Learning to Rank for Educational Search Engines

Arif Usta, Ismail Sengor Altingovde, Rifat Ozcan, and Özgür Ulusoy

***Abstract*—In this digital age, there is an abundance of online educational materials in public and proprietary platforms. To allow effective retrieval of educational resources, it is a necessity to build keyword-based search engines over these collections. In modern Web search engines, high-quality rankings are obtained by applying machine learning techniques, known as learning to rank (LTR). In this article, our focus is on constructing machine-learned ranking models to be employed in a search engine in the education domain. Our contributions are threefold. First, we identify and analyze a rich set of features (including click-based and domain-specific ones) to be employed in educational search. LTR models trained on these features outperform various baselines based on ad-hoc retrieval functions and two neural models. As our second contribution, we utilize domain knowledge to build query-dependent ranking models specialized for certain courses or education levels. Our experiments reveal that query-dependent models outperform both the general ranking model and other baselines. Finally, given well-known importance of user clicks in LTR, our third contribution is for handling singleton queries without any click information. To this end, we propose a new strategy to "propagate" click information from the other, similar, queries to the singleton queries. The proposed click propagation approach yields a better ranking performance than the general ranking model and another baseline from the literature. Overall, these findings reveal that both the general and query-dependent ranking models, trained using LTR approaches, yield high effectiveness in educational search, which may ultimately lead to a better learning experience.**

***Index Terms*—Educational search, learning to rank (LTR), query-dependent ranking, search engines.**

I. Introduction

Given the growing amount of digital educational materials (such as lecture notes, animations, and videos covering a large variety of subjects) and emergence of portals (public or proprietary) including such materials, there is a clear need for building Web-style keyword-based search engines over these collections. Such educational search engines would be essentially used by learners (e.g., students) and, hence, should be highly effective, as surfacing a large number of highly relevant results is likely to help increasing the learner's knowledge on her search topic. Earlier studies also show that optimizing search results for educational goals may ultimately improve learning gains of the users [1]. Educators (e.g., teachers) would also benefit from the educational search engines with high effectiveness, as they may reach more relevant materials for their own courses while spending less time and effort [2].

To obtain the most relevant results for a given query, modern Web search engines employ hundreds of features extracted from the queries, documents, and user behaviors. As it is impossible to combine all these features' scores using manually designed ranking functions, supervised machine learning, namely, learning to rank (LTR), techniques are employed to automatically build ranking models [3], [4]. LTR approaches allow capturing importance and interaction of a large number of features and result in ranking models that are highly effective. Therefore, nowadays, most commercial Web search engines apply a two-stage ranking strategy [5]. First, the documents in the collection are scored using simple matching metrics (such as BM25) over query–document pairs and document quality metrics based on Web graph (such as PageRank), and a candidate set, typically including a few thousands documents with the highest scores, is identified. In the second stage, this candidate set is reranked using an LTR model to obtain the final query result.

While earlier works address applying LTR for verticals in various domains (such as images, news, and e-commerce [3]), as far as we know, verticals specialized for education are usually overlooked. In most of the earlier works, search in the educational platforms (e.g., over learning object (LO) repositories) is conducted using manually designed ranking functions [2], [6]–[8]. Here, we argue that LTR approaches can be applied for building general and specialized ranking models based on the features available in the education domain, with the goal of providing higher search effectiveness and, ultimately, higher learning gains for the users.

In this article, we address the problem of constructing machine-learned ranking models to be employed in a search engine in the education domain. We base our analysis and evaluations on the data obtained from a search engine that is the part of a commercial online education platform, so-called Vitamin, for K–12 students in Turkey. Thus, our main use case is an educational search engine for K–12 students (specifically, those from fourth to eighth grade, as will be discussed later), while we mention other possible use cases in Section VI-F. Given the large number of K–12 students in Turkey, which is more than 16 million, and a very competitive

Manuscript received August 3, 2020; revised April 3, 2021; accepted April 19, 2021. Date of publication April 27, 2021; date of current version June 4, 2021. This work was supported by the Scientific and Technological Research Council of Turkey under Grant 113E065. The work of Ismail Sengor Altingovde was supported in part by the Turkish Academy of Sciences' Young Scientists Award Program. *(Corresponding author: Arif Usta.)*

Arif Usta and Özgür Ulusoy are with the Department of Computer Engineering, Bilkent University, Ankara 06800, Turkey (e-mail: arif.usta@bilkent.edu.tr; oulusoy@cs.bilkent.edu.tr).

Ismail Sengor Altingovde is with the Department of Computer Engineering, Middle East Technical University, Ankara 06800, Turkey (e-mail: altingovde@ceng.metu.edu.tr).

Rifat Ozcan is with the Microsoft Corporation, 0194 Oslo, Norway (e-mail: rifatozcan1981@gmail.com).

Digital Object Identifier 10.1109/TLT.2021.3075196

test-based process for admission to the prestigious high-schools and universities, Vitamin is a fairly popular platform with more than 1.2 million registered users and 4.3 million site visits per month. Vitamin includes a large number of educational materials in various formats (i.e., subject descriptions and questions stored as text and multimedia documents) that may be accessed via browsing over the collection or a simple keyword-based search [9]. Therefore, our dataset obtained from Vitamin is a fairly representative sample to conduct analyses and experiments for ranking in the context of an educational search engine for K–12 students.

A. Research Goal and Novel Contributions

The main research goal of this article is automatically building effective ranking models for an educational search engine. Our novel contributions toward this goal are threefold.

1) *Feature Engineering for LTR in Educational Search:* Feature engineering is a critical component for the success of LTR models [4], and hence, features employed by large search engines are considered as trade secrets and not disclosed. While it is possible to compute typical retrieval features over public document collections and analyze their importance for LTR (e.g., [10]), features based on the implicit user feedback (most crucially, user clicks to documents in the ranking) are hard to obtain, since it requires accessing query logs, and hence, only a few previous works analyze LTR performance in a setup with such features and mostly for general-purpose Web search engines (see, e.g., [11]). Therefore, an in-depth analysis of contribution of specific features to the performance of LTR algorithms based on *real user data* would be invaluable for an educational vertical.
   In this article, as our first contribution, we exploit a unique dataset obtained from the Vitamin platform to identify the features that would be most useful in the educational search context. This dataset allows us to extract click-based and domain-specific features, as well as representatives from the other feature categories employed in the literature, namely, query-specific, document-specific, query–document similarity-based, and session-based feature groups. To the best of our knowledge, this article is the first one that employs and analyzes such a wide range of features for applying LTR in an educational search engine. Our models trained on these features outperform both the original ranking generated by the Vitamin platform and those obtained by the ad-hoc retrieval functions and two well-known neural models [12], [13] that are based on the textual content of the queries and documents. We extensively evaluate the features employed in our models and provide valuable insights on the importance of different feature types in the ranking performance in the context of educational search.
2) *Query-Dependent Ranking Models for Educational Search:* As a second contribution, we leverage domain knowledge to build query-dependent ranking models [14] rather than a single general model to answer all queries. In particular, we exploit the knowledge of actual query strings and users who submitted queries to learn two types of specialized ranking models, namely, based on the "course" of query and "grade" (i.e., education level) of query issuer. Our findings reveal that building a ranker for each grade category yields higher retrieval accuracy than a single general model, as well as the baseline query-dependent models based on automatically created query clusters (i.e., as proposed in [15]).
3) *Click Propagation Algorithm for Educational Search:* Finally, as a third contribution, we focus on query-dependent models based on query frequency and build specialized models for singleton queries, that is, those appear only once in the query log. Such queries, by definition, lack previous click information and, hence, benefit the least from the LTR approaches [16]. As a solution, we propose a new strategy to "propagate" click information from the other, similar, queries with the click information to the singletons. Our approach is novel in that we do not attempt to make a prediction for the global click count of a document (as in other works, such as [17]); instead, we create a synthetic value for the document's click count for a given query. While doing so, we also do not rely on co-click information like most of the earlier works [17], [18], which by definition does not exist, but combine other clues (including domain-specific ones) such as overlap of results, similarity of query strings, and similarity of grade levels for the students who submitted the queries. Our experiments show that the performance of ranking model using enhanced click information outperforms the general model as well as another baseline strategy proposed by Gao *et al.* [17] to remedy the same problem for tail queries in Web search engines.

Overall, our findings reveal that the general and query-dependent ranking models, trained using LTR approaches, yield high effectiveness in educational search, which may ultimately lead to a better learning experience.

The rest of this article is organized as follows. In the next section, we review earlier works on educational search and LTR algorithms. In Section III, we first provide an in-depth discussion of features employed in the ranking models. Next, we discuss query-dependent ranking models based on "course," "grade," and "frequency" features of queries. Then, in Section IV, we introduce the click propagation algorithm for the singleton queries with no previous click information. Section V presents the dataset used in our experiments and the details of the relevance annotation process. In Section VI, we compare our findings with state-of-the-art baselines and show that the proposed models considerably improve the search engine performance. Section VII provides the conclusion and points to future research directions.

