Enriching News Articles with Related Search Queries

David Carmel*
Amazon Research
dacarmel@amazon.com

Yaroslav Fyodorov
Yahoo Research
yfyodorov@verizonmedia.com

Saar Kuzi†
University of Illinois
skuzi2@illinois.edu

Avihai Mejer
Yahoo Research
amejer@verizonmedia.com

Fiana Raiber
Yahoo Research
fiana@verizonmedia.com

Elad Rainshmidt
Yahoo Research
eladr@verizonmedia.com

ABSTRACT

Enriching the content of news articles with auxiliary resources is a technique often employed by online news services to keep articles up-to-date and thereby increase users' engagement. We address the task of enriching news articles with related search queries, which are extracted from a search engine's query log. Clicking on a recommended query invokes a search session that allows the user to further explore content related to the article. We present a three-phase retrieval framework for query recommendation that incorporates various article-dependent and article-independent relevance signals. Evaluation based on an offline experiment, performed using annotations by professional editors, and a large-scale online experiment, conducted with real users, demonstrates the merits of our approach. In addition, a comprehensive analysis of our online experiment reveals interesting characteristics of the type of queries users tend to click and the nature of their interaction with the resultant search engine results page.

ACM Reference Format:
David Carmel, Yaroslav Fyodorov, Saar Kuzi, Avihai Mejer, Fiana Raiber, and Elad Rainshmidt. 2019. Enriching News Articles with Related Search Queries. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313588

1 INTRODUCTION

Enriching news articles with auxiliary content that is extracted from rich and diverse sources is a popular trend used by Web news services to improve user engagement. Content enrichment solutions provide an opportunity to bring relevant knowledge, meaning, and added value to the content consumed. One such approach exposes the user to additional related news articles [30]. Other popular forms are based on user-generated content (UGC), where users are empowered to comment on an article, or to reply to others' comments, generating lively discussion threads [31]. Relevant tweets and blog posts can further enrich an article's content [20], providing complementary and sometimes controversial points of view.

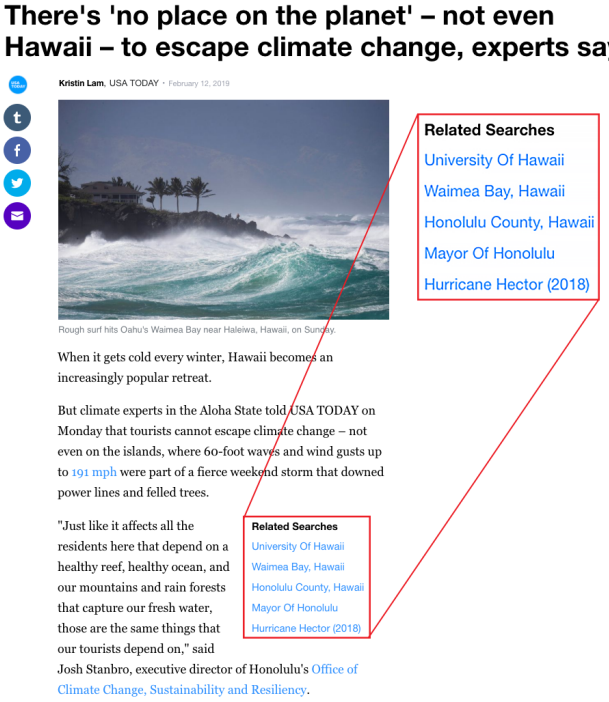

Figure 1: Example of recommended search queries.

These auxiliary sources provide a dynamic and continuously updated content stream, focused on the article's topic. This keeps the article relevant and fresh, thereby deepening users' engagement with the news service.

In this work we address the task of enriching news articles with related search queries that are extracted from a search engine's query log [5]. A search session is invoked when a user clicks on one of the recommended queries, and the subsequent search engine results page (SERP) exposes the user to supplementary content. An example is shown in Figure 1.

There are several requirements that we consider important for the recommended queries to satisfy. First and foremost, the queries should be related to the article's content. We consider the following two notions of query relatedness. The queries might focus on one of the topics covered in the article, allowing the user to further explore that specific topic. Such queries can potentially help users discover different angles of a story. Alternatively, recommended

*Part of this research was conducted while the author was working at Yahoo Research.
†Part of this research was conducted while the author was an intern at Yahoo Research.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313588

queries could expose new and interesting aspects related to the topic, which are not explicitly covered in the article [5]. Given that our task is to enrich news articles, only very fresh and emerging queries are considered relevant. Freshness, our second key requirement, is especially important when dealing with breaking news articles that discuss trending stories [11]. Privacy is another important requirement. Because the recommended queries are extracted from a query log, they must respect the privacy of the users who issued them, without exposing any personal or sensitive information [18]. In addition, queries should respect the article's tone and sentiment. For example, it would be inappropriate to recommend queries related to travel deals to Hawaii for an article dealing with a plane crash in Hawaii. Similarly, a query about buying sneakers for boys is an inappropriate recommendation for an article describing a police search for a missing boy. Like any other UGC, queries extracted from the log might be offensive or malformed [13, 34, 35]. We should avoid presenting such queries to users. Finally, although this is not our main focus, we can prioritize queries that can contribute to the overall monetization of the system while also improving user engagement.

We propose a three-phase retrieval approach to address the task. In the first phase, an initial query list is retrieved in response to the article using a limited set of features. In the second phase, a rich set of features are used to re-rank the list retrieved in the previous phase. The weights of the features are learned using a learning-to-rank approach [23]. We consider both article-dependent features, such as the textual and semantic similarity between the query and the article's title, as well as article-independent features, such as the query length and the number of distinct users who submitted the query. In the third phase, two types of filters are employed. One of the filters is used to improve the quality of the recommendations by removing semantically unrelated queries. The other filter is used to diversify the list of recommended queries by removing similar queries that express the same information need.

We conducted an offline evaluation using a dataset manually created by professional editors, as well as a large-scale online experiment with real users. The results of the two experiments demonstrate the merits of the proposed three-phase retrieval approach. A deeper analysis of our results reveals interesting insights regarding the characteristics of the queries users tend to click as well as the nature of the users' interaction with the resultant SERP . For example, we found that users are more likely to click on longer queries that are expressed as questions. In addition, we found that users who clicked on recommended queries are more likely to explore images and videos, and are less likely to click on organic results ("blue links").

2 RELATED WORK

News articles are often augmented with related social media content, including tweets and blogs [8], encyclopedic knowledge [25], and related search queries [5]. In what follows we cover some of the works most relevant to ours.

Mihalcea et al. [25] introduced the use of Wikipedia as a resource to automatically enrich a text with links to encyclopedic knowledge. Given an input text, important concepts are identified and automatically linked to the corresponding Wikipedia pages. Cheng et al. [9] showed that user queries are often triggered by previously consumed content. The query in such scenarios might be better served by a search engine that uses the originating page as contextual information. The authors used multiple features in a learning-to-rank approach for query recommendation. Some of the features are based on the textual similarity between the query and the page. Others are extracted from a page-query bipartite graph that was constructed using historical query logs. The main limitation of this approach is that it is more effective for popular pages that were visited many times by many different users, making it inherently inapplicable for fresh content such as news articles. Kong et al. [19] explored the reasons users initiate a search. The authors conducted an in-depth analysis using a large-scale query log, studying the relations between previously consumed content and the triggered queries. A feature-based approach is proposed to predict which queries a user is likely to issue after visiting a particular page. The features are computed based on the textual match between the query and the visited page, the user's search history, and the entities appearing on the page.

