Supporting Keyword Search in Product Database: A Probabilistic Approach

Huizhong Duan[1], ChengXiang Zhai[2], Jinxing Cheng[3], Rohit Kumar[4]
University of Illinois at Urbana-Champaign[1,2]
Walmart Labs[3,4]
{duan9[1], czhai[2]}@illinois.edu, {jim[3], rohit[4]}@walmartlabs.com

ABSTRACT

The ability to let users search for products conveniently in product database is critical to the success of e-commerce. Although structured query languages (e.g. SQL) can be used to effectively access the product database, it is very difficult for end users to learn and use. In this paper, we study how to optimize search over structured product entities (represented by specifications) with keyword queries such as "cheap gaming laptop". One major difficulty in this problem is the vocabulary gap between the specifications of products in the database and the keywords people use in search queries. To solve the problem, we propose a novel probabilistic entity retrieval model based on query generation, where the entities would be ranked for a given keyword query based on the likelihood that a user who likes an entity would pose the query. Different ways to estimate the model parameters would lead to different variants of ranking functions. We start with simple estimates based on the specifications of entities, and then leverage user reviews and product search logs to improve the estimation. Multiple estimation algorithms are developed based on Maximum Likelihood and Maximum a Posteriori estimators. We evaluate the proposed product entity retrieval models on two newly created product search test collections. The results show that the proposed model significantly outperforms the existing retrieval models, benefiting from the modeling of attribute-level relevance. Despite the focus on product retrieval, the proposed modeling method is general and opens up many new opportunities in analyzing structured entity data with unstructured text data. We show the proposed probabilistic model can be easily adapted for many interesting applications including facet generation and review annotation.

1. INTRODUCTION

In e-commerce, being able to let users explore the inventory conveniently has become a necessary and critical service for online retailers. Usually, the product inventory is stored in a structured/semi-structured database, where each entry is an entity representing a particular product and each column describes an aspect of the product. Table 1 shows an example database of laptops. In this example, we have seven entities and four attributes. Each attribute may have one or more values. We refer to each attribute-value pair as a *specification* (*abbr. spec*). Each entity is therefore represented by a set of specifications (specs). Traditionally, such data storage can be conveniently accessed by structured queries (e.g. SQL). For example, the query

*select * from laptop where hard-drive > 500G and blu-ray = "Yes"*

searches for laptops that have a large hard disk and blu-ray player. However, end users rarely understand the semantics of such structured queries, and even for a user who is familiar with the query language, it is still a challenge to construct effective queries due to the lack of knowledge of the data. For example, to search for laptops with dedicated graphics cards, we may have to write a long query listing all types of graphics cards except those integrated ones. It is thus necessary to allow users to search for products using keyword queries to express their preferences. Unfortunately, such natural language keyword queries do not clearly specify which products should be returned, thus making it rather challenging to accurately rank product entities so that highly relevant products would be ranked on the top. Traditional methods based on keyword matching are unlikely to work well due to the vocabulary gap between the product specifications and the keywords people use in search queries. Indeed, how to optimize ranking of product entities in response to a keyword preference query has not been well studied in the existing work and is largely an open research question. (See Section 2 for more details.)

As repeatedly shown in many other information retrieval (IR) tasks, the accuracy of a search system is largely determined by the soundness of the retrieval model adopted.

Table 1: Example laptop database

Brand	Hard Drive	Graphics	Blu-ray
HP	750G	Radeon HD 7690M XT	No
Dell	782G	NVIDIA N13P-GS	Yes
Acer	500G	Intel HD	No
Asus	128G	UMA	No
Acer	500G	Radeon HD 7640G	Yes
Asus	750G	Intel HD Graphics 3000	No
Sony	640G	Intel HD Graphics 4000	No

Proceedings of the VLDB Endowment, Vol. 6, No. 14

The lack of a sound retrieval model for product ranking, thus, has hindered the progress in optimizing the ranking accuracy of keyword search in product database. The main goal of our work is thus to develop a sound general probabilistic model for product entity retrieval that can be used in *all* keyword-based product entity search engines to optimize ranking accuracy. Since probabilistic retrieval models have enjoyed much success for ad hoc text retrieval tasks and can be well justified theoretically based on the probability ranking principle [24], and the query likelihood retrieval model[1], which can be derived based on probability ranking principle using query generation decomposition [18], is quite effective [22, 26], we follow a similar process of probabilistic reasoning, and propose a novel probabilistic model for product entity retrieval based on query generation.

In the proposed model, ranking is primarily based on the probability that a user interested in an entity will pose the query. The model attempts to simulate the query formulation process of a user and score each entity e for a query q based on the conditional probability $p(q|e)$ which captures the likelihood that a user who likes entity e would use query q (to retrieve entity e). Essentially, we associate with each candidate entity a hypothesis "user likes this entity" , and use the observed query q as evidence to infer which hypothesis is true (equivalently which entity is liked by the user). The posterior probability of each hypothesis (equivalently each entity) can then be used for ranking product entities for a given query. Such a model can naturally incorporate prior preferences over product entities into ranking in addition to modeling how well an entity matches a query; it can also naturally separate the two subtasks, i.e., entity type matching and entity preference scoring. As a first step in studying probabilistic models for product entity ranking, in this paper we focus on studying the second subtask of preference scoring. This is mainly because many existing product retrieval systems allow a user to choose a product category (which is quite easy for a user to do) in addition to entering keyword preferences, thus the main challenge in improving such a system is to improve the preference modeling. Naturally, an important future work would be to also study the orthogonal problem of entity type matching using the proposed general probabilistic framework.

A key component in our model from the perspective of preference scoring is to model the probability of using a word w in a query by a user who likes entity e, i.e., how we "generate" a query word from an entity. We propose to refine this component based on the attributes of product entities, which roughly models the following generation process of a query. A query is generated by repeatedly sampling a word as follows. A user who likes entity e would first sample an attribute to query according to a specification selection model $p(s|e)$ where s is a spec of e, and then sample a word w from an attribute-specific unigram language model $p(w|s)$. Such a decomposition allows us to model a user's preferences at the attribute level.

Different ways of estimating each component model will lead to different variants of ranking functions. We propose and study several ways to estimate the proposed model, leveraging the product specifications as well as the associated text data such as product reviews and logged search queries. In particular, in contrast to the existing way of using a text description as a whole for ranking entities, we treat text data in a novel way by learning attribute-specific language models from it, which can then be used to improve ranking accuracy for product entity retrieval.

Since product entity retrieval has not been well studied, there is no public available test collection that we can use for evaluation. To address this challenge, we created our own test collections from two major e-commerce systems. We run comprehensive experiments to evaluate the proposed models. Our experimental results show that the attribute-level modeling of relevance, enabled by the proposed model, is more effective than the baseline approaches which straightly model relevance at the entity level. Experiments also show that the proposed model can effectively leverage review and search log data to significantly improve product ranking accuracy, and as in the case of text retrieval, smoothing of the language models is critical and a robust estimate based on interpolation of the proposed model with entity-level language model works the best on our datasets.