II. Related Work

A. Educational Search

As children at primary or secondary schools are likely to be among the users of an educational search engine (as in the

K–12 use case employed in this article), we begin with reviewing information retrieval research addressing the children. There has been a recent interest in investigating search characteristics of young users [9], [19]. Some of these works are based on case studies with a group of young users [20]–[22], while the others utilize search engine logs to analyze search behavior of such users [9], [19], [23], [24]. One of the critical findings is that young users have difficulty in formulating queries for both general search tasks [19] and school-related (educational) tasks. Gyllstrom and Moens [25] present a link-based ranking algorithm, the so-called AgeRank, that favors Web pages that are appropriate for children. Eickhoff *et al.* [26] classify Web pages that are suitable for children. Torres *et al.* [27] propose a query recommendation method customized for children. Yang *et al.* [28] utilize deep convolutional neural networks to classify potential at-risk K–12 students. None of these aforementioned works build ranking models for educational search, as we do in this article.

Search as learning is an emerging paradigm that addresses the design of search systems to enhance learning experience [29]. To this end, some earlier works aim to improve the rankings of Web search engines. In [1], Syed and Collins-Thompson propose a ranking strategy specialized for vocabulary learning task, while Yilmaz *et al.* [30] rerank results based on their predicted educational subject category. In contrary, other works address search engines built on top of various semiformal educational platforms. Such platforms may include LOs, open educational resources (OER), or proprietary educational materials (such as those in the Vitamin platform employed in this article). In [2], Yen *et al.* employ a manually designed ranking function that takes into account popularity features and pairwise similarities of the LOs. The work in [7] describes explicit and implicit features to represent the quality of LOs and again uses manually designed functions to combine their scores for ranking. Pimentel *et al.* [8] propose a term clustering approach for query expansion and a manual ranking function for searching in an OER repository. In a closer work to ours, Ochoa and Duval [31] propose various relevance features tailored for the ranking of LOs. Their dataset involves top ten LOs retrieved for a set of ten (predefined) queries (i.e., each corresponding to a lesson in computer science). This article differs from [31] in several ways. First, their work is based on a learning platform for university students, while we build ranking models employing a query log sample (with 900 distinct queries) from an educational search engine used by K–12 students. This allows us to employ a rich set of features that also capture users' interaction with the search results. Second, in addition to general models, we also train query-dependent models. Third, we propose a solution to handle queries without any previous click information. Fourth, our experiments employ rank-aware effectiveness metrics, which is the state of the art for evaluating search systems [4]. In a recent survey [6], the methods for searching LOs in repositories are categorized as metadata-based, full-text, and hybrid; however, none of the surveyed methods (except [31], as discussed before) employ LTR approaches in the context of educational search.

While we focus on search engines, the related topic of recommendation systems is also addressed in education domain. Verbert *et al.* [32] survey context-aware recommender systems for learning. Peralta *et al.* [33] analyze impact of available metadata for educational sources on the performance of recommender systems. Tang and Pardos [34] employ recurrent neural networks for a recommendation system built on edX, a MOOC platform. In line with the recent works exploiting machine learning for recommendation systems, here, we apply LTR approaches in the context of educational search engines.

B. LTR in Web and Vertical Search Engines

In one of the earliest works on the topic of LTR, Qin *et al.* [35] introduce several features to be used in learning algorithms and categorize them into four groups, which are low-level content features (e.g., tf-idf), high-level content features (e.g., BM25), hyperlink features (e.g., PageRank), and hybrid features. Such features are widely used for LTR in Web search engines [3]. In [10], Macdonald *et al.* investigate the impact of query-specific features in LTR. In their seminal work, Agichtein *et al.* [11] introduce features based on the user behavior and demonstrate their importance for LTR. More recently, a deep learning model has been proposed to exploit implicit feedback in LTR, again for the Web search scenario [36]. LTR approaches have been employed for various vertical search engines [37]. One such work aims to construct a search engine for news articles. The challenge in such a domain is to use recency information of news, since the newer an article is, the more likely users tend to click that particular article. Therefore, using click through data, Wang *et al.* [38] propose a framework for modeling both topical relevance and freshness of news articles.

Another domain where LTR is applicable is keyword-based image search engines. For instance, in [39], Jain and Varma argue that using only textual features that are extracted from the Web pages including the images may not be adequate to train ranking models. Therefore, they first train a model to predict click counts of the images using both textual and visual features and then exploit the click data to rerank the initial result list of images for a given query. As far as we know, there is no previous work employing LTR approaches for an educational search engine, as proposed in this article.

C. Query-Dependent Ranking Models

Queries issued to a search engine may significantly vary according to various aspects, such as the popularity, length, and underlying information need [40], and hence, it is unlikely for a single ranking model to generate good rankings for all possible types of queries. For instance, a model that performs well for the popular queries by exploiting the prior click information as a key feature may fail for the tail queries, for which sparse or no click information is available. In [15], Geng *et al.* identify the k-nearest neighbor of a given query to determine the most relevant training instances to build a ranking model and report that query-dependent ranking models outperform the single general model using the state-of-the-art techniques.

In [40], Bian *et al.* train a separate model for each "query topic" (inferred from the training data) and then ensemble their results to obtain the final ranking for a test query. In a similar fashion, Giannopoulos *et al.* [41] aim to build a different model for each query intent. To this end, they first learn a ranking model for every training query and cluster these models and then train a model for all the queries that are clustered together in the previous step. In all of these earlier works, the queries are typically modeled together with the top-ranked retrieved documents (i.e., to compute pairwise similarity of two queries), while in this article, we essentially focus on purely query-specific features, namely, the course and grade of a query, available in an educational search scenario, as well as the query frequency.

D. Specialized Ranking Models for Singleton or Tail Queries

The click-through data have been exploited for various purposes in Web search and, especially, for LTR. However, benefits are limited for the queries with sparse or no click information in the query log. This problem generally occurs for queries having low frequencies, called tail queries. Recently, researchers have started to focus on this issue by trying to generalize learned models to perform well enough for tail queries, as well. In [18], Aktolga and Allan try to boost rarely clicked queries in a system where limited click-through data are available. They attempt to generate click-through features using the set of similar queries to the given query, which has no to little click data available. Their work categorize similar queries into three groups: similar queries that share at least one co-click, synonym queries that are lexically related to each other, and subset queries where one is included in the other as a subset. They claim that their models using three sets of similar queries perform better than the baseline model.

Again for the purpose of handling queries with sparse or no click data, Gao *et al.* [17] introduce click-through stream as the set of queries having co-click for a particular document in the query log given a certain document. Their calculation of click-through features differs from the ones in the literature in the sense that they also consider whether a click is the last click in the search session to give more importance for the documents that are clicked last. More recently, Jiang *et al.* [42] propose to represent both query and documents as a vector in same semantic space. They also provide a propagation algorithm to generate vector representations of unseen queries and documents using association with the vectors already generated. Different from these previous works, we do not rely on the co-click information, which by definition does not exist for the queries with no clicks, but combine other clues such as the overlap of results, similarity of the query strings, and similarity of the grade levels for the submitting students.

III. LTR for Educational Search: Dataset, Features, and Models

In this section, we first describe our dataset and extracted features that are utilized for training various ranking models in the context of an educational search engine. Next, we discuss how we construct the query-dependent models based on certain features (i.e., course, grade, and frequency) of the queries.

A. Dataset

In a typical LTR setup, the training data involve user queries, retrieved documents for each query, and explicit or implicit relevance labels for the documents of a query. The actual queries and their retrieved documents are usually extracted from a query log along with click data, and then, each query–document pair is represented by a large number of features based on the query, session, document, query–document similarity, clicks, etc. [35], [43]. Finally, for each pair, relevance judgment is obtained either explicitly via an editorial annotation process or implicitly such as by exploiting signals like click and dwell time, or both. As discussed before, earlier works either employ public datasets without real user interaction (e.g., lacking the click information) or anonymized datasets that do not allow us to evaluate and compare the importance of certain features (especially those based on the user interaction) in the ranking models neither for a general Web search engine nor for a vertical. Therefore, this article is first to define and assess various features for building ranking models in a vertical for educational search.

We use the data provided by a commercial Web-based educational platform, called Vitamin, used by K–12 students in Turkey. In particular, the dataset includes the following: 1) a query log sample that includes 66 908 queries issued to Vitamin (details are discussed later in Section V); and 2) metadata of the documents that appear in the log. Note that, while we generally refer to query results as documents, they are actually educational materials in various types (lecture, summary, exercise, animation, etc.), as provided in the associated metadata [9]. For the users in the log, all identification information is anonymized (except a hashed user-id field), and only some basic information, such as the grade of the user, is made available. In what follows, we describe the features extracted from this dataset to construct the training and test instances for LTR algorithms. We postpone discussing other details (such as the number of instances and splitting them into training and test sets) to Section V.

B. Features for Learning Models

Following the literature, we categorize our features into five main categories, namely, query-specific, document-specific, query–document similarity-based, session-based, and query–document click-based features. As discussed next, in each category, we extract widely used features employed in earlier works (see, e.g., [35]), as well as those that are specific to our application domain (such as course of a document, grade of a user, and type of a document). Overall, a data instance in our LTR setup, that is, a query–document pair, is represented by a 50-D feature vector (and a relevance label, which is discussed in Section V). The features used in our LTR models are presented in Table I along with their corresponding feature group they belong to.