Bordino et al. [5] addressed a task similar to ours. An article was represented using a set of Wikipedia entities that were extracted from it. A random walk on an entity-query graph, which is based on a long-history query log, was used to identify additional related entities and queries. In a following work, Bordino et al. [6] studied the question of what makes a result serendipitous, i.e., encourages users to explore further. For this purpose, the content of an entity graph was further enriched with metadata about the sentiment, writing quality, and topical category of the text surrounding an entity. The authors investigated whether entities that encompass more emotion provide better results. Their findings show that such entities are not enough to draw the user's attention, suggesting a more complex relationship between topics and entities in serendipitous search. While the entity-query graph provides a very appealing framework for content enrichment with related queries, it does not suit our news-article enrichment task, since only very fresh and emerging queries related to the article are considered relevant in our case. Indeed, our experiments show that queries with decreasing popularity are less suitable for recommendation in our case. On the other hand, recommending serendipitous queries for content enrichment might be very appealing in our setting. We leave this direction for future work.

Another related direction is automatic question generation from some form of input [14, 29]. Such techniques focus primarily on generating questions that can be answered by the given text, while our focus is on queries that enhance and elaborate the article's content. We note, however, that such generation techniques, modified for content enhancement, might be a suitable complementary approach that deserves further investigation. Synthetic query generation is another complementary approach [22]. Queries in the work by Lee et al. were generated from text passages selected by the user rather than identified in a query log. No emphasis was put on freshness.

3 RETRIEVAL FRAMEWORK

Given an article r and a corpus of queries Q, our goal is to produce a highly precise ranking of queries that fulfill several requirements.

The queries should: be relevant to the article, address recent events, disclose nothing about the user who issued them, respect the tone of the article, be non-offensive, and be possibly monetizable. To this end, we take a three-phase retrieval approach. In the first phase, an initial query list Q_{init} is retrieved in response to the article using a limited set of features. In the second phase, a rich set of features is used for re-ranking the n most highly ranked queries in Q_{init}. Additional refining filters are applied in the third phase and the top-k ($k < n$) recommendations are presented to the user. We now present further details for the three retrieval phases. Details regarding the creation of the query corpus are provided in Section 4.1.1.

3.1 First Phase: Initial Ranking

We represent each query $q \in Q$ using the query's raw text (q_T), a concatenation of the search result titles previously retrieved in different searches in which the same query was issued (q_R), and the entities identified in the query's text (q_E). Similarly, an article r is represented using its title (r_T); body (r_B); summary (r_S), which contains a subset of the top-θ terms selected from the article's body according to their TF.IDF values; and the set of entities identified in the article's title and body (r_E). Additional details about the entity-linking tools used are provided in Section 4.

The initial query retrieval is based on the similarity between the different texts representing the article and query, as well as a set F of article-independent query-quality measures:

$$Score_{\text{Init}}(q, r) \stackrel{def}{=} \sum_{X \in \{T,R\}; Y \in \{T,S\}} \lambda_{XY} Sim(q_X, r_Y) + \lambda_E Sim(q_E, r_E) + \sum_{f \in F} \lambda_f f(q); \quad (1)$$

$Sim(\cdot,\cdot)$ is an inter-text similarity measure described below; λ_{XY}, λ_E and λ_f are free parameters; $f(q)$ is the score assigned to q by a quality measure f. Inspired by the state-of-the-art performance of the Sequential Dependence approach in the Markov Random Field model [24], to compute the similarity between texts x and y, we use

$$Sim(x, y) \stackrel{def}{=} \lambda_T Sim_{\text{T}}(x, y) + \lambda_U Sim_{\text{U}}(x, y), \quad (2)$$

where $Sim_{\text{T}}(x, y)$ and $Sim_{\text{U}}(x, y)$ are the similarity values, computed using BM25 [28], between terms and unordered term-pairs (within a 5-term window), respectively, in x and y; λ_T and λ_U are free parameters.

We use four measures $f \in F$ to quantify different article-independent properties of the query. Popular queries that were issued by many users might be more interesting. Hence, we count the number of distinct users who issued the query during the last three weeks (**DistinctUsers**). We also examine the interaction of users with the SERP retrieved in response to the query: rich interaction with the SERP is a potential indicator of how attractive and informative it is, and hence of the high query quality. We compute the average number of cards[1] [32] on the page (**Cards**), the average number of clicked URLs after the query was issued (**ClicksPerUser**), and the clickthrough rate (CTR) of ads presented on the SERP (**AdsCTR**). Additional measures are integrated via our learning-to-rank approach in the second retrieval phase, as detailed in the next section.

[1]Cards are results presented directly on the SERP to spare a user an additional click [32].

Table 1: Features used in the learning-to-rank algorithm.

	Feature	Description
Article Dependent	EntitySim	$Sim(q_E, r_E)$
	TitleSim	$Sim(q_X, r_T)$ for $X \in \{T, R\}$
	SummarySim	$Sim(q_X, r_S)$ for $X \in \{T, R\}$
	BodySim	$Sim(q_X, r_B)$ for $X \in \{T, R\}$
	Overlap	Jaccard similarity between q_T and r_T
	W2V	Semantic similarity using word2vec
	ESA	Semantic similarity over Wikipedia concepts
Article Independent	QueryLength	Number of query terms
	AvgIDF , MaxIDF	Average and maximum IDF of query terms
	DistinctUsers	Number of distinct users that submitted the query
	RPM	Revenue per mille (thousand query submissions)
	AdsCTR	Clickthrough rate of ads presented on the SERP retrieved for the query
	ClicksPerUser	Number of clicked URLs divided by the number of users that issued the query
	DistinctClicked	Distinct number of clicked URLs
	DecDistinctUsers	Decaying sum of the number of users that submitted the query during the last three weeks
	AvgUsers , StdvUsers	Average and standard deviation of the number of users that submitted the query during the last three weeks
	AvgUserHistory	Average number of users who issued the query in a specific time period during the last three weeks

3.2 Second Phase: Learning to Rank

We re-rank Q_{init}, the initial query list retrieved in the first phase, to improve its quality. The re-ranking model was trained using a learning-to-rank (**LTR**) approach [23], details of which are provided in Section 4.1.4. Each pair of an article and query is represented as a feature vector. As in Equation 1, we consider two types of features: article dependent and article independent. The features[2] are listed in Table 1.