Although the probabilistic model introduced in this paper is primarily proposed for the purpose of improving product entity retrieval, it also serves as a general approach for modeling structured/semi-structured entity data (e.g. product specifications) coupled with unstructured text data (e.g. user reviews). The model naturally leads to many other interesting applications besides supporting keyword search. We explore the use of the proposed model in two interesting applications: facet generation and review annotation, and demonstrate the effectiveness with promising results.

The main contributions of this paper are:

1. **Probabilistic model for entity ranking:** We proposed the first probabilistic model for ranking product entities based on how well they match a user's keyword preference query which goes beyond the existing naive "black box" modeling of relevance in product entity ranking and models relevance in more detail at the level of attribute preferences. Although we only studied the model for product search in this paper, the proposed model is actually general and can be potentially applicable to ranking of any type of entities in response to keyword queries expressing preferences on multiple attributes of an entity.

2. **Learning attribute-level language model:** We proposed to adapt probabilistic topic model in a novel way to model any associated text data and learn from such data a set of attribute-level language models, which can not only improve accuracy of product ranking for keyword queries, but also support many other important applications such as facet generation and review annotation. We also proposed and studied multiple strategies for incorporating prior knowledge and smoothing to further improve the accuracy and robustness of the model.

3. **Large Scale Evaluation of multiple product entity ranking functions:** We created two datasets for evaluating keyword search in product databases. Both datasets are prepared by extracting the popular categories from major e-commerce systems. We made

[1]In the IR literature, such a probabilistic model is often referred to as "language modeling approach" or a "language model", emphasizing the modeling of text data with a probabilistic model.

a systematic comparison of multiple variants of the proposed general probabilistic model, and found that going beyond the entity-level "black box" modeling of relevance to achieve attribute-level modeling of relevance is important and can significantly improve ranking accuracy, and that text data such as user reviews and logged queries can be effectively leveraged using the proposed probabilistic model to improve product entity ranking.

4. **Exploring novel applications with the probabilistic model:** Beside supporting keyword search, we also explored the use of the proposed probabilistic model in novel applications for entity retrieval and review mining. We found that our model can be easily adapted for many different purposes including facet generation and review annotation, and achieved promising results. It is shown that the probabilistic model we proposed is general and can lead to many interesting studies in analyzing and mining structured entity data coupled with unstructured text data.

2. RELATED WORK

There have been extensive study of supporting keyword queries in databases, but most of the work assume that the returned results are a set of tuples that match all the keywords (see, e.g., [17, 1, 16]) without tackling the important problem of modeling *relevance*, which is critical for a problem such as product entity search. Standard IR models have been adapted to rank objects in databases (e.g., [15]), but the application is a straightforward use of existing retrieval models to score each single attribute, thus assuming the relevance definition implied in a traditional IR model. Liu et al. adapted existing IR model and optimized it for returning results assembled from joining tuples from multiple tables [21]. Demidova et al. proposed an incremental query construction approach based on keyword queries by iteratively obtaining feedback from users [9]. A main focus of this line of work is to deal with complex schema of the database and join tuples from different tables. It has not addressed the special notion of relevance in product search where we must model preferences of users.

Another major limitation of these previous work on supporting keyword queries is that they have mainly focused on lexical level of relevance. For instance, relevance is determined by matching keywords with different attributes of the entity. In this case, queries such as "radeon hd graphics laptop" may work reasonably well, but "laptop with dedicated graphics card" will clearly not work. As the method depends on the description of the entity, keywords that are not in the description cannot be matched well. This problem is commonly known as the " vocabulary gap". Some work have attempted to go beyond this simple notion of relevance and applied probabilistic models in IR to database object ranking [6], but the focus was on leveraging workload data to improve estimation of probabilistic models, and the queries considered are restricted to structured queries rather than plain natural language keyword queries as we address in the paper. Integration of IR and database search has been considered in some previous work, particularly in developing probabilistic models (see, e.g., [10, 11, 12]). A main goal of this line of work to extend standard relational algebra in such a way that it can handle probabilistic weights required for performing IR-style ranking, thus they have not addressed how to optimize ranking of database objects for unstructured keyword preference queries.

A significant amount of work has been done on XML retrieval, which is also related to our work in that the documents dealt with have weak structures and keyword queries are handled. A comprehensive review of the work in this line can be found in [19]. However, there are many important differences between XML retrieval and product entity retrieval. First, the unit of retrieval in XML retrieval is not clearly defined, making it a challenge to rank variable units. However, in product search, the units are very well defined as objects in the database. Second, while XML retrieval also deals with structured documents, the fields are not as well defined in the case of entity attributes. Third, the queries in XML retrieval tend to refer to structures whereas in product search that we are considering in this paper, the queries are pure keyword queries similar to those for Web search. Finally, the notion of relevance has a unique and different interpretation in product search, i.e., a relevant entity object is one whose attribute values match the preferred attribute values of a keyword query. A more general problem of XML retrieval is graph search. The problem is considered more difficult as it does not assume any schema or hierarchical property. Effort has been made to utilize the graph structure to optimize the efficiency and accuracy of keyword search on graphs [14]. But similar to XML retrieval, this line of work differs distinctively from our work in problem setup and research focus. Our work is also different from the entity retrieval problem in the form of expert finding [2] or entity search on the Web [25, 3, 8, 7] as we study explicitly defined entities with well organized structures.

Our work is also related to recent work on leveraging review data for ranking entities [13], in which the reviews associated with each entity are simply used as text representation of an entity, and keyword queries would be matched with these reviews directly to rank entities. Such an approach has ignored the attribute value descriptions of product entities completely, thus they do not offer any formal model for ranking product entities in a database with product specifications. Since it *solely* relies on product reviews, it would suffer from the problem of data sparsity as reviews are only available for some product entities, and even fewer of them have enough content for estimating an accurate model. In contrast, we leverage reviews as well as search queries as supplementary data to improve estimation of probabilistic models that would otherwise have to be based on product specifications only.

There are also some studies on e-commerce related applications based on product search log mining. Li et al. proposed a semi-supervised method trained with search queries and matched products for tagging query keywords with product meta-data [20]. Pound et al. studied facet discovery by mining a search log that contain structured product information [23]. These work are application oriented and utilize task specific mining heuristics. In comparison, our work in this paper is focused on optimizing the keyword search accuracy for product databases. The model we use is general for modeling structured entities for search.