TABLE I
FEATURES FOR EDUCATIONAL LTR

Feature Group	Feature Name	Type	Value
Query–Document Text Similarity	tf-idf title	real	[0, 1]
	tf-idf description	real	[0, 1]
	BM25 title	real	[0, 1]
	BM25 description	real	[0, 1]
Query-Specific	Query Frequency	real	[0, 1]
	User Count	real	[0, 1]
	Result Count	real	[0, 1]
	Top Document Count	real	[0, 1]
	Length (no. of tokens)	real	[0, 1]
	Length (no. of characters)	real	[0, 1]
	Has Grade Name	boolean	{0, 1}
	Has Course Name	boolean	{0, 1}
Document-Specific	Document Frequency	real	[0, 1]
	User Count	real	[0, 1]
	Document Course	boolean	{0, 1}
	Document Grade	boolean	{0, 1}
	Document Type	boolean	{0, 1}
Session-Based	Total click count	real	[0, 1]
	Unique click count	real	[0, 1]
	Total dwell time	real	[0, 1]
	Result count	real	[0, 1]
	isClicked	boolean	{0, 1}
	isSkipped	boolean	{0, 1}
	isMissed	boolean	{0, 1}
	Dwell time	real	[0, 1]
	Click count	real	[0, 1]
Query–Document Click-Based	Impression count	real	[0, 1]
	Click count	real	[0, 1]

1) Query–Document Text Similarity Features: The documents in our dataset typically have a title and description, while the latter can be a long discussion for a textual material but a short summary for a visual material, like an animation. Anyway, we compute the textual similarity of a query to each of these parts using two common metrics, namely, tf-idf and BM25, as the previous works in the literature. In order to calculate query–document text similarity features effectively (tf-idf and BM25), we apply the following preprocessing steps. First, we remove common stop words in Turkish as well as the punctuation and nonunicode characters. Then, we find stem of each word by extracting the first five characters as the stem, which is proven to be effective for agglutinative languages such as Turkish [44]. For each query session, calculated tf-idf and BM25 scores for each document are normalized into 0–1 range using the linear normalization method. The final list of features for this group can be seen in Table I.

2) Query-Specific Features: As two well-known features, for each query, we extract the query frequency (i.e., number of times this query is seen in the query log) and user count (number of different users who issued this query) [4]. Result count is self-explanatory, that is, the number of documents retrieved for the query, while top document count is the union of documents that are retrieved in the first page result list. The query length feature is expressed both in number of tokens and characters, as usual in LTR setups [45].

The last two features in this category are novel features specific to the educational search domain. We observed that students tend to write queries that include either a grade (like "polynomials *fifth grade*" or the course of the subject that they are looking for (e.g., "light *physics*"). Therefore, in the last two rows of Query-Specific group in Table I, we present two Boolean-valued features that represent whether a query string includes a course name or a grade, respectively.

3) Document-Specific Features: As in the case of queries, we form two features to represent the popularity of the documents, as follows: *the document frequency* is the total number of clicks for a given document in the dataset, and *user count* is the number of unique users who clicked this document. Both feature values are normalized across all training data.

Additionally, we have three other features, namely, course, grade, and type of documents, which are again specific to our domain. Specifically, each feature and their domains can be summarized as follows.

1) *Document course:* A document in our dataset may be associated with one of the five courses available in the system: Math, Turkish, Science, Social Sciences, and Revolution History.
2) *Document grade:* Each document covers a subject taught at a particular grade that ranges from fourth to eighth grade.
3) *Document type:* There are 15 different types of documents in our data. These types are based on the format and/or purpose of the material, such as animation, text, summary, quiz, video, exercise, etc.

As usual, while creating the actual feature vectors, these three categorical features are all converted to binary features for each possible value they can take, yielding a total of 25 binary features.

4) Session-Based Features: As discussed before, a key source of information for training ranking models is previous patterns of user behavior, which is typically captured in a query log. In particular, using a query log, one can either extract "long history" for a user, that is, reflecting all her previous searches, clicked or unclicked results, etc., or "short history" based on the current session where she submits the query that is being processed [46].

In this section, we focus on capturing the short history of the user. First, we describe a set of features that represent the aggregated user activity in previous query issues involved in the current search session, as in [47]. A search session involves the queries issued by the same user within 30 min. With these features, we try to capture user search behavior in the current session. The first four rows of Session-Based group in Table I present these features, namely, the number of results presented and clicks (either total or unique) observed in the current session, as well as the total dwell time over the clicked results. All features in this group are normalized according to the values in all unique sessions.

Second, we keep track of the users' detailed activities in previous query issues in the current session. Given that each data instance in an LTR setup is a query–document pair, the feature *isClicked* captures whether a document has been displayed to and clicked by the user previously in the current session. Similarly, the feature *isSkipped* tracks whether the user has chosen not to click on a particular document but clicked another one at a lower rank. The feature *isMissed* is to record for the documents that have been ranked at a lower rank than the rank of the last clicked document. As introduced in [48], these features represent what the user has done when she has encountered a document, which is a candidate to be ranked for her current query, in that session before submitting her current query. As the last two features, we also obtain dwell time and click count of the documents shown in the current result list and clicked by the user for the previous queries in the current session.

5) Query–Document Click-Based Features: The information of whether a document has been previously displayed and clicked (or, not) for a particular query is invaluable in predicting the relevance of the document for that query. We employ two such features, namely, *impression count* and *click count*, that represent the number of times a document is retrieved in the result list and clicked for the given query, respectively.

C. Query-Dependent Ranking Models

In this article, as will be presented in Section VI, we first utilize the aforementioned features for building a general ranking model, which is applied to rank results for all test queries and assess impact of certain feature categories and/or individual features on retrieval effectiveness. Additionally, we build query-dependent models taking into account characteristics of educational search domain.

1) Course-Specific Models: Unlike general Web search queries reflecting very diverse information needs, the queries submitted to our education vertical express more specific needs that are likely to be associated with a target course (or, rarely, more than one course). For instance, the query "properties of light" is relevant to Science course, whereas "greatest common divisor" is for Math. The user behavior/preferences may be different for queries targeting different courses (e.g., the users may prefer animation type results for Science but interactive exercises for Math), and hence, each course may require a different ranking function. Therefore, we manually labeled our training queries for the target course, which can be either one of the five courses available in the system, or the so-called General, for those queries that may have an intent exceeding the scope of a particular course. We then separately trained course-specific models using their respective training instances.

2) Grade-Specific Models: Inspired by the finding reported in [49] that grade level of children plays role in query type and search task outcomes, we group queries using the grade of the student who submit the query. Students in different levels of education may have different requirements, for example, fourth graders may prefer easy-to-grasp material including a shorter text and larger number of visual components in comparison to eighth graders. Their clicking behavior may also differ, that is, older students may be more inclined to read result snippets and, therefore, click selectively, while younger ones may not pay attention to such clues. Such possible differences can be handled by weighting corresponding features (be it the type or click-through rate (CTR) for a document) differently in the ranking models by building specialized rankers for each grade. In this article, using the features described before, we train five different models for the students that are in fourth to eighth grade, respectively.

3) Frequency-Specific Models: While domain-specific clues guide our decision for building course-specific and grade-specific models, a more general dimension to group queries is the query frequency. Earlier works (see, e.g., [4] and [16]) report that the ranking functions for the popular queries (i.e., those that are from the "head" of the heavy-tailed distribution of query frequencies) usually perform well, due to the abundance of the click information based on the previous displays (i.e., impressions). In contrast, learning successful models for the torso and tail queries is a challenging task due to the lack of invaluable user interaction data. Inspired by these observations, we build and evaluate specialized rankers for the singleton queries (i.e., those appear only once in the query log) and nonsingleton ones. While various approaches have been proposed in the literature to improve the retrieval effectiveness of tail queries (e.g., by calculating term-based similarity function for rare queries [50], or by collaborative ranking [51]), we are not aware of an approach that learns models for singletons and nonsingletons, separately. As a further step toward improving the model for singletons, we explore a strategy to approximate the values for the click-based features as discussed in the following section.

IV. Click Propagation for Singleton Queries

As mentioned before, the major problem of ranking models for tail queries is the lack of previous impressions and click data. In this article, apart from learning a model only for the singletons, we also attempt to generate synthetic values for the impression and click count features described previously, to improve the performance of the general and query-dependent models. Note that most of the earlier works in the literature [17], [52]–[54] also consider the queries with a few clicks in the scope of tail queries; hence, they are still able to use (albeit sparse) co-clicks, while we only focus on the singletons, that is, previously unseen queries, with no clicks *at all*.

Our proposed strategy exploits the observation that since the domain of a vertical is much more restricted than a general search engine, it is more likely to successfully identify other queries (with click information) *similar* to a singleton query based on the query content. For instance, consider the popular query "photosynthesis" versus a possible singleton like "the role of pigments in photosynthesis process." At the time of writing, top ten results for these queries from Google yielded only one overlapping result. While variety and depth of answers for these two queries may highly vary when submitted to a general-purpose search engine, in a vertical restricted to a certain domain, one can expect a larger overlap among the result lists, which can be exploited to improve the ranking for the latter query. Once such similar queries are found, we propagate their impression and click counts to the singleton query. In what follows, we present the detailed methodology for identifying similar queries and propagating feature values.

We determine the similar queries for a given singleton query in two steps. It is long known that the similarity of two queries might be best determined by the overlap of the result lists and, if available, overlap of the clicked documents (i.e., co-clicks [18]). In our case, the latter information does not exist, so we only rely on the former to create a set of candidate queries as our first step. Specifically, a query is in the candidate set if: 1) its result list includes at least one common document with that of the singleton query; and 2) at least one of these common documents is clicked for the former query (so that we will have some values to propagate at the end).

