whether users' are performing exploratory or lookup search. Our results, however, suggest that such behaviors are a consequence of search interfaces not supporting the kinds of decisions needed to navigate unfamiliar topics.

Finally, users stated almost unanimously that they preferred the system with query suggestions over the baseline (Table 5, Q1). In line with our expectations, users were reassured by the queries that search results matched their search intent (Table 5, Q9 and Table 4, Q6). These findings are suggestive that query suggestions help to bridge the knowledge gap between users who better understand their search domain and those that do not. What is not clear, however, is whether the system would be as positively received by users performing lookup search; the queries could be more distracting and the errors (non-relevant queries) more obvious.

In future work, we want to expand on several ideas touched upon by this work. The SERP embedding model introduced in this article provides a general method to condense information about sequences of documents to points in latent space. We are interested in whether this technique can be used to characterise whole search sessions, with the trajectory being used to distinguish between different search strategies or even to identify struggling behavior. Second, our findings suggest that augmented interfaces can make exploratory search behavior appear more lookup-like (fewer queries, fewer documents inspected per query, etc.). We want to investigate whether reducing the divergence between exploratory and lookup search behavior could be used as a general evaluation metric for exploratory search systems.

REFERENCES

[1] Jae-wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, and Radu Florian. 2010. Semantic annotation based exploratory search for information analysts. *Information processing & management* 46, 4 (2010), 383–402.

[2] Kumaripaba Athukorala, Dorota Głowacka, Giulio Jacucci, Antti Oulasvirta, and Jilles Vreeken. 2016. Is exploratory search different? A comparison of information search behavior for exploratory and lookup tasks. *Journal of the Association for Information Science and Technology* 67, 11 (2016), 2635–2651.

[3] Kumaripaba Athukorala, Alan Medlar, Antti Oulasvirta, Giulio Jacucci, and Dorota Glowacka. 2016. Beyond relevance: adapting exploration/exploitation in information retrieval. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*. ACM, 359–369.

[4] Kumaripaba Athukorala, Antti Oulasvirta, Dorota Głowacka, Jilles Vreeken, and Giulio Jacucci. 2014. Narrow or broad?: Estimating subjective specificity in exploratory search. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 819–828.

[5] Santosh Kumar Bharti and Korra Sathya Babu. 2017. Automatic keyword extraction for text summarization: A survey. *arXiv preprint arXiv:1704.03242* (2017).

[6] Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. 2011. Query suggestions in the absence of query logs. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 795–804.

[7] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[8] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 875–883.

[9] Claudio Carpineto, Renato De Mori, Giovanni Romano, and Brigitte Bigi. 2001. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems (TOIS)* 19, 1 (2001), 1–27.

[10] Duen Horng Chau, Aniket Kittur, Jason I Hong, and Christos Faloutsos. 2011. Apolo: making sense of large network data by combining rich user interaction and machine learning. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 167–176.

[11] Dongho Choi. 2017. *A Study of Information Seeking Behavior: Investigating Exploratory Behavior in Physical & Online Spaces*. Rutgers The State University of New Jersey-New Brunswick.

[12] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query Expansion with Locally-Trained Word Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 367–377.

[13] Marian Dörk, Sheelagh Carpendale, Christopher Collins, and Carey Williamson. 2008. Visgets: Coordinated visualizations for web-based information exploration and discovery. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008), 1205–1212.

[14] Tamas E Doszkocs. 1978. AID, an associative interactive dictionary for online searching. *Online Review* 2, 2 (1978), 163–173.

[15] Dorota Glowacka, Tuukka Ruotsalo, Ksenia Konuyshkova, Samuel Kaski, Giulio Jacucci, et al. 2013. Directing exploratory search: Reinforcement learning from user interactions with keywords. In *Proceedings of the 2013 international conference on Intelligent user interfaces*. ACM, 117–128.

[16] Brynjar Gretarsson, John O'donovan, Svetlin Bostandjiev, Tobias Höllerer, Arthur Asuncion, David Newman, and Padhraic Smyth. 2012. Topicnets: Visual analysis of large text corpora with topic modeling. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 2 (2012), 1–26.

[17] Qi He, Daxin Jiang, Zhen Liao, Steven CH Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. 2009. Web query recommendation via sequential query prediction. In *2009 IEEE 25th international conference on data engineering*. IEEE, 1443–1454.

[18] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[20] K Sparck Jones, Steve Walker, and Stephen E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments. *Information processing & management* 36, 6 (2000), 779–840.

[21] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*. ACM, 387–396.

[22] Mika Käki. 2005. Findex: search result categories help users when document ranking fails. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 131–140.