3. PROBABILISTIC MODELS FOR RANKING PRODUCT ENTITY

From the perspective of information retrieval, the problem of keyword search in product database (i.e. product entity retrieval) is related to several other retrieval tasks, but is unique in many ways. First, it is different from a regular text retrieval problem (e.g., ordinary Web search) in that the "documents" are product entities, which are very well structured and that the relevance of a product entity to a query is primarily based on how well the attribute values (i.e. specifications) of the product match the preferences of the user expressed in the query. This calls for fine-grained modeling of relevance at the attribute level. Second, it is also different from the entity retrieval problem in the form of expert finding [2] or entity search on the Web [8, 7] where the data assumed available are free text data and information extraction techniques are often used to extract relevant entities. Third, it is different from XML retrieval in that the schema for the entity database is generally assumed to be fixed, while the queries are keyword preference queries as opposed to the more structured XML queries. Finally, as a special case of keyword search on databases, our problem formulation has a clearly defined unit for retrieval and emphasizes on modeling relevance at a finer level to satisfy the preferences expressed in fuzzy keyword queries.

While many product search systems exist on the Web, there has been surprisingly little research on developing general product entity retrieval models for optimizing ranking accuracy in this special, yet important retrieval task. As was shown in other search tasks, developing computational retrieval models to model relevance accurately is the key to optimizing ranking accuracy in a search task.

To systematically optimize accuracy for product entity retrieval, we propose a novel general probabilistic model adapted the general idea of query likelihood retrieval model, study various ways to refine the model and estimate the component probabilities, and evaluate multiple specific product entity ranking functions obtained through using different estimation methods. Below we first present the proposed probabilistic entity retrieval model. Although our main motivation for developing this model is to optimize product entity retrieval, the model is actually general and potentially applicable to any entity retrieval problem where keyword queries are used to express preferences on various attributes.

3.1 Probabilistic Entity Ranking Based on Query Generation

Formally, we are given a set of entities E. Each entity e in E is described by a list of specifications S_e:

$$S_e = \{s|s \in S\} \quad (1)$$

where each specification $s = (a_s, v_s)$ is an attribute-value pair represented by an attribute name a_s (e.g. "brand") and a value v_s (e.g. "dell"), S is the set of all possible attribute-value pairs. Given a user entered keyword query q, our task is to rank the entities in E according to how likely they will satisfy the user's preferences encoded in q.

Following the derivation of the query likelihood retrieval model in [4], we model the relevance of an entity with conditional probability $p(e|q)$, which can be interpreted as the posterior probability that entity e is liked by the user after we observe the user's query q. With Bayes rule, we have:

$$p(e|q) = \frac{p(q|e) \cdot p(e)}{p(q)} \propto p(q|e) \cdot p(e) \quad (2)$$

Since $p(q)$ is only dependent on the query, it can be ignored for the purpose of ranking entities. Therefore, the ranking function only depends on two component probabilities. The first is $p(e)$, which is the prior probability that entity e is liked by a user (the term "prior" can be interpreted as our belief about the relevance of entity e *before* we even see a query). Intuitively, this prior probability may prefer a popular entity to a non-popular one. The second is $p(q|e)$, which is the likelihood that we would observe query q if e is indeed relevant (i.e., liked by the user). This conditional probability can capture how well entity e matches the query q in the sense that if a user likes entity e, the user would likely pose a query matching the attribute values of entity e. The posterior probability of relevance $p(e|q)$ can be regarded as our updated belief about the relevance of entity e *after* observing the query q; ranking based on this posterior probability would naturally prefer an entity that has a high prior probability as well as matches the given query well.

From retrieval perspective, the inclusion of a prior probability $p(e)$ naturally enables us to incorporate any query-independent factors into our ranking function (e.g. popularity statistics of products). In this paper, however, we do not assume any knowledge about the entities, thus we simply assume a non-informative (uniform) prior. Our ranking function would then boil down to ranking solely based on $p(q|e)$, which is essentially the query likelihood retrieval function which has proven quite effective for regular text retrieval [22, 26]. However, the challenge is now how to further refine $p(q|e)$ so that we can accurately model the special notion of relevance in product search.

Conceptually, we can refine $p(q|e)$ by modeling the process of query formulation of users. Imagine a user likes an entity e, and we would examine the question how such a user would formulate a query in order to retrieve entity e. Intuitively, the user would have to specify two components in the query: 1) the entity type (e.g., "laptop"), and 2) preferences on attribute values for the target entity (e.g., "small", "cheap"). We can thus assume our query has two parts correspondingly, i.e., $q = (q_t, q_p)$, where q_t is a term denoting the desired entity type and q_p is a keyword query expressing preferences on attribute values. A user's choice of q_t is logically independent of the preferences q_p. Thus we have $p(q|e) = p(q_t, q_p|e) = p(q_t|e)p(q_p|e)$. That is, our task now is to model separately how a user expresses the desired category of entity and how a user expresses preference of values on each attribute.

While in general, the selection of entity category may also be uncertain, in virtually all real applications of product search, the categories of all the products in a database are usually known. Indeed, a user is often asked to select a category of products in addition to entering preference keywords. That is, $p(q_t|e)$ is no longer uncertain in most applications and we have $p(q_t = c|e) = 1$ if c is the category of e, and $p(q_t = c|e) = 0$ for all other categories. The consequence of this assumption is that we would only consider entities of the same category as the category selected by the user (since all other entities would have a zero probability for $p(q_t|e)$, thus also a zero posterior probability $p(e|q)$). In the following, we therefore focus on discussing how we further decompose $p(q_p|e)$ and estimate the model. We want to stress, though, that our proposed model can easily accommodate alterna-

tive ways of refining $p(q_t|e)$ (e.g., to accommodate inexact category matching based on ontology).

Again, we refine $p(q_p|e)$ by exploring how a user expresses preferences if the user likes entity e. Intuitively, if a user likes entity e, the user must have liked some of the specifications of e. Thus, it is reasonable to assume that the user would formulate a preference query by first selecting an interesting specification of e and then choosing appropriate words to describe his/her preference on the value of the chosen specification. That is, the probability that a user, who likes entity e, would use word w in the query is given by

$$p(w|e) = \sum_{s \in S} p(w|s)p(s|e) \tag{3}$$

where $p(w|s)$ is the unigram language model probability that a user would use word w in the query if the user likes specification s, satisfying the constraint $\sum_{w \in V} p(w|s) = 1$. Naturally, $p(s|e)$ captures the intensity that users are interested in spec s, as opposed to other specs of the entity. It satisfies $\sum_{s \in S} p(s|e) = 1$.

For example, if e is well known as a cheap small laptop, then we could assume that both $p(\text{“}size = small\text{”}|e)$ and $p(\text{“}price = under\ \$250\text{”}|e)$ are reasonably high, likely higher than other specifications on “RAM” or “warranty”. Also, we would expect the language model conditioned on the specification “$(size = small)$” would give higher probabilities to words such as “small”, “portable”, or “lightweight” than other words such as “fast” or “powerful”.

Thus the probability of generating a multiword preference query q_p based on entity e would be

$$p(q|e) = \prod_{w \in V} [\sum_{s \in S} p(w|s)p(s|e)]^{c(w,q)} \tag{4}$$

where $c(w, q)$ is the count of word w in q. Note that for convenience, here we have dropped the subscript p in q_p and simply use q to denote preference query q_p since there is no concern of ambiguity.