In the second step, we compute a similarity score of each candidate query q_i to the singleton query q_s exploiting the following three types of evidence.

1) *Grade Similarity:* The function $G(q_i, q_s)$ returns 1 if the students who submit the queries q_i and q_s are at the same grade, and returns 0, otherwise.
2) *Query Text Similarity:* We compute the cosine similarity of the query strings as $C(q_i, q_s)$.
3) *Result List Similarity:* The function $J(q_i, q_s)$ computes the Jaccard coefficient between the result lists of q_i and q_s.

Based on these components, the similarity score $S(q_i, q_s)$ is calculated as follows:

$$S(q_i, q_s) = \alpha \times G(q_i, q_s) + \beta \times C(q_i, q_s) + \gamma \times J(q_i, q_s)$$

where α, β, and γ are weight coefficients determined experimentally. The highest scoring N candidate queries form the final set of similar queries to q_s. In the experiments, we restrict the set size N to 10 for runtime efficiency and to avoid introducing noise by less similar queries.

In the propagation stage of our approach, for each document d in the result list of the singleton query q_s, we obtain a synthetic value for the *ClickCount* feature based on the values of the similar queries q_i, as follows (of course, if the result list of q_i lacks d, its contribution is 0):

$$q_{i_{c,i}} = \frac{\sum_{s=1}^{N} q_{s_{c,i}} \times S(q_i, q_s)}{N}.$$

The *ImpressionCount* feature is computed in a similar fashion.

V. Experimental Setup

A. Training and Test Sets

As discussed before, we use the data provided by a commercial Web-based educational platform, called Vitamin, used by a large number of K–12 students in Turkey. In particular, the query log sample includes 66 908 queries (18 638 of which are unique) issued to the search engine of Vitamin in December 2013. These queries are submitted by $18K$ unique users, and on the average, a user asks 3.61 queries in 1.92 sessions. In an earlier study [9], we have analyzed the query log and highlighted the similarities and differences of the searches made by the users of this vertical to those made in a general-purpose Web search engine.

For the purposes of this article, we sampled (uniformly at random) 900 unique queries from the log, which are in total submitted 3169 times. We chronologically sorted these submissions (i.e., instances), and in all our experiments, we use the first 80% according to timestamps as the training set and the rest as the test set, similar to a previous study [55].

B. Relevance Annotation

The query log includes (at most) top 25 documents (i.e., internal doc-ids) retrieved for a query, together with additional information for those that are clicked. For each submission of a given unique query, we obtained the list of retrieved documents from the log. The union set of these result lists constitutes the answers to be annotated for a query.

Then, we asked judges to annotate these documents given query text, document title, and document description. Specifically, we split the set of 900 queries to nine equally sized mutually exclusive groups and assigned each piece to a different judge. The list of judges consists of graduate students and professors, all of whose native language is Turkish.

For the annotation, we carried out two different labeling. The first one is categorical annotation of query text in terms of course, to which that query may belong. We had five different courses initially, which is derived from the query log, and we also added another course category named "General Course," which we can use for queries that cannot be categorized among possible course candidates, such as the query "games" that seem to seek for game-based resources for any course in the system. In total, we have six different courses that could be matched for a given query, which are Math, Turkish, Science, Social Sciences, Revolution History, and General Course.

The second part is the usual annotation scheme for LTR datasets, that is, to give relevance score for each document associated with a particular query. Each query–document pair is annotated with one of the following relevance scores:

1) 0—irrelevant (i.e., the document is irrelevant to the query);
2) 1—mostly relevant (i.e., course and subject of the document matches with the query, yet document does not satisfy the user needs according to the query text);
3) 2—exact match (i.e., precisely what the query asks for).

By annotating 900 unique queries, we obtained 3169 annotated query instances to be used for LTR algorithms. There are 16.4 documents annotated per query instance, yielding 52 260 query–document pairs in our dataset.

VI. Experiments

A. Baseline Models

As our first goal is investigating retrieval effectiveness of LTR in the context of an educational vertical, we employ two traditional ad-hoc matching functions, namely, *tf-idf* and

TABLE II
RETRIEVAL EFFECTIVENESS OF BASELINE METHODS

	NDCG@5	**NDCG@10**
Vitamin SE	0.6370	0.6711
tf-idf Title	0.6561	0.6710
tf-idf Description	0.6259	0.6530
BM25 Title	0.6578	0.6735
BM25 Description	0.6455	0.6655
tf-idf Linear (0.7)	0.6608	0.6722
BM25 Linear (0.7)	**0.6692**	**0.6804**

BM25, as traditional baselines. The final ranking in commercial general-purpose Web search engines is not directly based on such functions; however, they are employed both in the first stage retrieval (i.e., to generate candidate documents) and for generating various features for the second stage retrieval such as LTR algorithms. Hence, these ad-hoc functions are strong indicators of relevance. We apply each matching function to compute the relevance of a query to either document title or description, and we obtain four different baseline rankings.

Additionally, we also employ a linear-weighted combination of the scores computed for the document title and description fields using each of the matching functions. For the weight parameter in linear combination, we used the parameter tuning method. The best results are obtained when the weight for the query–title matching score is set as 0.7.

We employ rank-aware effectiveness metrics, namely, NDCG and/or ERR, which are state of the art for evaluating search systems (see, for example, [4] for their definition).

Table II shows NDCG scores at cutoff values of 5 and 10 for each baseline method for the test set. The first row presents search performance of original ranking system of Vitamin platform's search engine (SE). Our findings show that using document titles yields higher effectiveness than using full descriptions for both tf-idf and BM25 functions. Furthermore, the rankings created by both of the matching functions using titles outperform the original ranking provided by Vitamin SE. The linear combination of scores (i.e., the last two rows in Table II) for the title and the description fields also proves to be useful, especially for BM25, which yields the highest performance over the test set. In the following experiments, we report BM25 Linear as the internal baseline (i.e., Baseline BM25), as well as the scores of Vitamin SE as the external baseline, that is, the platform's native search system.

B. Performance of General Educational Ranking Model

For training our ranking models, we employ a well-known LTR algorithm, namely, LambdaMART [56]. Table III shows the gains we achieved using the LTR model over original ranking of Vitamin and baseline BM25 (repeated from Table II for easy comparison), in terms of the NDCG and ERR metrics. We see that the general LTR model (i.e., the last row in Table III) trained with our derived feature set outperforms Vitamin's original ranking significantly, that is, by a margin of more than 14% considering the NDCG@5 scores. Additionally, the general LTR model is also superior to the BM25 baseline and outperforms the latter by almost 11%, again in terms of NDCG@5. Our results here confirm the success of the LTR approach in the context of a vertical for educational search.

TABLE III
RETRIEVAL EFFECTIVENESS OF GENERAL RANKING MODEL AND MODELS PER FEATURE GROUP

		NDCG		**ERR**	
		@5	@10	@5	@10
Baselines	Vitamin SE	0.6370	0.6711	0.5521	0.5583
	Baseline BM25	0.6692	0.6804	0.5762	0.5614
Feature-Based LTR Models	Query–Document Text	0.6868	0.7095	0.5793	0.6030
	Query Specific	0.6370	0.6711	0.5522	0.5584
	Document Specific	0.6747	0.7101	0.5648	0.5646
	Session-Based	0.6349	0.6686	0.5481	0.5549
	Query–Document Click	0.7382	0.7583	0.6055	0.6123
General LTR	**All Features**	**0.7742**	**0.7888**	**0.6205**	**0.6266**

In addition to the aforementioned baselines, we also experiment with two state-of-the-art Neural IR approaches using *Matchzoo* [57] library, which are *DSSM* [12] and *DRRM* [13]. We chose these algorithms because they fall into two different broad categories of Neural IR, namely, representation and interaction-based approaches, respectively. We briefly review these methods as follows.

1) *DSSM:* As a representation-based model, DSSM tries to represent the query and documents retrieved in the result list in the same semantic space. The algorithm first feeds one-hot encoded word vector representation into the deep network. Then, size of the representation vector is reduced by word-hashing method, which is n-gram letters of words with special start and end characters. Next, through multiple deep layers, final representations of both query and documents are obtained in vectors of size 128. For each query–document pair, *Cosine Similarity* of their associated representations is calculated and fed into the last layer of the network, softmax, where probabilities are provided as the output.
2) *DRRM:* This is an interaction-focused neural model that aims to find patterns of matching on the basic representations of query and documents. The authors argue that semantic matching, which is the objective followed in the representation-based models, is not enough to capture relevance matching. They claim that for relevance matching, there are other important factors to be considered by a Neural IR model, which are exact matching signals, term importance, and diverse matching requirements.