3.2.1 *Article-Dependent Features.*

Textual Similarities. We use several textual similarity estimates between the different texts representing the article and query. One group of features, which we also used in the initial retrieval phase, includes the BM25 similarity between the entities identified in the article and query (**EntitySim**). We use two entity-linking tools to identify entities in texts (see Section 4 for details). Hence, considering all possible combinations of the entity sets extracted by the two tools, we get four features. Another group of features includes the similarities between the query and the article's title (**TitleSim**), body (**BodySim**), and summary (**SummarySim**), where the query is represented using either its raw text or by a concatenation of its result titles. While a single similarity estimate (BM25) was

[2]We applied Z-score normalization over feature values, which we found to be more effective than applying Min-Max normalization or not normalizing at all.

used to instantiate Equation 2 in the first retrieval phase, here we use three estimates: TF.IDF , BM25 [28], and LM [21] (language models with Jelinek-Mercer smoothing). We get 18 features: 3 similarity estimates (TF.IDF , BM25 , and LM) × 3 texts representing the article (r_T, r_S, and r_B) × 2 texts representing the query (q_T and q_R). In addition, we compute the Jaccard similarity between the query's raw text and the article's title (**Overlap**), where texts are represented by stemmed unigrams after stopword removal.

Semantic Similarities. We use two semantic similarity measures to address the potential vocabulary mismatch between the query and article, which cannot be captured by the textual similarity measures defined above. The first is based on Explicit Semantic Analysis (**ESA**) [12], in which the query's text (q_T) and the article's title (r_T) are embedded into a Wikipedia concept space. Specifically, each text is represented as a weight vector over all Wikipedia concepts (roughly 6M concepts), where the weight of a concept is the similarity value (computed using BM25) between the text and the concept's Wikipedia page. The second measure is based on a word2vec (**W2V**) representation [26]. We train a W2V model using Wikipedia, and represent each text using the centroid of its terms' W2V vectors. We use cosine similarity to compare the query and corresponding article representations for both ESA and W2V . While ESA captures semantic relatedness through a comparison over a concept space, W2V captures the same notion through a comparison of the context of the two texts. Using more complicated entity embedding techniques is an interesting future direction we intend to explore [4].

3.2.2 Article-Independent Features. As already noted, we use multiple article-independent measures to quantify different properties of the query. The first group of measures are computed based on the query's text: query length (**QueryLength**), and average (**AvgIDF**) and maximum (**MaxIDF**) IDF of the query terms. We assume that longer queries with higher IDF values are more interesting. The DistinctUsers measure, which was also used in the first retrieval phase, quantifies query popularity. In addition, high-revenue queries might be favored not only to increase the system's revenue, but also because they are likely to be popular. Hence, we also consider the revenue per thousand query submissions on resultant SERP s (**RPM**). To capture the informativeness and attractiveness of the resultant SERP , we compute several features based on the interaction of users with the SERP: the CTR on Ads (AdsCTR), the average number of clicks performed per user (ClicksPerUser), and the distinct number of clicked URLs (**DistinctClicked**). Finally, we use several features to quantify the trendiness of a query. Queries that were issued only a few times in the past but became popular lately might be more interesting than those whose popularity has been decreasing over time. Hence, we examine the submission history of the query per hour and compute the average (**AvgUsers**) and standard deviation (**StdvUsers**) of the number of distinct users who submitted the query during a one-hour sliding time window over the past three weeks. We also compute a weighted sum of the number of users who submitted the query, where the value in each hour is multiplied by a decaying factor (**DecDistinctUsers**). We experimented with several decaying factors and ended with the following three: Light (0.996), Moderate (0.993), and Heavy (0.986). These provided different decay patterns in user interests for a large portion of the queries. The higher the decaying factor, the lower the difference between weights assigned to different hours. For example, the weight associated with the number of distinct users who issued the query (a day, a week, two weeks) ago will be (0.908, 0.510, 0.260), (0.845, 0.307, 0.094), and (0.713, 0.094, 0.009), for the Light , Moderate , and Heavy decaying factors, respectively. In addition, we divide the three-week hourly submission history into five equal time periods and compute the average number of distinct users who submitted the query in each period (**AvgUserHistory**), resulting in five features.

3.3 Third Phase: Query Filtering

We apply two types of filters on the query ranking produced in the second phase. The first is used to further improve the quality of the recommended queries. The filter is based on the ESA and W2V representations of the query's text and article's title. We filter out queries for which the cosine similarity between their ESA or W2V representation and the corresponding representation of the article is below a given threshold, τ_{ESA} or τ_{W2V}, respectively.

The second filter is used to diversify the list of queries presented to the user by eliminating highly similar queries that express the same information need. To this end, we scan the query ranking from top to bottom. If the similarity between the next query in the ranking and all the preceding queries is below a given threshold τ_{JAC}, the query is considered novel and is added to the final result list; otherwise, it is ignored and the next query in the list is examined. The similarity between queries is measured using Jaccard, where each query is represented via the union of its unigrams and bigrams.

4 EVALUATION

4.1 Experimental Setting

4.1.1 Query Extraction and Indexing. To guarantee that all the articles will be associated with the latest queries, we update the queries dataset (and index) on a regular basis. This is especially important for breaking news articles, where relevant queries start to appear very quickly after the event occurs. New queries that hit the log of the Yahoo commercial Web search engine are identified every four hours and added to the dataset. To ensure that private information about the users who issued these queries will not be revealed, we only consider queries that were searched at least 10 times (by different users) in a day and at least twice during the last 4 hours. Additionally, we filter out offensive and adult queries using blacklists manually curated by in-house professional editors. Finally, we filter out non-Latin queries and queries longer than 10 terms.

For each instance of a query submission, we collect the query's raw text, the top-10 results ("blue links") presented on the SERP and clicks on these results, the displayed ads and the clicks on these ads, and the cards (direct answers) presented on the SERP . We then combine the queries based on their raw text (after lower-casing, but before stemming) over a history of three weeks. Hence, each entry in our query dataset contains a unique query with its top-100 aggregated search result titles scored by their frequency, the number of distinct users who submitted the query, the clickthrough rate for ads, revenue per 1000 submissions (RPM), and the submission history per hour during the last 3 weeks. The 3-week restriction

Table 2: Examples of article titles and queries labeled relevant ('+') and non-relevant ('−') by the editors.

Teachers in Philadelphia Plan a Black Lives Matter Week
+ Black Lives Matter Movement
− Lesson Plans for Teachers
Oregon Man Finds Year-Old Lottery Ticket and Wins $1 Million 8 Days Before it Expired
+ Oregon Lottery Winning Numbers
− Million Man March
3 Colors You Should Always Wear in a Job Interview
+ What to Say in an Interview
− Wear TV 3

ensures that only relatively fresh queries will be served. Our query dataset, on average, contains approximately 500K queries.

4.1.2 Offline Evaluation. Since our query dataset is constantly updated, we fixed a dataset of 411,646 queries collected on April 25, 2017, and randomly sampled a set of 3082 news articles published on this date on Yahoo News. By varying the free-parameter values in Equation 1, we collected a large set of queries to be judged by in-house professional editors. The results were pooled together and up to 30 queries were selected per article, based on their aggregated score across the different free-parameter variations. Binary relevance labels were assigned by the editors; each query was annotated by one editor. The editors were instructed to label queries that satisfied all the requirements discussed in Section 1 as relevant. Table 2 lists a few examples of queries labeled relevant and non-relevant by the editors. We randomly sampled 250 articles for testing. Two thirds of the remaining articles were used to train the learning-to-rank model and one third to set parameter values.