Clearly, our model allows a user to include preferences on multiple attributes in a single query as it should. Intuitively, the model would favor those entities with attribute values whose preference language model can explain the preference words in the query well, effectively capturing the relevance criterion for product ranking, i.e., ranking product entities whose attribute values match the user's preferences in the query well on the top.

To avoid underflow caused by multiplication of small values, we can score an entity based on the log-likelihood of the query given an entity, which leads to the following general scoring function for product entity ranking:

$$score(q, e) = \log p(q|e) = \sum_{w \in V} c(w, q) \log \sum_{s \in S} p(w|s)p(s|e) \tag{5}$$

Since in general, the available data for model estimation would be limited, appropriate smoothing is necessary to avoid zero probabilities. To do this, we assume that there exists a “generic specification” (s_g) whose corresponding specification preference language model is a general background language model θ_B that would give a non-zero probability to every word (token) in our specification database. By allowing such a generic specification to be chosen potentially for every entity, we can ensure that our estimated models would not assign zero probability to any query word that occurs in our database. Specifically, we can assume that with probability λ, the user would choose this generic specification when formulating the query (i.e., $p(s_g|e) = \lambda$), and thus have

$$score(q, e) = \sum_{w \in V} c(w, q) \log \left[\lambda p(w|\theta_B) + (1 - \lambda) \sum_{s \in S} p(w|s)p(s|e)\right] \tag{6}$$

The background language model θ_B can be estimated based on normalized word counts in the entire database.

It is now clear that in order to make such a general model actually work for product ranking, we must be able to estimate the following two component models:

1. **specification selection model ($p(s|e)$)**: this is the probability that a user who likes entity e would include a preference on the specification s in the query.

2. **specification preference language model ($p(w|s)$)**: this is the probability that a user would use word w in the query to describe a preference if the user likes specification s.

By exploring different ways of estimating these two probabilities, we can derive many interesting instantiations of this general model, which would lead to distinct ranking functions. In this sense, our model not only provides a theoretical foundation for optimizing product entity search, but also serves as a constructive road map for exploring many interesting ranking functions. In the following, we will first discuss how to estimate these component models based solely on the product specifications, and then we study how to leverage the available text data for product entities to solve the vocabulary gap problem and improve the model estimation.

3.2 Model Estimation Based on Entity Specifications

As previously stated, the key question in model estimation is to estimate the preference selection probability $p(s|e)$ and preference language model $p(w|s)$. Without assuming any further knowledge or search log data available, we can only use the product specification data stored in the database to estimate our model.

Let us first look at the specification selection probability. Indeed, without additional knowledge, it is very difficult to guess which attributes are more interesting to a user who likes entity e. Therefore a conservative estimate would be to assume each attribute of entity e is equally likely to be selected. We refer to this estimate the

Uniform Specification Selection (USS):

$$p(s|e) = \begin{cases} 1/|S_e| & s \in S_e \\ o & otherwise \end{cases} \tag{7}$$

Clearly, USS is a coarse estimation. As an alternative estimate, rather than assuming a uniform posterior distribution, we assume a uniform prior $p(s) = 1/|S|_u$, where $|S|_u$ is the count of unique specs in S. Then we can derive the new estimate,

Uniform Prior Selection (UPS):

$$p(s|e) = \frac{p(e|s)p(s)}{\sum_{s' \in S} p(e|s')p(s')} = \frac{p(e|s)}{\sum_{s' \in S} p(e|s')} \tag{8}$$

where $p(e|s)$ is assumed to be uniform distributed over all the entities containing spec s, and zero otherwise:

$$p(e|s) = \begin{cases} 1/|E_s| & e \in E_s \\ o & otherwise \end{cases} \quad (9)$$

where E_s is the set of entities that contain spec s.

In effect, this estimate is similar to the Inverse Document Frequency (IDF) used in document retrieval. A specification unique to entity e would be more likely chosen by a user to express a preference in the query, at least more likely than a very popular feature that is shared by many entities.

For the preference language model $p(w|s)$, a reasonable estimate would be to concatenate the attribute name and value into a short text description and normalize the counts of words in such a text description. That is,

Attribute-Value Text:

$$p(w|s) = \frac{c(w,s)}{\sum_w c(w,s)} \quad (10)$$

where $c(w,s)$ is the count of word w in the concatenated text description of specification s.

These estimates can be plugged into our general entity ranking model to obtain different specific ranking functions, which we will evaluate later in the paper.

3.3 Improve Estimation by Leveraging Associated Text Data

The estimators discussed above are based solely on the entity database, which make them general. However, as the text data in the entity database is quite limited, these estimates would unlikely be accurate in capturing users' language models. To solve the problem of vocabulary gap, we propose to leverage the user generated text data to improve the estimation of our model. The most useful data is the search log of product search engine, where we can associate user queries with entities by looking at user engagement behavior. However, this requires a product search engine that have already works reasonably well and moreover, the search log data is usually proprietary and thus can only be leveraged inside industry labs. User reviews, on the other hand, are public, easy to obtain and does not have any prerequisite on search engines. Indeed, product reviews have become increasingly available on the Web. They are composed by users and is thus a homogenous datasource of search queries.

In this paper, we propose a general method for taking advantage of any kind of text data associated with product entities to improve keyword search in product database, including both logged queries and user reviews, by treating them as data samples generated from the models of the corresponding entities.

3.3.1 A Mixture Model of Review Text Data

Without loss of generality, let us consider a "training set" (for our model) composing of a set of entities E and a set of text descriptions R. Each entity $e \in E$ is associated with a text based description $r_e \in R$. If more than one such descriptions are available for an entity, we can combine them to form a long description if they are of the same type, or use them separately to estimate parallel models if they are different. Our key insight in leveraging text descriptions for estimating both the specification selection model and the preference language model is that we can assume the text associated with entity e, r_e, is generated based on entity e through a similar generation process of an expected query. For previous queries that resulted in clicking the entity, this is obvious; for user reviews, we assume that when a reviewer writes a review, the reviewer would first sample a specification of the entity to discuss in the review, and then sample words discussing the selected corresponding attribute and value. Formally, the log-likelihood of observing text description r_e is thus:

$$\log p(r_e|e) = \sum_{w \in V} c(w, r_e) \log \left[\lambda p(w|\theta_B) + (1-\lambda) \sum_{s \in S} p(w|s)p(s|e)\right] \quad (11)$$

We can then use the Maximum Likelihood estimator to estimate both the specification selection model $p(s|e)$ and the specification preference language model $p(w|s)$ by maximizing the following likelihood function of the entire dataset:

$$\mathcal{F} = \sum_{e \in E} \sum_{w \in V} c(w, r_e) \log \left[\lambda p(w|\theta_B) + (1-\lambda) \sum_{s \in S} p(w|s)p(s|e)\right] \quad (12)$$