In the experiments, we used the training and test sets described before. Further parameter optimization is done by using a validation set. Although the performance of both models is promising, general LTR model still surpasses both models. In particular, the DSSM (DRRM) model yields NDCG@5 and NDCG@10 scores of 0.7493 (0.6912) and 0.7717 (0.7132), respectively. The inferior performance of these models can be due to the following reasons. First, both neural models are trained using the textual data (of queries and

TABLE IV
TOP TEN MOST FREQUENT FEATURES IN LAMBDAMART TREES

Feature Name	Feature Group	Frequency
click count	Query–Document Click-Based	1411
dwell time	Session-Based	1065
BM25 description	Query–Document Textual Similarity	965
impression count	Query–Document Click-Based	668
t-fidf title	Query–Document Textual Similarity	641
unique click count	Session-Based	252
document grade #24	Document Specific	248
document type #27	Document Specific	175
document type #25	Document Specific	170
document course #19	Document Specific	148

TABLE V
TRAINING AND TEST INSTANCE COUNTS FOR EACH CATEGORY

Feature	Category	Instance Count	
		Training	Test
Grade	4	875	201
	5	300	90
	6	374	82
	7	874	205
	8	113	55
Course	Math	432	92
	Turkish	424	107
	Science	344	121
	Social Sciences	137	38
	Revolution History	116	33
	General Course	1038	242
Frequency	Non-Singleton	2024	510
	Singleton	512	123
None		2536	633

documents), as proposed in the papers introducing these methods, while our models exploit various types of features. Second, our annotated dataset may not be as large as that required for training such deep models. The latter claim is in line with a recent work [58] arguing that neural ranking models may not improve retrieval effectiveness, especially, in limited data scenarios. Nevertheless, our experiments presented here justify the choice of the traditional LambdaMART algorithm to train our LTR models in this setup. In the next section, we present insights on the feature groups that enable the superior performance of the general LTR model.

C. Feature Group Analysis

Given that our 50-D feature vector includes features from five distinct groups, it is important to analyze which of these groups provides the highest contribution to the performance of the LTR model. To this end, we conduct a feature group ablation study.

In Table III (second row), we present the performance results of each feature group using LTR models that are learned by only features belonging to that group. Our results show that the best performing feature group is the query–document click features, including the click and impression count of documents for a given query. While we use the latter two features separately, they are usually combined to form a single feature (i.e., CTR) that is shown to be crucial for LTR in Web search [17], justifying our finding.

From Table III, we see that textual similarity features based on tf-idf and BM25 perform better when they are used together with other features within an LTR model than being used alone for ranking (cf. Table II). Query-specific and session-based features alone can outperform neither BM25 baseline nor Vitamin's original ranking. Yet, the latter features prove to be beneficial when they are used in combination with the other feature groups, as reflected in the performance of the general LTR model that employs all features and outperforms all models using a single feature group. As a final observation, we see that document-specific features, including a document's course and grade information, are more helpful in ranking than query-specific and session-based features, and indeed, their performance is even better than Vitamin and BM25 baseline. In addition to the feature ablation study provided in Table III, we provide further insights regarding the importance of individual features in the model learned by the LTR algorithm, namely, LambdaMART. Table IV shows top ten most frequent features used in LambdaMART trees, as an indicator of feature importance. Results of both studies for feature analysis are consistent with each other, that is, the click count feature is found as the most important one among all features. In addition to well-known textual similarity-based features, it can also be seen in Table IV that our devised features (i.e., the features depicting type of the document) contribute to the performance of the LTR model in the educational search setup.

These findings further justify our use of query-dependent ranking models (discussed next), especially based on the course and grade features, which seem to be valuable signals to obtain higher effectiveness.

D. Query-Dependent Ranking Models

We aim to figure out whether we can improve retrieval performance by having specialized ranking models for different query groups. As discussed before, we categorized our queries based on the values of one of the three features, namely, the course of the query, the grade of the user issuing the query, and the query frequency. In our setup, we employ the actual values for each one of these features, obtained by either explicit labeling (i.e., for the "course" of a query) or extracted from the available metadata (i.e., for the "grade" information of the person who issued the query and for the frequency information of the query). Note that, even when there is no such *a priori* information, it is possible to predict the values of these features with certain confidence; for example, Yilmaz *et al.* [30] discuss the prediction of course category for a given query. For each of these three features, we trained a specific ranking model for each of its categories (e.g., for the course feature, we built a ranker for the queries that fall into each category, such as Math, Science, etc.). The training and test instance counts vary for each category, as shown in Table V.

We again used NDCG and ERR metrics at cutoff values of 5 and 10. Results show that having different models for each query category improves the average performance besides

TABLE VI
RETRIEVAL EFFECTIVENESS OF GENERAL AND COURSE-SPECIFIC RANKING MODELS (VITAMIN SE IS THE BASELINE)

Course Category	NDCG						ERR					
	Vitamin SE		General		Course-Specific		Vitamin SE		General		Course-Specific	
	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10
Math	0.7429	0.7534	0.7459	0.7547	**0.7854**	**0.7839**	0.6407	0.6452	0.6386	0.6438	**0.6570**	**0.6700**
Turkish	0.6659	0.7114	**0.8034**	**0.8122**	0.7564	0.7886	0.5604	0.5697	**0.6486**	**0.6527**	0.6123	0.6199
Science	0.5632	0.6086	0.6889	0.7155	**0.6890**	**0.7193**	0.4068	0.4162	**0.4519**	**0.4599**	0.4370	0.4324
Social Sciences	**0.6798**	**0.6954**	0.6075	0.6492	0.6568	0.6953	0.5438	0.5496	0.4906	0.5024	**0.5109**	**0.5320**
Revolution History	0.7560	0.7711	**0.8404**	**0.8448**	0.8377	0.8232	0.6011	0.6065	**0.6537**	**0.6592**	0.6069	0.6409
General Course	0.5984	0.6361	0.8283	0.8261	**0.8389**	**0.8447**	0.5820	0.5861	0.7051	0.7071	**0.7082**	**0.7106**
AVG	0.6370	0.6711	0.7742	0.7888	**0.7775**	**0.7923**	0.5521	0.5583	**0.6205**	**0.6266**	0.6156	0.6218

TABLE VII
RETRIEVAL EFFECTIVENESS OF GENERAL AND GRADE-SPECIFIC RANKING MODELS (VITAMIN SE IS THE BASELINE)

Grade Category	NDCG						ERR					
	Vitamin SE		General		Grade-Specific		Vitamin SE		General		Grade-Specific	
	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10
4th Grade	0.6473	0.6851	**0.7996**	**0.8054**	0.7857	0.8030	0.6107	0.6162	**0.6650**	**0.6696**	0.6514	0.6531
5th Grade	0.6452	0.6640	0.6194	0.6540	**0.6504**	**0.6759**	**0.5596**	**0.5661**	0.5294	0.5367	0.5486	0.5375
6th Grade	0.4722	0.5257	0.7419	0.7464	**0.7672**	**0.7619**	0.2949	0.3041	0.4274	0.4327	**0.4475**	**0.4480**
7th Grade	0.6732	0.7005	0.8463	0.8476	**0.8512**	**0.8599**	0.5948	0.6004	0.7122	0.7154	**0.7129**	**0.7192**
8th Grade	0.6968	0.7410	0.7453	0.7594	**0.7633**	**0.7629**	0.5506	0.5562	0.6010	0.6060	**0.6095**	**0.6181**
AVG	0.6370	0.6711	0.7742	0.7888	**0.7833**	**0.7945**	0.5521	0.5583	0.6205	0.6266	**0.6266**	**0.6284**

improving the performance for most of the categories separately. Details are given in the following subsections.

1) Performance of Course-Specific Ranking Models: As we can see from Table VI, in terms of NDCG scores, average performance of course-specific ranking models is slightly better than the general model's performance, in which all query instances are used, that is, without any categorization for training and testing. Apart from the overall improvement, there is also improvement for queries for particular courses, namely, for Math, Science, and General Course. Although the best performance for Social Sciences course seems to be obtained with Vitamin's original ranking, we also improved the NDCG@5 score for this course from 0.6075 to 0.6568 with respect to the general model.

Another observation is that the general model outperforms course-specific models for the course categories of Turkish and Revolution History. This result is due to the fact that these courses are text-oriented courses; therefore, documents related to those courses have longer texts, which automatically improves the textual features we have, which are tf-idf and BM25. Therefore, since we have more instances with the general model, it behaves better than the course-specific models. However, we believe that if we had enough number of instances for each course type, then we might have expected this behavior to change.

Similar to the trends obtained with the NDCG metric, results for the ERR metric for Math and Science courses also indicate that course-specific models outperform the general model. Yet, in terms of ERR, average performance of course-specific models is slightly worse than that of the general model.

TABLE VIII
RETRIEVAL EFFECTIVENESS (AVERAGE) OF QUERY-DEPENDENT RANKING MODELS (VERSUS AUTOMATIC CLUSTERING BASELINES AND GENERAL LTR)

		NDCG		ERR	
		@5	@10	@5	@10
Automatic Clustering Approaches	Query-Specific	0.7200	0.7387	0.5825	0.5887
	Result-Based	0.7592	0.7726	0.6092	0.6176
Proposed Categories (Pre-determined)	Course-Specific	0.7775	0.7923	0.6156	0.6218
	Grade-Specific	**0.7833**	**0.7945**	**0.6266**	0.6284
	Frequency-Specific	0.7772	0.7895	0.6239	**0.6307**
General LTR		0.7742	0.7888	0.6205	0.6266

2) Performance of Grade-Specific Ranking Models: We have five different values (categories) for grade feature indicating grades of users who submit the query to the search engine. Therefore, we trained five different models for this experiment, and the results show that grade-specific models enhance the retrieval performance. The results also indicate that categorizing queries based on this feature yields the best results in terms of the average retrieval performance compared to the course-specific and the general ranking models.