We are mostly focused on high precision. Hence, to estimate the effectiveness of our approach, we used the precision of the top-3 (p@3) and top-5 (p@5) results, the mean average precision of the top-5 results (MAP), and the mean reciprocal rank of the top-5 results (MRR). In addition to the above-mentioned metrics, we also study in Section 4.2.2 the effectiveness of our approach in terms of the overall expected system revenue. Statistically significant differences in performance were determined using the two-tailed paired t-test at a 95% confidence level.

4.1.3 Online Evaluation. In addition to the offline evaluation performed using the dataset described above, we also conducted an online experiment involving two parts. Our goal in the first part was to collect data for training a learning-to-rank model using explicit user feedback rather than editorial judgments. During a period of 40 days (December 22, 2017 until January 30, 2018) our recommended queries were presented to all Yahoo News users. The users were from the United States and used only desktop devices. The initial ranking (first retrieval phase) and the filters (third retrieval phase) were applied with the free-parameter values learned using our offline dataset. The learning-to-rank approach was not employed in this part of the experiment. After applying the two filters, the retrieval score assigned to queries in the first phase was linearly interpolated with the query's RPM value using a free parameter ρ; we set $\rho = 0.3$ following an analysis performed using the offline dataset, as shown in Section 4.2. This linear interpolation allowed us to control the balance between query relevance and system revenue.

The collected feedback was used to train two learning-to-rank models so as to overcome the bias incurred in users' clicking decisions.[3] (Additional analysis about the position bias is provided in Section 4.2.8.) The first, henceforth **Skip** , was learned using an approach similar to "Click > Skip Above" [17]. Specifically, the training data consisted of all the searches (approximately 35K) in which a "skip" occurred: a user clicked on a query in rank $i > 1$, and skipped (did not click on) the queries ranked above it. All the queries ranked above rank i, and the unclicked queries ranked below it, were considered non-relevant. The query in rank i and all the clicked queries ranked below it were considered relevant. The second model, **Balanced** , was trained using the data collected to train the Skip model as well as searches in which the highest ranked query was clicked. For these searches, all the clicked queries were considered relevant, while the non-clicked were deemed non-relevant. We note that the number of searches in which a user clicked on the highest ranked query was much higher than those in which a skip occurred. Hence, we randomly subsampled the data to get a balanced training set.

In the second part of the experiment, we studied the effectiveness of our approach via A/B testing [15]. We simultaneously launched five buckets each of which served 5% of Yahoo News traffic; users were randomly split between buckets. This experiment lasted 10 days. One of the buckets corresponded to the model used in the first part of the experiment (**Control**). Two additional buckets corresponded to the two models described above. Finally, as was the case in the first part of our experiment, we also linearly interpolated the retrieval score of a query with its RPM value ($\rho = 0.3$), which resulted in two additional buckets: **Skip+RPM** and **Balanced+RPM** .

Since our main focus is on improving user engagement, we use the clickthrough rate on the recommended queries (CTR) to estimate the effectiveness of our approach. In addition, since any click on one of the recommended queries initiates a standard search, we also report the total system revenue per thousand sessions that were initiated by such queries (Revenue) and the query reformulation rate (Reformulation) on the SERP . Low reformulation rate presumably implies that the user was satisfied with the results without having to refine the query, attesting to the effectiveness of the recommended query. We report the improvement in percentages with respect to the Control model.

4.1.4 Additional Implementation Details. We used Apache Lucene[4] for experiments. We applied porter stemming and removed stopwords using Lucene's English stopword list. We used an online variant of RankSVM with a linear kernel [16] to train the re-ranking model. The algorithm uses pairwise training examples and iteratively updates the model via the AROW update step [10]. The hyper-parameters were set to optimize p@5 over the validation set. To identify entities in a text, we applied two in-house entity-linking tools. Each entity was represented using a Wikipedia entity ID

[3] Two thirds of the data was used to train the learning-to-rank models and one third to set parameter values.
[4] http://www.lucene.apache.org

Table 3: Offline results. 'i' and 'l' mark statistically significant differences with Init and Init+LTR+ESA+W2V , respectively. The best result in a column is underlined.

	p@3	p@5	MAP	MRR
Init	44.3_l	44.6_l	58.0_l	62.3_l
Init+ESA	57.5^i_l	55.1^i	73.0^i_l	76.9^i_l
Init+W2V	55.9^i_l	52.4^i_l	71.2^i_l	73.8^i_l
Init+ESA+W2V	60.3^i	$\underline{57.0}^i$	76.2^i_l	78.3^i_l
Init+LTR	54.4^i_l	45.8_l	71.5^i_l	74.8^i_l
Init+LTR+ESA	61.9^i	52.4^i_l	77.4^i	80.3^i
Init+LTR+W2V	59.1^i_l	51.4^i_l	76.9^i_l	79.6^i_l
Init+LTR+ESA+W2V	$\underline{62.5}^i$	55.8^i	$\underline{78.6}^i$	$\underline{80.8}^i$

along with a score reflecting the confidence of the tool. To compute $Sim(q_E, r_E)$, the IDF of each entity was scaled using the confidence score assigned to it by the entity-linking tool [27].

We mapped the article-independent feature values into a five-dimensional binary feature vector by dividing the feature's range into equal-sized bins. The value in a bin equals one if the feature's value falls in that bin's interval. This binary transformation resulted in better performance compared to using the raw feature values. We applied a logarithmic transformation to the AdsCTR , ClicksPerUser and DistinctClicked features since the ranges of these features were rather large; binary transformation was not applied in this case.

Free-Parameter Values. As mentioned above, free-parameter values were set using a validation set in the online and offline experiments. The number of queries initially retrieved using Equation 1 per article was $n = 25$; $k = 5$ queries were presented to users. The number of terms used to create the article's summary was $\theta = 10$. In Equation 1, we set $\lambda_{TT} = \lambda_{TS} = 10$, $\lambda_{RT} = \lambda_{RS} = 3.5$, $\lambda_E = 2.5$ and $\lambda_f = 0.25$. In Equation 2, we set $\lambda_T = 0.8$ and $\lambda_U = 0.2$. The article's summary (r_S) and the entities identified in the query (q_E) and article (r_E) are unordered sets of tokens, hence when computing the similarity with these texts we set $\lambda_T = 1$ and $\lambda_U = 0$. The free parameters of BM25 ($k1$ and b) and the Jelinek-Mercer smoothing parameter were set to standard values: 1.2, 0.75 and 0.1, respectively. We used W2V 's Continuous Bag-of-Words (CBOW) model. The dimension of the vectors and the window size were set to 300 and 5, respectively; all other parameters were set to default values. In the ESA and W2V filters, we set $\tau_{ESA} = 0.15$ and $\tau_{W2V} = 0.6$. To filter out similar queries, we set $\tau_{JAC} = 0.66$.