[23] Antti Kangasrääsiö, Yi Chen, Dorota Głowacka, and Samuel Kaski. 2016. Interactive Modeling of Concept Drift and Errors in Relevance Feedback. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. ACM, 185–193.

[24] Diane Kelly, Karl Gyllstrom, and Earl W Bailey. 2009. A comparison of query and term suggestion features for interactive searching. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 371–378.

[25] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[26] Jürgen Koenemann and Nicholas J Belkin. 1996. A case for interaction: A study of interactive information retrieval behavior and effectiveness. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 205–212.

[27] Saar Kuzi, Anna Shtok, and Oren Kurland. 2016. Query expansion using word embeddings. In *Proceedings of the 25th ACM international on conference on information and knowledge management*. ACM, 1929–1932.

[28] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. 1188–1196.

[29] Matteo Lissandrini, Davide Mottin, Themis Palpanas, and Yannis Velegrakis. 2020. Graph-Query Suggestions for Knowledge Graph Exploration. In *Proceedings of The Web Conference 2020*. 2549–2555.

[30] Shixia Liu, Michelle X Zhou, Shimei Pan, Yangqiu Song, Weihong Qian, Weijia Cai, and Xiaoxiao Lian. 2012. Tiara: Interactive, topic-based visual text summarization and analysis. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 2 (2012), 1–28.

[31] Chao Ma and Bin Zhang. 2018. A New Query Recommendation Method Supporting Exploratory Search Based on Search Goal Shift Graphs. *IEEE Transactions on Knowledge and Data Engineering* 30, 11 (2018), 2024–2036.

[32] G. Marchionini. 2006. Exploratory search: from finding to understanding. *Commun. ACM* 49, 4 (2006), 41–46.

[33] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2012. Citeology: visualizing paper genealogy. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*. ACM, 181–190.

[34] Alan Medlar and Dorota Glowacka. 2018. How Consistent is Relevance Feedback in Exploratory Search?. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1615–1618.

[35] Alan Medlar, Kalle Ilves, Ping Wang, Wray Buntine, and Dorota Glowacka. 2016. PULP: A system for exploratory search of scientific literature. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 1133–1136.

[36] Alan Medlar, Joel Pyykkö, and Dorota Glowacka. 2017. Towards Fine-Grained Adaptation of Exploration/Exploitation in Information Retrieval. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces* (Limassol, Cyprus) *(IUI '17)*. Association for Computing Machinery, New York, NY, USA, 623–627. https://doi.org/10.1145/3025171.3025205

[37] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[38] Mandar Mitra, Amit Singhal, and Chris Buckley. 1998. Improving automatic query expansion. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 206–214.

[39] Saeed Mohajeri, Hamman W Samuel, Osmar R Zalane, and Davood Rafiei. 2016. BubbleNet: An innovative exploratory search and summarization interface with applicability in health social media. In *2016 International Conference on Digital Economy (ICDEc)*. IEEE, 37–44.

[40] Atsushi Otsuka, Yohei Seki, Noriko Kando, and Tetsuji Satoh. 2012. QAque: faceted query expansion techniques for exploratory search using community QA resources. In *Proceedings of the 21st International Conference on World Wide Web*. 799–806.

[41] Pearl Pu, Li Chen, and Rong Hu. 2011. A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 157–164.

[42] Joseph John Rocchio. 1971. Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing* (1971), 313–323.

[43] Tuukka Ruotsalo, Giulio Jacucci, and Samuel Kaski. 2020. Interactive faceted query suggestion for exploratory search: Whole-session effectiveness and interaction engagement. *Journal of the Association for Information Science and Technology* (2020).

[44] Mark Sanderson and Bruce Croft. 1999. Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 206–213.

[45] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge.

[46] Milad Shokouhi. 2013. Learning to personalize query auto-completion. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 103–112.

[47] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 553–562.

[48] Ryen W White, Mikhail Bilenko, and Silviu Cucerzan. 2007. Studying the use of popular destinations to enhance web search interaction. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 159–166.

[49] Ryen W White and Resa A Roth. 2009. Exploratory search: Beyond the query-response paradigm. *Synthesis lectures on information concepts, retrieval, and services* 1, 1 (2009), 1–98.

[50] Hamed Zamani and W Bruce Croft. 2017. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 505–514.

[51] Jian Zhao, Christopher Collins, Fanny Chevalier, and Ravin Balakrishnan. 2013. Interactive exploration of implicit and explicit relations in faceted datasets. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2080–2089.

[52] Jianling Zhong, Weiwei Guo, Huiji Gao, and Bo Long. 2020. Personalized Query Suggestions. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1645–1648.