Looking at Table VII, we can clearly see that the grade-specific models enhance the NDCG scores by almost 1% in comparison to the general model. In particular, for each grade-specific model (except the one for the fourth grade), there is improvement with respect to the general model.

TABLE IX
RETRIEVAL EFFECTIVENESS OF GENERAL AND FREQUENCY-SPECIFIC RANKING MODELS (VITAMIN SE IS THE BASELINE)

	NDCG						ERR					
Frequency Category	**Vitamin SE**		**General**		**Frequency-Specific**		**Vitamin SE**		**General**		**Frequency-Specific**	
	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10
Non-singleton	0.6446	0.6757	0.8180	0.8186	**0.8249**	**0.8280**	0.5688	0.5740	**0.6586**	0.6614	0.6571	**0.6645**
Singleton	**0.6054**	**0.6521**	0.5923	0.6386	0.5795	0.6301	0.4814	**0.4932**	0.4743	0.4869	**0.4864**	0.4906
AVG	0.6370	0.6711	0.7742	0.7888	**0.7772**	**0.7895**	0.5521	0.5583	0.6205	0.6266	**0.6239**	**0.6307**

Evaluation results with the ERR metric reveal similar trends to those with the NDCG metric. Although we could not improve the average search performance by using course-specific models in terms of the ERR metric in the previous section, with the models learned for each grade category, we achieve a relative improvement of 1% in ERR@5 scores with respect to the general model performance.

Overall, we show that specialized models with respect to the searcher's grade outperform both general LTR models and Vitamin's baseline ranking. This is an important finding as it also supports our hypotheses that students at different grades may have different search characteristics, and hence, they would benefit more from specialized ranking models.

3) Automatic Clustering Approaches for Query-Dependent Learning: To further justify our decision of exploiting query categories (i.e., course, grade, and frequency) while building query-dependent ranking models, we utilize two different automatic query clustering approaches, as in [15] and [40], as further baselines.

1) *Query-Specific Clustering:* In this case, we employ only query-specific features (see Table I), which are also utilized in LTR algorithms. The values for each feature are normalized (among the training query instances) before the clustering. Then, we employ the well-known k-means algorithm (from Python's scikit-learn library) to generate the query clusters. The optimum value for the number of clusters (k) is found to be 3 using the well-known elbow method (based on the plot of the distortion (i.e., squared sum of distances) versus the number of clusters).
2) *Result-Based Clustering:* As an alternative, we exploit the retrieved result lists to cluster the queries. We compute the pairwise distance of queries in terms of the Jaccard similarity score between top 25 results of each query pair. On top of the resulting similarity matrix, we apply spectral clustering. In this case, the best performing value for number of clusters (k) is found to be 4.

In Table VIII, we present the performance of the query-dependent ranking models utilizing these automatically created query clusters and compare to our models based on the query categories. We see that using result-based clustering yields better performance than the query-specific clustering. However, models based on the automatically generated query clusters result in the ranking effectiveness scores (in terms of NDCG and ERR) that are both inferior to the general model and those based on our query categories (i.e., the performance of the models based on the grade category is still the best for most of the cases, as shown in Table VIII). These findings justify our choice of exploiting domain knowledge and specifically, using query categories, for building query-dependent ranking models in the educational search setup.

4) Performance of Frequency-Specific Ranking Models: Confirming the earlier findings in Table III, our feature ablation analysis with the general ranking models has revealed that the most useful feature group for LTR is click-based features, namely, document impression and click count per query. These features provide an important signal about whether a given document is related to the given query. However, a well-known problem is that there are query instances with very sparse click feedback, or none at all. A particular group of such queries are singletons, queries that are issued only once, for which there can be no previous record of document impression and, hence, no click information.

In this section, we investigate performance of ranking models for the singleton queries. Specifically, we categorize our training and test query instances based on their total frequency, so that those that appear only once in the entire set of queries fall into the singleton category, and the others are called as nonsingleton queries. In our setup, the number of singleton queries is much less than that of the nonsingleton queries. In this case, evaluation results with NDCG and ERR metrics show slightly different trends, which can be seen from Table IX. For singleton queries, in terms of the NDCG metric, neither the general model nor the singleton-specific model can outperform Vitamins's original ranking with ad-hoc retrieval functions. In terms of ERR, the singleton model outperforms both the general model and Vitamins SE, but only at the cutoff value of 5. These findings imply that singleton queries are less likely to benefit from LTR as click-related features are not available, regardless of how the model is created, that is, using all the available queries or only singleton queries. In contrast, nonsingleton queries are benefiting from the machine-learned ranking, and models specifically trained for such queries can even outperform the general model (as the inclusion of singletons in training such models may cause noise and reduce the performance also for nonsingletons). These findings also justify the click propagation model we propose for singleton queries, as evaluated in the next section.

E. Performance of the Click Propagation for Singletons

As discussed before, we propose a new approach to generate synthetic values for the impression and click count features of the singleton queries, which are otherwise not available for such queries and may mislead the training process.

TABLE X
RETRIEVAL EFFECTIVENESS OF RANKING MODELS FOR ONLY SINGLETON QUERIES

Approach	**NDCG**		**ERR**	
	@5	@10	@5	@10
Vitamin SE	0.6054	**0.6521**	0.4814	0.4932
Singleton Model	0.5795	0.6301	0.4864	0.4906
ClickStream Features	0.5828	0.6335	**0.4885**	0.4911
+ Discount	0.5782	0.6245	0.4826	0.4904
+ Propagation Algorithm	**0.6061**	0.6367	0.4829	**0.5028**

To serve as a further baseline for our approach, we also employ a smoothing method proposed to remedy click sparsity. In [17], Gao *et al.* introduce the notion of the *click-through stream* of a document, that is, the list of all the previous queries for which the document is clicked, and then, they obtain the values of various CTR features (for a given query–document pair) using this stream. We refer the latter set as *clickstream features* to distinguish from the impression and click count features described before. In their first smoothing method, Gao *et al.*[17] address the queries that have only a few clicks. In particular, they first construct a bipartite graph of queries and documents and apply a random walk to expand the click-through streams of documents, over which the click-stream features are computed. Obviously, this approach does not help the queries with no clicks, as we aim to improve in this article. Their second approach, the so-called *discount method*, addresses the latter case. Their method estimates the values of the clickstream features for queries with no clicks based on the values of the features computed when a click-through stream includes only a single query. Intuitively, their strategy assigns a value that is only slightly larger than 0 for clickstream features of queries with no clicks. In our experiments, we computed the values of the clickstream features, and on top of them, we also applied the discount method to handle the singleton queries with no clicks.

As in the previous section, we only train and test the model using singleton queries, and while doing so, we first calculate and append clickstream features for singleton queries and apply the aforementioned methods to obtain approximate values, namely, our propagation algorithm to smooth click-based features and discount method adopted from [17] to smooth clickstream features. The evaluation results for this scenario, given in Table X, show that our proposed algorithm outperforms all of its competitors, that is, not only Vitamin baseline and the model trained for singletons using raw feature values (as described in the previous section), but also the rankers based on the features obtained with the smoothing techniques of [17], for NDCG@5 and ERR@10 metrics. Regarding the smoothing techniques introduced in [17], we observe that while clickstream features slightly improve ranking effectiveness of the model, the discount method used on top of it does not seem to be helpful. The reason can be that the discount method assigns the same approximate feature value for each query instance, and since the number of instances of the singleton model is less compared to the general model, this might introduce noise to the model. Overall, our proposed algorithm improves the performance of the singleton query model by approximately 2%. To sum up, we show that all three query-dependent ranking models usually outperform the general model, and the performance of the frequency-specific model can be further improved by our click propagation algorithm. The additional gains by query-dependent models, despite being modest (i.e., up to 2%), are in line with the literature (e.g., Geng *et al.* [15] also report similar gains for Web search) and important to further improve effectiveness in an educational search engine.

F. Discussions

1) Evaluation in Terms of Learning Gains: All three contributions in this article have a common goal, namely, to improve retrieval effectiveness of educational search engines. A question that arises naturally is to what extent improvement in search effectiveness translates to learning gains. Earlier works evaluate impact of search on learning by conducting pre- and postassessments via tests, summaries, and/or user studies [59], [60]. For instance, Collins-Thompson *et al.* [59] assess learning outcomes in Web search using a laboratory-based user study, where they collect pre- and postsearch questionnaires and a postsearch survey from 42 participants. Obviously, such assessment techniques are not applicable for us, as we employ 900 real queries (extracted from a past search log) submitted by around $3K$ students, who are not available for a postsearch interview. Having said that, earlier studies also show that optimizing search results for educational goals may ultimately improve learning gains of the users. Specifically, the work in [1] presents a retrieval strategy that generates search results tailored for a vocabulary learning task and reports that improving learning gains is attainable using such optimized rankings. Based on the latter finding, here, we also evaluate performance of machine-learned ranking models using rank-aware effectiveness metrics, assuming that a search engine with high effectiveness (i.e., ranking a large number of relevant documents at top positions) would also lead learning gains and leave alternative evaluation scenarios as a future work.