4.2 Experimental Results

4.2.1 Offline Results. The results of our offline evaluation are presented in Table 3. We study the effectiveness of the query rankings produced in the first (Init), second (Init+LTR), and third (Init+LTR+ESA+W2V) retrieval phases. We also examine the contribution of the LTR approach and the semantic filters (ESA and W2V) when integrated in different stages of our framework. Note that the similar-queries filter, presented in Section 3.3, was applied to all query rankings. We can see that re-ranking the query list retrieved in the first phase using our LTR model yields statistically significant performance improvements over the initial ranking (Init+LTR vs. Init). Applying the semantic filters (either after the first retrieval phase or after the second phase) substantially improves the performance in both cases. The ESA filter is somewhat more effective than W2V ; however, integrating the two filters leads to further improvements compared to using each alone. Interestingly, Init+ESA+W2V outperforms Init+LTR , further attesting to the clear merits of applying the semantic filters. The best results are almost always attained for Init+LTR+ESA+W2V when considering all three retrieval phases; Init+LTR+ESA+W2V is outperformed only by Init+ESA+W2V for p@5 , but the difference is not statistically significant.

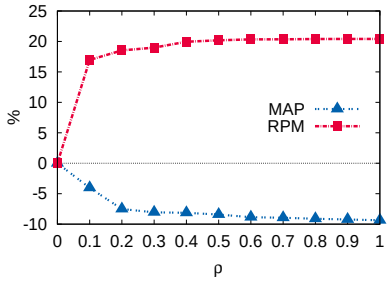

Figure 2: The effect of ρ on MAP and average RPM of the top-five results when interpolating the retrieval score of a query with its RPM value. We report the relative improvement with respect to using the original model Init+LTR+ESA+W2V ($\rho = 0$).

4.2.2 Relevance versus Revenue. After a user clicks on a recommended query, she is taken to a regular SERP where a variety of ads and sponsored links may appear next to organic search results. Such commercial sources of information can increase the overall revenue of the system. Similar to past work on sponsored search [36], we study the tradeoff between recommending queries that are intended to increase user engagement and those aimed at maximizing system revenue. To facilitate the promotion of high-revenue queries, we linearly interpolated the final retrieval score of a query after the third retrieval phase with its RPM value using a free parameter ρ. Figure 2 presents the effect of ρ on MAP and average RPM of the top-five results; the percentages indicate the improvement with respect to using only the original score, i.e., employing the Init+LTR+ESA+W2V model. As ρ increases and more weight is given to the query's RPM value, MAP decreases and the system's expected revenue (RPM) increases. The steepest drop in performance is observed for $\rho \leq 0.3$. For $\rho > 0.3$, MAP continues to drop at a much slower pace.

In Table 4 we show examples of queries recommended by our algorithm for $\rho = 0$, where no weight was given to the query's RPM value, and for $\rho = 0.3$, where a moderate weight was assigned.[5] We see that the queries recommended for $\rho = 0.3$ have a stronger commercial intent. These queries include keywords that users are more likely to use before making a purchase. Conversely, the queries recommended for $\rho = 0$ have a stronger news intent and tend to be more informative. For example, based on a random sample of 280 articles, we found that the terms "sale", "tickets" and "buy" were

[5]We note that in both example articles, the queries recommended for $\rho > 0.3$ were identical to those recommended for $\rho = 0.3$, but they were not necessarily ranked in the same order.

Table 4: Examples of article titles and the top-three recommended queries for different values of ρ. The higher the value of ρ, the more weight is given to the query's RPM value.

	iPhone 9 and X Plus? New iPhone Rumours, Release Dates and Everything We Know
$\rho = 0$	1. iPhone 11 Release Date 2. iPhone Store Near Me 3. iPhone Battery Drains Fast
$\rho = 0.3$	1. Can iPhones Get Viruses 2. iPhone 8 Plus Price 3. Used iPhones for Sale
	Sacred Riana's Weird 'America's Got Talent' Magic Act
$\rho = 0$	1. America Got Talent 2018 2. Courtney America Got Talent 3. America Got Talent Judges
$\rho = 0.3$	1. America Got Talent Tickets 2. NBC America Got Talent 3. America Got Talent 2018

more popular for $\rho = 0.3$. These terms appeared in 6.1%, 5.4% and 4.6% of the recommended queries, respectively, for $\rho = 0.3$ vs. 1.8%, 0% and 1.4% for $\rho = 0$. In contrast, the terms "news", "today" and "twitter" were more popular for $\rho = 0$: they appeared in 9.3%, 5.7% and 2.1% of the queries for $\rho = 0$ vs. 5%, 2.1% and 0% for $\rho = 0.3$.

4.2.3 Online Results. The results of the online evaluation are presented in Table 5. We inspect three metrics quantifying different aspects of our approach: CTR , the user's engagement; Revenue , the system's revenue; and Reformulation , a proxy for the user's dissatisfaction with the search results. Supporting our offline results, we observe substantial improvements in CTR when applying the LTR model in all four buckets; the highest improvement is attained for the Balanced model. Examining Revenue , we see a decrease in revenue for both the Skip and Skip+RPM models, even though the latter explicitly incorporates RPM into the retrieval process. Comparing Balanced with Balanced+RPM , we observe a trend similar to the one we saw in our offline experiment: the CTR is higher for the Balanced model, which uses RPM as a feature in the learning-to-rank model, whereas Revenue is higher for Balanced+RPM , which also explicitly interpolates the RPM value of a query with its retrieval score. The best performance is attained by the Balanced model, which had the highest CTR , second highest Revenue , and lowest Reformulation . Table 6 lists a few examples of queries recommended by this model.

Figure 3 illustrates the change in CTR during the 10-day experiment. As was the case thus far, we report the relative improvement in CTR with respect to the Control model. We first observe that among the 4 considered models, Balanced was the most stable throughout the experiment, especially during the last 6 days. We also see that Balanced is the only model for which the change in CTR was always positive. In contrast, for the other three models, the CTR decreased compared to the Control model at least once during the experiment. These findings, along with the results presented in Table 5, attest to the effectiveness of the Balanced model.

Table 5: Online results. The relative improvement (in percentages) over the Control model. The best result in a row is underlined.

	Skip	Balanced	Skip+RPM	Balanced+RPM
CTR	18.3	<u>22.6</u>	6.5	16.1
Revenue	−23.6	17.4	−23.5	<u>22.4</u>
Reformulation	14.4	<u>−13.5</u>	−7.0	0.6

Table 6: Examples of article titles and the top-three recommended queries by the Balanced model.

Hawaii Officials Plead for Visitors to Keep Travel Plans 1. Is it Safe to Travel to Hawaii Right Now 2. Hawaii Travel Advisory 3. Travel to Hawaii
Prince Harry and Meghan Markle ask U.S. Bishop to Deliver Wedding Address 1. How Old is Meghan Markle and Prince Harry 2. Prince Harry and Meghan Markle Wedding Date 3. How did Meghan Markle Meet Prince Harry
Sears is Closing 40 Stores in 24 States – Here's the List 1. Sears Store Closing List 2018 2. Sears Store Near Me 3. Sears Store Locations

Table 7: Features used in the Balanced model ranked in descending order of attributed importance.