To do so we employ an Expectation-Maximization (EM) algorithm. In the E-step, we compute the contribution of each specification in generating each word in the text data:

$$p(s|w,e) = \frac{p(w|s)p(s|e)}{\sum_{s' \in S} p(w|s')p(s'|e)} \quad (13)$$

$$p(\theta_B|w,e) = \frac{\lambda p(w|\theta_B)}{\lambda p(w|\theta_B) + (1-\lambda) \sum_{s \in S} p(w|s)p(s|e)} \quad (14)$$

In the M-step, we re-estimate the model parameters:

$$p(s|e) = \frac{\sum_{w \in V} c(w, r_e)(1 - p(\theta_B|w,e))p(s|w,e)}{\sum_{s' \in S} \sum_{w \in V} c(w, r_e)(1 - p(\theta_B|w,e))p(s'|w,e)} \quad (15)$$

$$p(w|s) = \frac{\sum_{e \in E} c(w, r_e)(1 - p(\theta_B|w,e))p(s|w,e)}{\sum_{w' \in V} \sum_{e \in E} c(w', r_e)(1 - p(\theta_B|w',e))p(s|w',e)} \quad (16)$$

where V is the vocabulary set, $c(w, r_e)$ is the count of word w in r_e.

It is important to note that $p(s|e)$ should be non-zero only when $s \in S_e$. That is, when selecting attribute specifications to generate words, we can only select from those valid specifications for the particular entity e, and for different entities, this "feasible" set of specifications would generally be different (since products differ from each other on at least one attribute). We can ensure this property by initializing $p(s|w,e)$ in the following way:

$$p(s|w,e) = \begin{cases} 1/|S_e| & s \in S_e \\ o & otherwise \end{cases} \quad (17)$$

It is also worth noting that the background language model θ_B is a necessary component in the estimation in order to ensure the learned language models $p(w|s)$ are discriminative.

3.3.2 Maximum a Posterior (MAP) Estimation of the Mixture Model

One problem with the MLE is that the EM algorithm can be easily trapped in local maxima. To alleviate this issue, we need to provide some "guidance" to the estimator. Indeed, in practice, we usually have some prior knowledge of the language people use to describe certain attributes of entities. For instance, we expect to see words such as "cheap" and "expensive" when people talk about "price". If such prior knowledge can be incorporated into the estimator, it can guide the algorithm to find more accurate models.

To do this we employ Maximum a Posterior (MAP) estimator. We assume the knowledge is given in the same format as our model, and consider them as conjugate priors. Specifically, we use $p(w|\tilde{a})$ to denote the prior probability of using word w to describe attribute a, reflecting our belief in the language people use in general. We then maximize the posterior probability of the model. This is done by applying Dirichlet prior to Equation 16:

$$p(w|s) = \frac{\mu p(w|\tilde{a}_s) + \sum_{e\in E} c(w,r_e)(1-p(\theta_B|w,e))p(s|w,e)}{\mu + \sum_{w'\in V}\sum_{e\in E} c(w',r_e)(1-p(\theta_B|w',e))p(s|w',e)} \quad (18)$$

where a_s is the attribute (name) of spec s.

Generating Prior Knowledge. Clearly, it is infeasible to manually create all priors. To automatically discover such knowledge, we employ a co-occurrence analysis algorithm. More specifically, we analyze the co-occurrences of attribute names and all keywords in the text data. We use normalized pointwise mutual information (NPMI) [5] to compute the pseudo counts. NPMI for word w and attribute a is defined as:

$$i_n(w,a) = \frac{\log \frac{p(w,a)}{p(w)p(a)}}{-\log p(w,a)} = \frac{\log \frac{N\cdot c(w,a)}{c(w)c(a)}}{\log \frac{N}{c(w,a)}} \quad (19)$$

where $c(w)$ and $c(a)$ are counts of sentences that word w and (the name of) attribute a appear, respectively; $c(w,a)$ is the count of sentences that w and a co-occur; N is the total number of sentences in the text data. It is worth noting that an attribute name could be a multi-word phrase. Therefore, we allow partial counts when dealing with occurrences. For instance, if a sentence contains word "screen" but not "size", we count it as 1/2 times of occurrence for attribute "screen size".

One nice property of NPMI is that its range of values is $[-1,+1]$. The upper bound and lower bound indicate perfect positive and negative correlation, respectively; value 0 indicates complete independence. We only keep words that have positive correlation to the attributes:

$$i'_n(w,a) = \begin{cases} i_n(w,a) & i_n(w,a) > 0 \\ o & otherwise \end{cases} \quad (20)$$

Then we compute $p(w|\tilde{a})$ by normalizing the above concurrence measure over all the words:

$$p(w|\tilde{a}) = \frac{i'_n(a,w)}{\sum_{w'\in V} i'_n(a,w')} \quad (21)$$

3.3.3 Smoothing and Interpolation

While the estimated $p(s|e)$ and $p(w|s)$ can be directly plugged into the general ranking model, the specification selection model cannot be used for "unseen" entities that are not associated with any text data, because the above estimation methods do not estimate $p(s|e)$ for them. One way to circumvent this issue is to "back off" to $p(s)$ for these entities, where $p(s)$ is computed as:

$$p(s) = \sum_{e\in E} p(s|e)p(e) \quad (22)$$

With the assumption of uniform distribution of $p(e)$, this back-off model can be easily computed based on the learned parameters.

An even better solution is to combine these estimates with the estimates discussed in Section 3.2 through interpolation. In general, such an interpolation would allow us to utilize all the available evidence and has been shown to be an effective strategy in our experiments.

Note that given the text data, we could also build a language model directly for each entity. Indeed, such a "blackbox" strategy could give us a query generation model (i.e., $p(w|e)$) directly, which we can use to compute the likelihood of a query q conditioned on entity e, thus allowing us to rank these entities that have text data. Unfortunately, the model estimated using this strategy can only be used to rank entities that have associated text data. In other words, this model is not *generalizable*. In contrast, our proposed mixture model above enables us to learn *generalizable* component models since $p(w|s)$ can be used for ranking any product entities with specification s. (Clearly such products do not have to have their own reviews/queries). However, for an entity with sufficient text data, the blackbox strategy may help alleviate the potential errors introduced by attempting to infer the latent specification selection process. Thus, a combination of this strategy with the mixture model estimation can potentially improve the robustness of the retrieval model, which is confirmed in our experiments.

3.4 Indexing and Search

With the estimated model parameters, we can use Equation 6 to rank product entities given any query. However, due to efficiency concern, we usually cannot afford to score every product entity for each query. Therefore, we need to build an indexing structure to allow the most promising candidate to be retrieved efficiently.

To do so, we first aggregate the specification selection model and the preference language model offline and compute a multinomial word distribution for each product entity:

$$p(w|e) = \sum_{s\in S} p(w|s)p(s|e) \quad (23)$$

Then we threshold $p(w|e)$ to get a set of index words for each entity:

$$W_e = \{w|p(w|e) > \sigma\}$$

That is, we drop the binding between a term and an entity if we are not confident enough, in order to avoid unnecessary computation wasted on non-promising candidates.