2) Generalizability and Limitations of the Findings: We envision that an educational search engine utilizing the aforementioned machine-learned ranking models can be used by both stakeholders, namely learners (i.e., students) and educators (i.e., teachers). Furthermore, the features employed for training such models should be available in most platforms. Precisely, query-specific, session-based, and click-based features can be obtained from the query logs that are typically stored in modern search engines. Query–document text similarity and document-specific features can be easily computed using textual content and/or metadata of documents. Therefore, for any educational platform with a categorization of users (e.g., based on the grade of students, expertise level of users, etc.) and/or learning materials (e.g., based on the course/subject of documents, metadata of LOs, etc.), general and specialized ranking models as discussed here can be used.

Having said that, as our experimental dataset includes the queries and interactions from K–12 students (specifically, only those from fourth to eighth grades) in the Vitamin platform, certain findings should be interpreted in this context. For instance, the grade-specific ranking models are found to be more effective in our experiments, which coincides with the earlier findings that search behavior of children is affected by grade level of students [19], [49]. If we had search data also for teachers, the situation could be different, for instance, the course-specific model might have been found to be more useful for such users, as they are experts in their subjects and can specify their information needs accurately. While such limitations imply possible directions for future work, our main finding, namely, the importance and applicability of general and specialized ranking models in educational search, still holds.

VII. Conclusion

This article shed light on various aspects of building machine-learned models for ranking in the context of an educational search engine. First, using the query logs of a commercial platform, namely, an educational search engine called Vitamin SE for K–12 students, we identified the features that are most likely to be useful in such a context. We employed these features to train traditional LTR algorithms. We found that it is possible to outperform the original ranking of Vitamin SE by up to 14% and other retrieval baselines, namely, representative ad-hoc and neural methods, by up to 11% and 2.5%, respectively. While such gains are expectable with machine-learned models, our more interesting findings are due to the detailed analysis of feature groups. Our analysis revealed that while query–document click features are the most useful for ranking (as in the case of general-purpose Web search engines), there are domain-specific features, such as documents' course or grade information that also help enhancing the ranking of the educational search engine.

As a second contribution, we built query-dependent ranking models rather than a single general model to answer all the submitted queries. In particular, inspired by the findings of our feature analysis, we trained different models by grouping queries with respect to three different features, namely, based on the course of the query, the grade of the user issuing the query, and the frequency of the query. Our evaluation results showed that query-dependent ranking models are usually superior to both the general model and baseline models built on automatically created query clusters. In particular, by creating different ranking models for queries based on the grade information of the user who issued the query, it is possible to outperform the ranking performance of the general model by up to 1%.

As our third contribution, to improve the ranking performance for the singleton queries that have no associated click information in the query logs, we proposed a click propagation algorithm to propagate the click information from the other, similar, queries with the click information to the singletons. Despite its simplicity, being restricted to educational search context, this algorithm that makes use of clues such as the overlap of results, similarity of the query strings, and similarity of the grade levels (for the submitting students) is found to work very well. Experimental evaluations have shown that it helps to improve the ranking performance for singleton queries up to 2% (relative to the baseline without such click propagation) and outperforms another baseline approach proposed for similar purposes in the literature.

In our future work, we plan to focus on personalized ranking models that take into account each student's learning habits.

References

[1] R. Syed and K. Collins-Thompson, "Optimizing search results for human learning goals," *Inf. Retrieval J.*, vol. 20, no. 5, pp. 506–523, 2017, doi: 10.1007/s10791-017-9303-0.

[2] N. Y. Yen, T. K. Shih, L. R. Chao, and Q. Jin, "Ranking metrics and search guidance for learning object repository," *IEEE Trans. Learn. Technol.*, vol. 3, no. 3, pp. 250–264, Jul.–Sep. 2010, doi: 10.1109/TLT.2010.15.

[3] H. Li, *Learning to Rank for Information Retrieval & Natural Language Processing*. San Rafael, CA, USA: Morgan Claypool, 2011.

[4] O. Chapelle and Y. Chang, "Yahoo! learning to rank challenge overview," in *Proc. Int. Conf. Yahoo! Learn. Rank Challenge*, Jun. 2010, vol. 14, pp. 1–24.

[5] B. B. Cambazoglu *et al.* "Early exit optimizations for additive machine learned ranking systems," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, Feb. 2010, pp. 411–420, doi: 10.1145/1718487.1718538.

[6] G. A. Osorio-Zuluaga and N. D. Duque-Mendez, "Search and selection of learning objects in repositories: A review," in *Proc. 13th Latin Amer. Conf. Learn. Technol.*, Oct. 2018, pp. 513–520, doi: 10.1109/LACLO.2018.00090.

[7] J. Sanz-Rodriguez, J. M. Dodero, and S. S. Alonso, "Ranking learning objects through integration of different quality indicators," *IEEE Trans. Learn. Technol.*, vol. 3, no. 4, pp. 358–363, Oct.–Dec. 2010, doi: 10.1109/TLT.2010.23.

[8] M. A. H. Pimentel, I. B. Sant'Anna, and M. D. D. Fabro, "Searching and ranking educational resources based on terms clustering," in *Proc. Int. Conf. Enterprise Inf. Syst.*, Mar. 2018, pp. 507–516, doi: 10.5220/0006647305070516.

[9] A. Usta, I. S. Altingovde, I. B. Vidinli, R. Ozcan, and O. Ulusoy, "How K-12 students search for learning?: Analysis of an educational search engine log," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2014, pp. 1151–1154, doi: 10.1145/2600428.2609532.

[10] C. Macdonald, R. L. T. Santos, and I. Ounis, "On the usefulness of query features for learning to rank," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2012, pp. 2559–2562, doi: 10.1145/2396761.2398691.

[11] E. Agichtein, E. Brill, and S. Dumais, "Improving Web search ranking by incorporating user behavior information," in *Proc. 29th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Aug. 2006, pp. 19–26, doi: 10.1145/1148170.1148177.

[12] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for Web search using clickthrough data," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2013, pp. 2333–2338, doi: 10.1145/2505515.250566.

[13] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, "A deep relevance matching model for ad-hoc retrieval," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2016, pp. 55–64, doi: 10.1145/2983323.2983769.

[14] Z. Dou, R. Song, J. Wen, and X. Yuan, "Evaluating the effectiveness of personalized Web search," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 8, pp. 1178–1190, Aug. 2009, doi: 10.1109/TKDE.2008.172.

[15] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum, "Query dependent ranking using K-nearest neighbor," in *Proc. 31st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2008, pp. 115–122, doi: 10.1145/1390334.1390356.

[16] D. Downey, S. Dumais, and E. Horvitz, "Heads and tails: Studies of Web search with common and rare queries," in *Proc. 30th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2007, pp. 847–848, doi: 10.1145/1277741.1277939.

[17] J. Gao, W. Yuan, X. Li, K. Deng, and J.-Y. Nie, "Smoothing clickthrough data for Web search ranking," in *Proc. 32nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2009, pp. 355–362, doi: 10.1145/1571941.1572003.

[18] E. Aktolga and J. Allan, "Reranking search results for sparse queries," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2011, pp. 173–182, doi: 10.1145/2063576.2063606.

[19] S. Duarte Torres, I. Weber, and D. Hiemstra, "Analysis of search and browsing behavior of young users on the Web," *ACM Trans. Web*, vol. 8, no. 2, Mar. 2014, Art. no. 7, doi: 10.1145/2555595.

[20] A. Druin *et al.*, "How children search the internet with keyword interfaces," in *Proc. 8th Int. Conf. Interact. Des. Children*, Jun. 2009, pp. 89–96, doi: 10.1145/1551788.1551804.

[21] Y. Kammerer and M. Bohnacker, "Children's Web search with Google: The effectiveness of natural language queries," in *Proc. 11th Int. Conf. Interact. Des. Children*, Jun. 2012, pp. 184–187, doi: 10.1145/2307096.2307121.

[22] C. Eickhoff, P. Dekker, and A. P. de Vries, "Supporting children's Web search in school environments," in *Proc. 4th Inf. Interact. Context Symp.*, Aug. 2012, pp. 129–137, doi: 10.1145/2362724.2362748.

[23] S. Duarte Torres, D. Hiemstra, and P. Serdyukov, "An analysis of queries intended to search information for children," in *Proc. 3rd Inf. Interact. Context Symp.*, Aug. 2010, pp. 235–244, doi: 10.1145/1840784.1840819.

[24] A. Usta, I. S. Altingovde, R. Ozcan, and Ö. Ulusoy, "Re-finding behaviour in educational search," in *Proc. 23rd Int. Conf. Theory Pract. Digit. Libraries*, Sep. 2019, pp. 401–405, doi: 10.1007/978-3-030-30760-843.

[25] K. Gyllstrom and M.-F. Moens, "Wisdom of the ages: Toward delivering the children's Web with the link-based agerank algorithm," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2010, pp. 159–168, doi: 10.1016/j.ipm.2012.12.006.

[26] C. Eickhoff, P. Serdyukov, and A. P. de Vries, "A combined topical/non-topical approach to identifying Web sites for children," in *Proc. 4th ACM Int. Conf. Web Search Data Mining*, Feb. 2011, pp. 505–514, doi: 10.1145/1935826.1935900.

[27] S. D. Torres, D. Hiemstra, I. Weber, and P. Serdyukov, "Query recommendation in the information domain of children," *J. Assoc. Inf. Sci. Technol.*, vol. 65, no. 7, pp. 1368–1384, Jul. 2014, doi: 10.1002/asi.23055.