1 − 5	6 − 10	11 − 15	16 − 19
RPM	AvgUsers	DistinctUsers	DistinctClicked
BodySim	QueryLength	EntitySim	MaxIDF
TitleSim	ClicksPerUser	W2V	AdsCTR
Overlap	DecDistinctUsers	AvgUserHistory	ESA
SummarySim	StdvUsers	AvgIDF	

4.2.4 Feature Analysis. We next analyze the contribution of the features and feature groups used in the learning-to-rank approach employed in the second retrieval phase. To this end, we arrange the features in descending order of their weights in learning-to-rank model and assign each feature with a score that is the reciprocal of its rank. For example, the scores of the top-three features are 1, $\frac{1}{2}$ and $\frac{1}{3}$, respectively. These feature scores are then averaged over groups of features as follows. For each of the article-independent features, excluding AdsCTR , ClicksPerUser , and DistinctClicked , we average the scores of their corresponding five binary features (refer back to Section 4.1.4). In addition, for DecDistinctUsers , the scores are also averaged over the three decaying factors and for AvgUserHistory over the five time periods. For the article-dependent features, TitleSim , SummarySim , and BodySim , we average the scores over the three similarity measures (BM25 , TF.IDF and LM)

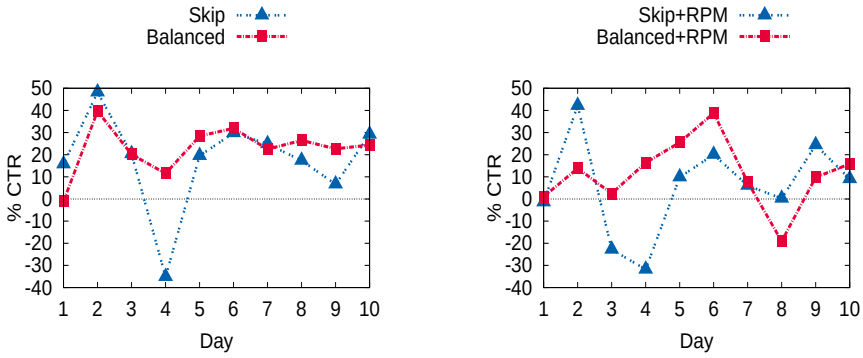

Figure 3: Relative improvement in CTR over time.

and the two texts representing the query (q_T and q_R). Finally, for EntitySim , we average the scores over the four features corresponding to the different combinations of the entity-linking tools.

Table 7 presents the features in descending order of importance for the Balanced model, which is the best performing model among the four we examined. We see that among the top-five features, four (TitleSim , SummarySim , BodySim and Overlap) are estimates of the textual similarity between the article and query, which attests to the importance of the textual similarity features in our task. The semantic similarity features (ESA and W2V), on the other hand, are ranked much lower. This may be because these features are also used to filter out queries in the third retrieval phase, resulting in some redundancy. RPM is the most important article-independent feature, which might explain the lift in Revenue we observed in Table 5 for the Balanced model. AdsCTR is among the least important features. Because this feature was also used in the first retrieval phase, it might be somewhat redundant in the second phase. The AvgUsers , DecDistinctUsers , and StdvUsers features, which quantify the trendiness of the query, are among the top-10, supporting our postulation about the importance of presenting users with very fresh and popular queries. Finally, inspecting the textual similarity features, we found that among the 3 similarity estimates, BM25 , LM , and TF.IDF , BM25 is the most important while LM is the least important. In addition, we found that features measuring the similarity with the query's text (q_T) are less important than those considering the concatenation of the result titles retrieved in response to the query (q_R).

4.2.5 Query Characteristics. To get a better sense of the type of queries users tend to click, we examined the characteristics of all the queries recommended by the five models tested in the second part of our online experiment. We distinguish between clicked and non-clicked queries, where the values extracted per each clicked (non-clicked) query are scaled by the number of times the query was presented and clicked (non-clicked). We performed a similar analysis for queries recommended by each of our five models separately. Here, we scaled the values attained per query by the number of times the query was presented to the users in the corresponding bucket. We consider the following five attributes: (i) Questions : percentage of queries formulated as questions[6]; (ii) Length1 : query length in words; (iii) Length2 : query length excluding stopwords of all queries except those that begin with a question word (iv) SW1 : percentage of stopwords in a stopword list that appear in the query [3]; and, (v) SW2 : percentage of terms in the query that are stopwords[7] [3].

Table 8: Characteristics of queries. Questions : percentage of queries starting with a question word. Length1 : query length in words. Length2 : query length excluding stopwords and queries starting with a question word. SW1 and SW2 : stopword-based measures [3].

	Questions	Length1	Length2	SW1	SW2
Clicked	5.37	4.39	4.00	1.08	6.55
Not Clicked	4.95	3.87	3.56	0.81	5.53
Control	3.47	3.35	3.12	0.60	4.18
Skip	5.93	4.10	3.77	0.89	5.84
Balanced	4.65	3.95	3.64	0.85	5.77
Skip+RPM	6.00	3.97	3.64	0.88	6.06
Balanced+RPM	4.65	3.93	3.62	0.83	5.73

Table 8 presents the results. Comparing clicked versus non-clicked queries, we see that users click on longer queries, which are more likely to contain stopwords and start with a question word. This implies that users prefer queries expressed in a natural language as opposed to those solely containing important keywords, e.g., "how old is prince harry" vs. "prince harry age". The queries recommended by the Control model, for which the lowest CTR was attained, are shorter than those recommended by the other models and are the least likely to contain stopwords or start with a question word. The queries recommended by Skip are longer than those recommended by Balanced , and are more likely to start with a question word. Nevertheless, Balanced outperformed Skip in terms of CTR , indicating that these attributes are important but insufficient in determining query relevance.

[6] We consider queries that start with one of the following question words: who, where, why, when, how, what, which, whose, whom, is, are, do, does, did, was, were, has, have, will and can.
[7] We use Lucene's English stopword list.

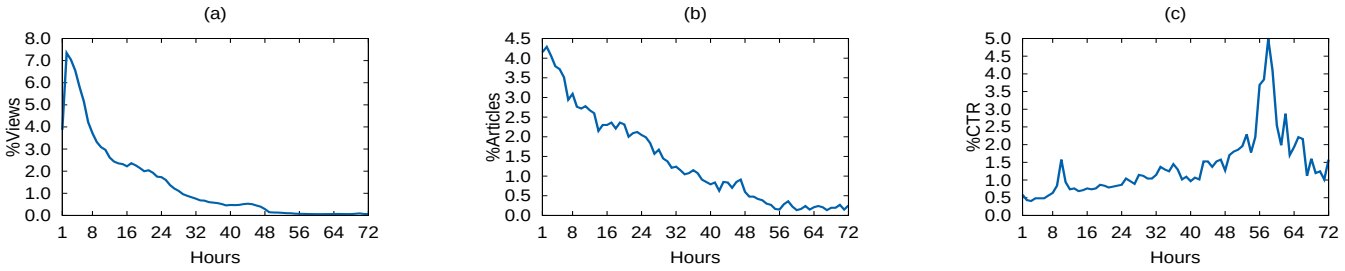

Figure 4: The change in (a) the percentage of article views, (b) the percentage of articles with at least five clicks on recommended queries, and (c) the CTR on the queries as a function of the number of hours that passed since an articles was first viewed.