We then build inverted index for words and product entities, based on W_e and $p(w|e)$. In the meanwhile, we rewrite Equation 6 in the same way as in classic language modeling

approach, leading to the following scoring function:

$$score(q,e) = \sum_{w \in q \cap W_e} c(w,q) \log \left[1 + \frac{(1-\lambda)p(w|e)}{\lambda p(w|\theta_B)}\right] + \alpha_q$$
$$\propto \sum_{w \in q \cap W_e} c(w,q) \log \left[1 + \frac{(1-\lambda)p(w|e)}{\lambda p(w|\theta_B)}\right] \quad (24)$$

where

$$\alpha_q = \sum_{w \in V} c(w,q) \log \lambda p(w|\theta_B)$$

is a factor only dependent on query q. Therefore it does not affect ranking of product entities and can be omitted in scoring. With this setup, we can efficiently retrieve the candidates through the word-entity index and score them for ranking.

4. EXPERIMENTS

4.1 Datasets and Evaluation Metrics

Evaluation of the proposed models is challenging since no previous work has studied our problem setup and as a result, there is no existing test collection that we can use for evaluation. We thus had to construct our own datasets.

For this purpose we developed two different evaluation datasets. The first dataset consists of a full crawl of the "Laptop & Netbook Computers" category of *Bestbuy.com*[2], a popular e-commerce website for electronics. Our crawl includes all the specs and reviews of each laptop. There are in total 864 laptops in the database; on average, each entity has 44 specifications. Among these laptops, only 260 of them have user reviews. For evaluation we construct a query set with the following procedure. We first extract a set of simple queries by sampling "laptop" queries from a commercial query log. We filter these queries so that each query contains a well written descriptor on a single attribute. Examples in this query set are "thin laptop", "quad core laptop". We then use the strategy introduced by Ganesan and Zhai [13] to develop a relatively harder query set, simulating long and comprehensive preference queries that we expect users to enter. This is done by randomly combining multiple descriptors from the simple queries. As discussed in [13], because current product search engines cannot support such queries very well, it is difficult to find them in the search log. Thus it is necessary to simulate the query set in such a manner. An example of "difficult" query is "large screen quad core backlit keyboard laptop". To quantitively evaluate the performance of product entity retrieval, we pool together the retrieval results from several baseline models and our proposed models. A well trained industrial annotator is then asked to label the pool of candidates with 5 levels of relevance. On average, 60 entities are judged for each query. In total we obtained 40 queries with annotations for this dataset. On average there are 2.8 keywords per query, and 3.8 keywords per query for the hard queries.

The second dataset consists of several major categories in electronics from another popular e-commerce website, *Walmart.com*[3]. This includes the categories of *laptop*, *camera*, *camcorder*, *TV* and *ebook reader*. In total, the database consists of 1066 entities; on average, each entity has 14.0 specifications. Each product entity is associated with a set of queries by thresholding the number of clicks observed in a search log accumulated for over a year. Queries are randomly sampled from commercial queries in Walmart search engine, and are annotated in the same way as the first dataset. In total we obtain 425 queries with annotations. On average, each query has 2.4 keywords.

The two datasets are referred to as the *Bestbuy* dataset and the *Walmart* dataset, and they are used to evaluate the effectiveness of our model estimated with review data and search log data, respectively. The evaluation is based on the metric of Normalized Discounted Cumulative Gain (NDCG). We use cutoff at 5, 10 and 20 as our primary metrics. Since the first impression is usually delivered by the top results, NDCG@5 measures the immediate satisfaction to the users' search. While NDCG@10 is a widely used metric for the quality of first page results, on product search engines, each result page usually contains more than 10 items (e.g. 15 on *Bestbuy.com* and 16 on *Walmart.com*). This is because users are more willing to explore in product search as compared to Web search. Therefore, we include NDCG@20 to measure the long-range user satisfaction.

4.2 Experiment Results

Table 2 shows the comparison between the baseline models and our proposed probabilistic models that are estimated solely based on entity specs data. LM is the baseline language model estimated by treating all the specs of each entity as a single document. Language model is a well established model for document retrieval [22, 26]. In this work, we use query likelihood language model with Jelinek-mercer smoothing [26]. Another baseline method we compare with is the Query Aspect Modeling (QAM) method proposed by Ganesan and Zhai [13]. In this method, the search query is assumed to be pre-segmented into multiple preference queries, each covering one aspect of the product. Then the method evaluates each of these preference queries separately and combine the results to obtain the ranking for the original query. The best combining method, i.e. average score method, is implemented in our evaluation (denoted QAM). All the other models tested are attribute-value-based models (AM) under the same general framework proposed in this paper. The spec preference language models $p(w|s)$ of all AM models are estimated with the Attribute-Value Text method (see Section 3.2). AM-USS uses Uniform Specification Selection as the estimate of specification selection model (see Section 3.2). We consider AM-USS as another baseline system (where the estimations are not optimized). These baseline models largely represent the state-of-the-art methods for supporting keyword queries in probabilistic databases based on keyword matching.

AM-UPS refines the estimation of specification selection model by adopting the Uniform Prior Selection estimate (Section 3.2). AM-Base-LM and AM-UPS-LM are the weighted interpolation models with AM-Base and LM, AM-UPS and LM, respectively. The best performance on each metric is shown in bold font. We use †, ‡ and § to mark the models that show statistically significant improvement on all evaluation metrics over LM, QAM and AM-Base, respectively.

From the table we can see that when used alone, LM is a relatively strong baseline compared to AM-USS. QAM slightly improves over LM, especially on hard queries. This

[2]http://www.bestbuy.com
[3]http://www.walmart.com

Table 2: Entity retrieval models based on estimation with entity specs on Bestbuy dataset

	All Queries			Hard Queries		
	NDCG@5	NDCG@10	NDCG@20	NDCG@5	NDCG@10	NDCG@20
LM	0.542	0.574	0.612	0.579	0.636	0.701
QAM	0.546	0.577	0.616	0.588	0.642	0.707
AM-USS	0.476	0.520	0.527	0.528	0.592	0.616
AM-UPS§	0.525	0.564	0.604	0.567	0.614	0.671
AM-USS-LM§	0.533	0.577	0.613	0.570	0.640	0.702
AM-UPS-LM†‡§	**0.579**	**0.611**	**0.640**	**0.630**	**0.672**	**0.727**

Table 3: Entity retrieval models based on estimation with entity specs and reviews on Bestbuy dataset