[28] Z. Yang, J. Yang, K. Rice, J. Hung, and X. Du, "Using convolutional neural network to recognize learning images for early warning of at-risk students," *IEEE Trans. Learn. Technol.*, vol. 13, no. 3, pp. 617–630, Jul.–Sep. 2020, doi: 10.1109/TLT.2020.2988253.

[29] A. Hoppe, P. Holtz, Y. Kammerer, R. Yu, S. Dietze, and R. Ewerth, "Current challenges for studying search as learning processes," in *Proc. 7th Workshop Learn. Educ. Web Data*, May 2018. [Online]. Available: https://www.tib.eu/fileadmin/Daten/dokumente/forschung-entwicklung/LILE_Workshop_SALIENT_position_paper.pdf

[30] T. Yilmaz, R. Ozcan, I. S. Altingovde, and O. Ulusoy, "Improving educational Web search for question-like queries through subject classification," *Inf. Process. Manage.*, vol. 56, no. 1, pp. 228–246, Jan. 2019, doi: 10.1016/j.ipm.2018.10.013.

[31] X. Ochoa and E. Duval, "Relevance ranking metrics for learning objects," *IEEE Trans. Learn. Technol.*, vol. 1, no. 1, pp. 34–48, Sep. 2008, doi: 10.1109/TLT.2008.1.

[32] K. Verbert *et al.* "Context-aware recommender systems for learning: A survey and future challenges," *IEEE Trans. Learn. Technol.*, vol. 5, no. 4, pp. 318–335, Apr. 2012, doi: 10.1109/TLT.2012.11.

[33] M. Peralta, R. Alarcon, K. Pichara, T. Mery, F. Cano, and J. Bozo, "Understanding learning resources metadata for primary and secondary education," *IEEE Trans. Learn. Technol.*, vol. 11, no. 4, pp. 456–467, Oct.–Dec. 2018, doi: 10.1109/TLT.2017.2766222.

[34] S. Tang and Z. A. Pardos, "Personalized behavior recommendation: A case study of applicability to 13 courses on edX," in *Proc. 25th Conf. User Model. Adapt. Pers.*, Jul. 2017, pp. 165–170.

[35] T. Qin, T.-Y. Liu, J. Xu, and H. Li, "LETOR: A benchmark collection for research on learning to rank for information retrieval," *Inf. Retrieval*, vol. 13, no. 4, pp. 346–374, 2010, doi: 10.1007/s10791-009-9123-y.

[36] X. Dai *et al.*, "U-rank: Utility-oriented learning to rank with implicit feedback," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 2373–2380, doi: 10.1145/3340531.3412756.

[37] B. Long and Y. Chang, *Relevance Ranking for Vertical Search Engines*. San Mateo, CA, USA: Morgan Kaufmann, 2014.

[38] H. Wang, A. Dong, L. Li, Y. Chang, and E. Gabrilovich, "Joint relevance and freshness learning from clickthroughs for news search," in *Proc. 21st Int. Conf. World Wide Web*, Apr. 2012, pp. 579–588, doi: 10.1145/2187836.2187915.

[39] V. Jain and M. Varma, "Learning to re-rank: Query-dependent image reranking using click data," in *Proc. 20th Int. Conf. World Wide Web*, Mar. 2011, pp. 277–286, doi: 10.1145/1963405.1963447.

[40] J. Bian, X. Li, F. Li, Z. Zheng, and H. Zha, "Ranking specialization for Web search: A divide-and-conquer approach by using topical ranksvm," in *Proc. 19th Int. Conf. World Wide Web*, Apr. 2010, pp. 131–140, doi: 10.1145/1772690.1772705.

[41] G. Giannopoulos, U. Brefeld, T. Dalamagas, and T. Sellis, "Learning to rank user intent," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2011, pp. 195–200, doi: 10.1145/2063576.2063609.

[42] S. Jiang *et al.*, "Learning query and document relevance from a Web-scale click graph," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2016, pp. 185–194, doi: 10.1145/2911451.2911531.

[43] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2002, pp. 133–142, doi: 10.1145/775047.775067.

[44] F. Can, S. Kocberber, E. Balcik, C. Kaynak, H. C. Ocalan, and O. M. Vursavas, "Information retrieval on turkish texts," *J. Assoc. Inf. Sci. Technol.*, vol. 59, no. 3, pp. 407–421, Dec. 2008, doi: 10.1002/asi.20750.

[45] S. Tyree, K. Q. Weinberger, K. Agrawal, and J. Paykin, "Parallel boosted regression trees for Web search ranking," in *Proc. 20th Int. Conf. World Wide Web*, Mar. 2011, pp. 387–396, doi: 10.1145/1963405.1963461.

[46] P. N. Bennett *et al.*, "Modeling the impact of short- and long-term behavior on search personalization," in *Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Aug. 2012, pp. 185–194, doi: 10.1145/2348283.2348312.

[47] R. W. White, P. N. Bennett, and S. T. Dumais, "Predicting short-term interests using activity-based search context," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2010, pp. 1009–1018, doi: 10.1145/1871437.1871565.

[48] B. Xiang, D. Jiang, J. Pei, X. Sun, E. Chen, and H. Li, "Context-aware ranking in Web search," in *Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2010, pp. 451–458, doi: 10.1145/1835449.1835525.

[49] D. Bilal and J. Gwizdka, "Children's query types and reformulations in Google search," *Inf. Process. Manage.*, vol. 54, no. 6, pp. 1022–1041, Nov. 2018, doi: 10.1016/j.ipm.2018.06.008.

[50] J. Xu and G. Xu, "Learning similarity function for rare queries," in *Proc. 4th ACM Int. Conf. Web Search Data Mining*, Feb. 2011, pp. 615–624, doi: 10.1145/1935826.1935912.

[51] K. Zhou, X. Li, and H. Zha, "Collaborative ranking: Improving the relevance for tail queries," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2012, pp. 1900–1904, doi: 10.1145/2396761.2398540.

[52] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2000, pp. 407–416, doi: 10.1145/347090.347176.

[53] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "Clustering user queries of a search engine," in *Proc. 10th Int. Conf. World Wide Web*, May 2001, pp. 162–168, doi: 10.1145/371920.371974.

[54] N. Craswell and M. Szummer, "Random walks on the click graph," in *Proc. 30th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2007, pp. 239–246, doi: 10.1145/1277741.1277784.

[55] F. Cai, S. Wang, and M. de Rijke, "Behavior-based personalization in Web search," *J. Assoc. Inf. Sci. Technol.*, vol. 68, no. 4, pp. 855–868, Apr. 2017, doi: 10.1002/asi.23735.

[56] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao, "Adapting boosting for information retrieval measures," *Inf. Retrieval*, vol. 13, no. 3, pp. 254–270, Sep. 2010, doi: 10.1007/s10791-009-9112-1.

[57] J. Guo, Y. Fan, X. Ji, and X. Cheng, "MatchZOO: A learning, practicing, and developing system for neural text matching," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2019, pp. 1297–1300, doi: 10.1145/3331184.3331403.

[58] W. Yang, K. Lu, P. Yang, and J. Lin, "Critically examining the "neural hype": Weak baselines and the additivity of effectiveness gains from neural ranking models," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2019, pp. 1129–1132, doi: 10.1145/3331184.3331340.

[59] K. Collins-Thompson, S. Y. Rieh, C. C. Haynes, and R. Syed, "Assessing learning outcomes in Web search: A comparison of tasks and query strategies," in *Proc. 1st ACM Conf. Human Inf. Interact. Retrieval*, Mar. 2016, pp. 163–172, doi: 10.1145/2854946.2854972.

[60] F. Moraes, S. R. Putra, and C. Hauff, "Contrasting search as a learning activity with instructor-designed learning," in *Proc. 27th ACM Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 167–176, doi: 10.1145/3269206.3271676.

Arif Usta received the B.Sc. and M.Sc. degrees in computer science from Bilkent University, Ankara, Turkey, in 2012 and 2015, respectively, where he is currently working toward the Ph.D. degree in computer science with the Department of Computer Engineering.

His research interests include information retrieval and databases, and especially, developing machine learning approaches for the problems in these fields.

Ismail Sengor Altingovde received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Bilkent University, Ankara, Turkey, in 1999, 2001, and 2009, respectively.

He is currently an Associate Professor with the Department of Computer Engineering, Middle East Technical University, Ankara.

Dr. Altingovde was one of the recipients of Yahoo! Faculty Research and Engagement Program Award in 2013, and the Distinguished Young Scientist Award of the Turkish Academy of Sciences in 2016.

Rifat Ozcan received the B.Sc. degree from Bilkent University, Ankara, Turkey, in 2003, the M.Sc. degree from the University of Texas at Arlington, Arlington, TX, USA, in 2004, and the Ph.D. degree from Bilkent University, in 2011, all in computer science.

He is currently a Data and Applied Scientist with the Microsoft Development Center Norway, Oslo, Norway. His research interests include information retrieval, efficiency, and effectiveness of search engines.

Özgür Ulusoy received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 1992.

He is currently a Professor with the Department of Computer Engineering, Bilkent University, Ankara, Turkey. He has authored or coauthored more than 130 articles in archived journals and conference proceedings. His current research interests include Web databases and Web information retrieval, multimedia database systems, social networks, and cloud computing.