Table 9: Distribution of clicks on organic results and the various verticals shown on the SERP .

	Balanced	SameQuery	AfterNewsURL	AfterNewsOther
Organic	10.82	36.72	38.60	54.84
News	32.28	22.96	19.45	5.11
Entities	5.63	3.79	7.00	2.96
Images	25.25	10.01	17.26	7.57
Videos	13.92	6.02	8.37	3.69
Ads	10.89	8.11	6.64	20.02
Other	1.21	12.39	2.68	5.81

4.2.6 Recommended versus Native Queries. In Section 4.2.3 we studied the interaction of users with the SERP presented after a recommended query was clicked in terms of Revenue and Reformulation . In what follows, we compare users' interactions with the SERP once they open it either by clicking on a recommended query or by issuing a native query themselves. We extracted all the searches initiated by queries that were recommended by the Balanced model. This was done during a non-consecutive two-day time period in which the model served all Yahoo News users. (These users were from the United States working on desktop devices.) We compared these searches with those invoked by native user queries. Specifically, we consider searches in which the user issued a query, not necessarily after reading a news article; this query was also clicked by users presented with queries recommended by the Balanced model (SameQuery). In addition, we consider searches initiated by a user query that was issued within the first five minutes after the user read a news article, where no other actions were performed in between. We distinguish between searches in which at least one (AfterNewsURL) or none (AfterNewsOther) of the query terms appeared in the originating article's URL; the former queries are more likely to be related to the news article. We excluded searches that were initiated by 40 different navigational queries, such as "facebook", "youtube", and "google". Since we recommend fresh queries that were added to our dataset recently (e.g., the day before), for the native searches we inspected not only the two days mentioned above, but also the searches that took place a day before, i.e., we inspected four days in total.

In Table 9 we present the click distribution on different result types (verticals) that are shown on the page [2], including Organic (organic "blue links" results), News (news articles), Entities (mostly presenting information about persons and locations [7]), Images , Videos and Ads . We can see that among the four types of searches we inspected, users arriving to a search page after clicking on a recommended query are more likely to examine additional news articles that are potentially related to the previously read article. These users are also more likely to click on images and videos. This might suggest that recommended queries elicit users to seek auxiliary visual information to further deepen their knowledge and satisfy their curiosity. Similar to Kong et al. [19], we see that users are more likely to seek information about various entities after reading a news article (see Balanced and AfterNewsURL vs. SameQuery and AfterNewsOther).

Although the SameQuery searches were invoked by queries that were also recommended by the Balanced model, they are quite different. For example, in the SameQuery searches, users are more inclined to click on organic results. This is also the case for the AfterNewsURL and AfterNewsOther searches, which are invoked following an interaction with a news article. We also see that the Ads vertical receives the highest percentage of clicks in the AfterNewsOther searches. Presumably queries leading to these searches are less likely to be related to the originating news article and are therefore more likely to have some commercial intent.

4.2.7 Article Freshness. As already noted, time plays a key role when dealing with news, especially breaking news, articles. In this work, we have taken a number of steps to promote fresh and emerging queries. First, the query index was updated on a regular basis. Second, only queries that were issued in the past three weeks were kept in the index. Third, several features that quantify the trendiness of queries were used in the learning-to-rank model (i.e., AvgUsers , StdvUsers , DecDistinctUsers and AvgUserHistory).

While the emphasis so far was on query freshness, in what follows we focus on article freshness. In Figure 4 we examine various aspects of the change in the users' behavior when interacting with news articles, and our recommended queries in particular, as a function of the time that has passed from the moment an article was first viewed. This analysis is based on logs collected during the second part of the online experiment. Here, we do not distinguish between the different buckets. At any given time, we only consider articles with at least five clicks on the recommended queries. All

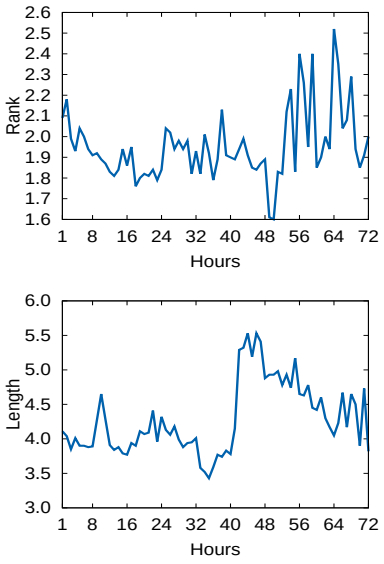

Figure 5: The change in the average click rank and average query length as a function of the number of hours that passed since an articles was first viewed.

the numbers are sum-normalized with respect to the total attained during the three-day time period

We start by studying the percentage of views an article received in an hour over the course of the three days. We observe a sharp decline in the number of views. For example, we found that on average an article reaches 50.1% of its views during the first ten hours and 81.2% during the first day. This finding demonstrates the relatively short lifespan of news articles and the fast pace of news consumption. We next examine the percentage of articles with at least five clicks on the recommended queries in an hour. Here again we observe a sharp decline, which testifies to the need for a fast algorithm that can produce recommendations immediately after the article is published. Inspecting the CTR (sum normalized over the three-day time period), we see that users start clicking on recommendations from the very first moment the article is viewed, i.e., already in the first hour. We observe a steady lift in CTR during the first two days, and a rather sharp increase during the third day. We postulate that the few articles that still get clicks during the third day discuss compelling and ongoing events, in which case users are more inclined to explore further. Indeed, we found that the CTR of articles with clicks in the first day only was 3.4 times lower than that of articles with clicks in all three days.

In Figure 5 we examine the properties of the clicked queries as a function of article freshness. We notice that the average rank of the clicked queries is rather stable in the first two days (average: 1.91; standard deviation: 0.09) and is slightly higher during the third day (average: 2.03; standard deviation: 0.24). A similar observation can be made about query length that increases during the third day. We attribute these changes to the somewhat different nature of the articles that are still being read during the third day and the users' desire for additional related content for such articles.

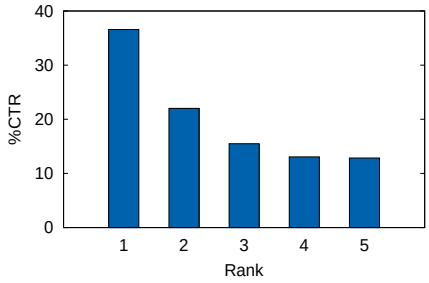

Figure 6: Sum-normalized CTR across the top-five ranks.

For example, we found that the average query length was 3.8 for articles that received clicks during the first day only, and 4.3 for those that received clicks during the entire three-day time period. Integrating the article's freshness into the query recommendation algorithm is an interesting future research direction.