	All Queries			Hard Queries		
	NDCG@5	NDCG@10	NDCG@20	NDCG@5	NDCG@10	NDCG@20
LM	0.680	0.706	0.732	0.592	0.621	0.675
QAM	0.697	0.726	0.757	0.626	0.663	0.726
AM-Base	0.591	0.638	0.696	0.491	0.558	0.639
AM-MLE§	0.686	0.721	0.738	0.625	0.653	0.688
AM-MAP§	0.697	0.721	0.742	0.625	0.676	0.710
AM-Base-UPS §	0.662	0.718	0.784	0.587	0.651	0.743
AM-MLE-UPS†§	0.698	0.736	0.797	0.637	0.682	0.764
AM-MAP-UPS†§	0.711	0.755	0.803	0.666	0.710	0.771
AM-Base-LM§	0.685	0.710	0.734	0.589	0.623	0.671
AM-MLE-LM†§	0.702	0.730	0.743	0.633	0.655	0.694
AM-MAP-LM†§	0.705	0.734	0.758	0.626	0.668	0.715
AM-Base-UPS-LM†‡§	0.722	0.761	0.796	0.661	0.705	0.768
AM-MLE-UPS-LM†§	0.708	0.752	0.797	0.651	0.706	0.768
AM-MAP-UPS-LM†‡§	**0.729**	**0.774**	**0.811**	**0.684**	**0.734**	**0.784**

is in accordance with the findings in [13]. AM-UPS significantly improves over AM-USS with the more accurate estimation for specification selection. Although AM-UPS used alone does not show improvement over LM and QAM, the interpolated model AM-UPS-LM significantly outperforms all three baseline methods. This verifies that the proposed models and the traditional language modeling approach are complementary to each other in effect. While LM method estimates accurate language models for entities with sufficient data, our model provides the generalizability with language models estimated on the attribute level.

Table 3 shows the comparison of different methods based on product review data. Here the LM method is estimated with both specs and review data. AM-Base is another baseline method, where each spec preference language model is estimated using MLE on a long review document constructed by pooling together all the reviews of entities with this spec. In fact, this is just the result of the first iteration of our EM estimation algorithm. AM-MLE and AM-MAP are the MLE estimate (Section 3.3.1) and MAP estimate (Section 3.3.2) of the query generation model, respectively. The $*$-UPS models use UPS estimate (Section 3.2) to interpolate with the corresponding specification selection model. The $*$-LM models are models interpolated with general LM. Again, we use bold font to show the best performance for each metric. †, ‡ and § are used to mark the models that have statistically significant improvement on all evaluation metrics over LM, QAM and AM-Base, respectively.

From the table, we can see that LM is a stronger baseline than AM-Base, and QAM outperforms LM on the hard queries, which are in accordance with previous findings in Table 2. Similarly, we also see AM-Base-LM slightly outperforms LM. This shows that even though sometimes the estimation is not good enough to improve the baseline language model by itself, it still captures a different level of relevance signals that could be used to alleviate the impact of data sparsity.

The proposed estimation methods, i.e. AM-MLE and AM-MAP, improve over the simple estimation method (AM-Base) significantly. This verifies the effectiveness of the MLE and MAP estimation. We can also see that both MLE and MAP outperforms ML slightly, especial for the hard queries. Between the two estimation methods, MAP estimation performs slightly better than MLE. This validates the positive effect of incorporating prior knowledge in model estimation.

The UPS interpolated models ($*$-UPS) all show significant performance boost from the original models. This is also in accordance with our previous finding in Table 2. These findings confirm that the "IDF" effect in specification selection is indeed positively correlated with real users' preferences.

We see improvements in almost all LM interpolated models ($*$-LM) (over their base models). The best performance is achieved by AM-MAP-UPS-LM. The results show that with the use of advanced estimates and product review data, we can train effective models to capture the attribute level of relevance. Compared with the entity language model which models a coarse level of relevance, our models are superior. By interpolating the two types of models, we can achieve even more robust and accurate estimates.

It is worth noting that in the evaluation, the "hard" queries do not necessarily have lower NDCG values compared to the

Table 4: Entity retrieval models based on estimation with entity specs on Walmart dataset

	NDCG@5	NDCG@10
LM	0.384	0.303
AM-USS	0.381	0.299
AM-UPS	0.392	0.308
AM-USS-LM	0.386	0.306
AM-UPS-LM	**0.393**	**0.310**

Table 5: Entity retrieval models based on estimation with entity specs and logged queries on Walmart dataset

	NDCG@5	NDCG@10
LM	0.501	0.392
AM-Base	0.507	0.394
AM-Base-UPS	0.509	0.401
AM-MAP	0.509	0.401
AM-MAP-UPS	0.515	0.406
AM-MAP-UPS-LM†§	**0.518**	**0.411**

"easy" queries, as NDCG is a metric normalized by the ideal DCG value on each query.

In Table 4 and Table 5 we run similar experiments with the Walmart dataset. Again, the experiments confirm that UPS is a better estimate for specification selection model $p(s|e)$ as compared with USS. In general, the use of text data clearly improves the search performance. The experiments also confirm that the mixture model (AM-MAP-$*$) is a superior method for utilizing text data, as it outperforms all baseline systems including LM, AM-Base and AM-Base-UPS. We also observe that the combination of our model and language model would always lead to a performance boost. As in consistent with the evaluation on the Bestbuy dataset, the best performance is achieved by AM-MAP-UPS-LM.

It is worth mentioning that although in this work we separated the product type detection from product preference matching and focused on modeling the latter component, the model we use can actually provide a natural solution to product type detection, by treating product type as a special attribute in product specification. Indeed, this method is used in the experiments with Walmart dataset and achieved good performance.

4.3 Search Efficiency

As discussed in Section 3.4, we can build index to ensure the efficiency of the proposed models by computing an indexing word set for each entity using threshold σ. In this section, we study the impact of the threshold on search efficiency and search accuracy. In Figure 1 we plot the average running time (milliseconds) of AM-MAP-UPS model for different threshold σ. For comparison we also plot the average running time of language model (LM). Both models are ran on a single machine with Intel core i7 2.7GHZ processor and 8GB Ram.

We can see that our model is much more efficient than language modeling approach in general. This is because the learned model is able to capture the most discriminative topical words for each entity and demote/discard the meaningless general words. We also observe that both the running time and the retrieval performance (in Figure 2) stabilized

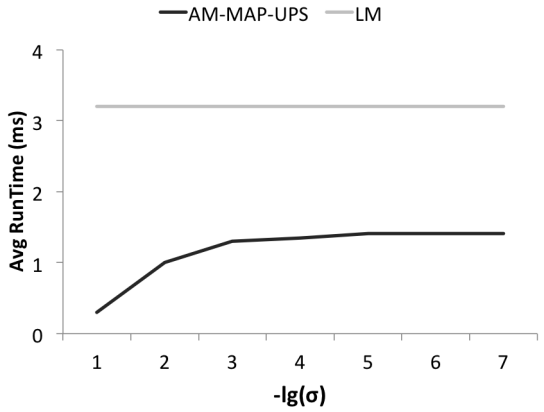

Figure 1: Average search time for different threshold σ

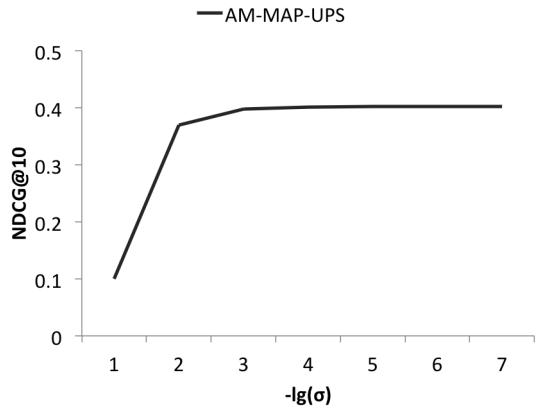

Figure 2: NDCG@10 for different threshold σ

after $\sigma < 1e-4$ This indicates we have included most of the word-entity associations produced by our model.