4.2.8 Position Bias. To learn feature weights using user feedback in the second retrieval phase, we experimented with two different training sets so as to overcome the bias incurred in users' clicking decisions [17]. In what follows, we study the severity of the position (trust) bias wherein users tend to click on highly ranked items even if they are less relevant than those ranked lower. In Figure 6 we show the CTR across the five ranks in which queries were recommended during a two-week time period to all Yahoo News users. During this time the top-five queries recommended by the Balanced model were randomly shuffled across ranks. As expected, a rather sharp position bias can be observed: the higher the query is presented, the more likely it is to receive clicks. This finding, along with the differences in performance observed earlier between the Skip and Balanced models, attest to the importance of correctly interpreting user clicks. Testing additional approaches is left for future work [1, 33].

5 CONCLUSIONS AND FUTURE WORK

Content enrichment lets users access information they may not be aware of, such as related articles or relevant facts or events, thereby serving to increase engagement. We addressed the task of enriching news-articles by recommending related search queries that are extracted from a search engine's query log. The three-phase approach we proposed for generating the recommendations is based on a learning-to-rank model that uses various article-dependent and article-independent features. Our approach also integrates several highly effective semantic query filters. Empirical evaluation performed using an offline dataset created by professional editors and a large-scale online experiment with real users attested to the effectiveness of our approach. A deeper analysis of the results revealed interesting characteristics of the types of queries users tend to click and the unique behavior of users after clicking on such queries. For future work we intend to study the effectiveness of our approach when applied to other domains. For example, focusing on specific categories, such as sports, finance or scientific articles with a unique jargon (e.g., Jets, Dolphins and Giants are sport team names). We also plan to study the influence of the user interface on user engagement, considering the position of the recommendations

within the article, the number of recommendations per article and the possible differences between desktop and mobile devices.

Acknowledgements We thank the reviewers for their comments. We also thank Jonathan Braude, Adi Grossman, Ori Koral, Dan Pelleg, Ido Yablonka and Vadim Zak for technical assistance and valuable discussions.

REFERENCES

[1] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W. Bruce Croft. 2018. Unbiased Learning to Rank with Unbiased Propensity Estimation. In *Proceedings of SIGIR*. 385–394.

[2] Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean-Francois Crespo. 2009. Sources of evidence for vertical selection. In *Proceedings of SIGIR*. 315–322.

[3] Michael Bendersky, W. Bruce Croft, and Yanlei Diao. 2011. Quality-biased ranking of web documents. In *Proceedings of WSDM*. 95–104.

[4] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning Structured Embeddings of Knowledge Bases. In *Proceedings of AAAI*. 301–306.

[5] Ilaria Bordino, Gianmarco De Francisci Morales, Ingmar Weber, and Francesco Bonchi. 2013. From Machu_Picchu to Rafting the Urubamba River: Anticipating Information Needs via the Entity-query Graph. In *Proceedings of WSDM*. 275–284.

[6] Ilaria Bordino, Yelena Mejova, and Mounia Lalmas. 2013. Penguins in Sweaters, or Serendipitous Entity Search on User-generated Content. In *Proceedings of CIKM*. 109–118.

[7] Horatiu Bota, Ke Zhou, and Joemon M Jose. 2016. Playing your cards right: The effect of entity cards on search behaviour and workload. In *Proceedings of CHIIR*. 131–140.

[8] Xuezhi Cao, Kailong Chen, Rui Long, Guoqing Zheng, and Yong Yu. 2012. News Comments Generation via Mining Microblogs. In *Proceedings of WWW Companion*. 471–472.

[9] Zhicong Cheng, Bin Gao, and Tie-Yan Liu. 2010. Actively Predicting Diverse Search Intent from User Browsing Behaviors. In *Proceedings of WWW*. 221–230.

[10] Koby Crammer, Alex Kulesza, and Mark Dredze. 2009. Adaptive Regularization of Weight Vectors. In *Proceedings of NIPS*. 414–422.

[11] Na Dai, Milad Shokouhi, and Brian D Davison. 2011. Learning to rank for freshness and relevance. In *Proceedings of SIGIR*. 95–104.

[12] Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of IJCAI*. 1606–1611.

[13] Parth Gupta and José Santos. 2017. Learning to Classify Inappropriate Query-Completions. In *Proceedings of ECIR*. 548–554.

[14] Michael Heilman and Noah A. Smith. 2010. Good Question! Statistical Ranking for Question Generation. In *Proceedings of NAACL HLT*. 609–617.

[15] Katja Hofmann, Lihong Li, and Filip Radlinski. 2016. Online Evaluation for Information Retrieval. *Foundations and Trends in Information Retrieval* 10, 1 (2016), 1–117.

[16] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of SIGKDD*. 217–226.

[17] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of SIGIR*. 154–161.

[18] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. 2007. I know what you did last summer: query logs and user privacy. In *Proceedings of CIKM*. 909–914.

[19] Weize Kong, Rui Li, Jie Luo, Aston Zhang, Yi Chang, and James Allan. 2015. Predicting Search Intent Based on Pre-Search Context. In *Proceedings of SIGIR*. 503–512.

[20] Ralf Krestel, Thomas Werkmeister, Timur Pratama Wiradarma, and Gjergji Kasneci. 2015. Tweet-Recommender: Finding Relevant Tweets for News Articles. In *Proceedings of WWW Companion*. 53–54.

[21] John D. Lafferty and Chengxiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR*. 111–119.

[22] Chia-Jung Lee and W Bruce Croft. 2012. Generating queries from user-selected text. In *Proceedings of IIiX*. 100–109.

[23] Tie-Yan Liu. 2011. *Learning to Rank for Information Retrieval*. Springer. I–XVII, 1–285 pages.

[24] Donald Metzler and W. Bruce Croft. 2005. A Markov random field model for term dependencies. In *Proceedings of SIGIR*. 472–479.

[25] Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proceedings of CIKM*. 233–242.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of NIPS*. 3111–3119.

[27] Hadas Raviv, Oren Kurland, and David Carmel. 2016. Document Retrieval Using Entity-Based Language Models. In *Proceedings of SIGIR*. 65–74.

[28] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of TREC*.

[29] Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The First Question Generation Shared Task Evaluation Challenge. In *Proceedings of INLG*. 251–257.

[30] J. Ben Schafer, Joseph A. Konstan, and John Riedl. 2001. E-Commerce Recommendation Applications. *Data Min. Knowl. Discov.* 5, 1-2 (Jan. 2001), 115–153.

[31] Erez Shmueli, Amit Kagian, Yehuda Koren, and Ronny Lempel. 2012. Care to Comment?: Recommendations for Commenting on News Stories. In *Proceedings of WWW*. 429–438.

[32] Milad Shokouhi and Qi Guo. 2015. From Queries to Cards: Re-ranking Proactive Card Recommendations Based on Reactive Search History. In *Proceedings of SIGIR*. 695–704.

[33] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. In *Proceedings of WSDM*. 610–618.

[34] Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 2012. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of CIKM*. 1980–1984.

[35] Harish Yenala, Ashish Jhanwar, Manoj K Chinnakotla, and Jay Goyal. 2017. Deep learning for detecting inappropriate content in text. *International Journal of Data Science and Analytics* (2017), 1–14.

[36] Yunzhang Zhu, Gang Wang, Junli Yang, Dakan Wang, Jun Yan, Jian Hu, and Zheng Chen. 2009. Optimizing search engine revenue in sponsored search. In *Proceedings of SIGIR*. 588–595.