5. OTHER APPLICATIONS

Although the query generation model is primarily proposed for entity retrieval, it provides a general probabilistic framework which could also lead to many useful applications. In this section, we demonstrate the usability of our model by exploring two novel applications: facet generation and review annotation. We show the parameters in our model can be easily adapted for these tasks and achieve very encouraging results.

5.1 Facet Generation

Facet Generation is an important application for e-commerce websites. Its purpose is to engage users and help them clarify their preferences for the product search engine. To do this, the result page of a product search system usually provides a list of facets, i.e. attributes of products, on a sidebar by the search results. Traditionally, the facet list is generated by hiring experienced analysts to manually select a subset of attributes. The task is very laborious as we need to generate facets for each category of products.

Table 6: Query independent facet generation for laptop category

Most Popular Facets	Least Popular Facets
Graphics Card	ENERGY STAR Qualified
Pointing Device	Multi-Carrier
Audio	Built-in Webcam
Hard Drive Type	Product Height
Brand	Wi-Fi Built In
Computer Hard Drive Size	System Bus
Operating System	BBY Software Installer
Processor	Green Compliance
Video Memory	Color Category
Battery Life	Touchscreen

Table 7: Query specific facet generation

q_1:	*surround sound laptop*	
a_1:	Audio	$\log p = -6.5$
q_2:	*gaming laptop*	
a_1:	Graphics Card	$\log p = -7.3$
a_2:	Gaming Series	$\log p = -11.2$
q_3:	*ssd large screen laptop*	
a_1:	Computer Hard Drive Size	$\log p = -12.5$
a_2:	Screen Size	$\log p = -18.1$
q_4:	*quad core blu ray laptop*	
a_1:	Graphics Card	$\log p = -11.7$
a_2:	Processor	$\log p = -14.2$
a_3:	Blu-ray Player	$\log p = -14.4$
a_4:	Optical Drive	$\log p = -17.3$

Moreover, the manually generated facets do not necessarily match users' interests and cannot reflect the change of such interests over time.

In this section we show how we can use the parameters of our proposed model as a building block to automatically generate the facets based on popular interests. Besides the traditional (query independent) facet generation, we further show that our model can also be used for generating facets tailored to the search intent of each query.

5.1.1 Query Independent Facet Generation

Let us use $p(a)$ to denote the probability that a user is interested in attribute a when searching for products. $p(a)$ can be computed by summing over the marginal probability $p(s)$ of all the specifications defined on attribute a:

$$p(a) = \sum_{s \in \{s|a_s=a\}} p(s) = \sum_{s \in \{s|a_s=a\}} \sum_{e \in E} p(s|e)p(e) \quad (25)$$

With the assumption of uniform $p(e)$, we can easily compute $p(a)$ from our learned models. In Table 6 we show the most and least popular attributes for the laptop category. The results are intuitively very meaningful, as the attributes ranked at the top of the facet list (e.g. *Graphics Card*, *Audio*, *Hard Drive*, *Brand*) are mostly the commonly concerned aspects when users shop for laptop computers, while the facets at the bottom are product features that do not affect buying decisions very much.

5.1.2 Query Specific Facet Generation

Table 8: Sample review annotation for HP - ENVY Spectre Ultrabook

t_1:	*Excellent display*
s:	Graphics Card: Intel HD Graphics 3000
t_2 :	*the best sound on the planet for a notebook*
s:	Audio: Beats Audio
t_3:	*very fast SSD hard drive*
s:	Hard Drive Type: SSD (Solid State Drive)
t_4:	*WiFi is super fast*
s:	Networking: Built-in 10/100/1000 Gigabit
t_5:	*and the weight is perfect for long use on a trip*
s:	Pointing Device: Multitouch Imagepad
t_6:	*Touchpad is a bit skiddish*
s:	Pointing Device: Multitouch Imagepad

The facets are useful for engaging users and helping users refine their shopping preferences, but the static facets are not very effective as they cannot capture the user intentions in the search query. To solve this problem, we further study query specific facet generation.

Let $p(a|q)$ denote the probability that users are concerned about attribute a when issuing query q. $p(a|q)$ can be computed as:

$$\begin{aligned} p(a|q) &= \sum_{s \in \{s|a_s=a\}} p(s|q) \\ &\propto \sum_{s \in \{s|a_s=a\}} p(s)p(q|s) \\ &\propto \sum_{s \in \{s|a_s=a\}} p(s) \prod_{w \in q} [\lambda p(w|\theta_B) + (1-\lambda)p(w|s)] \end{aligned} \quad (26)$$

Using $p(a|q)$ as a scoring function, we can dynamically discover the important facets for each query. Table 7 shows the top most suggestions with their probabilities ($\log p(a|q)$) for several example queries. In general we find our model performs very well, especially for short queries. For all the queries in the examples, the most related facet (i.e. attribute) is ranked at the top or the second position; for queries with multiple focuses (query q_3, q_4), all the related facets are successfully discovered.

5.2 Review Annotation

Another interesting application we explore is review annotation. In this task, we want to automatically detect what feature(s) of the product each review sentence is commenting on. This is done by matching the review sentences with the specs of the corresponding product. With our learned model, we use the conditional probability of $p(s|t,e)$ to rank each sentence t in the review of entity e:

$$\begin{aligned} p(s|t,e) &\propto p(s|e)p(t|s,e) \\ &\propto p(s|e) \textstyle\prod_{w \in t} [\lambda p(w|\theta_B) + (1-\lambda)p(w|s)] \end{aligned} \quad (27)$$

Table 8 shows an example of annotated review. We can see that despite some mistakes (e.g. sentence t_5), the model works reasonably well for the task. Indeed, we observe the model performs very well in annotating short review sentences. Such annotations can be used to better organize the reviews and generate useful opinion summarizations.

6. CONCLUSIONS

This paper is primarily focused on optimizing the accuracy of search for keyword queries in product database, a very important problem not well addressed in the existing work. We proposed a novel probabilistic entity retrieval model based on query generation, and studied different ways of estimating the model parameters based on entity specification and associated text data. The proposed models were evaluated on two newly created test set, and it was shown that they significantly outperformed the baseline systems representing the state-of-the-art keyword matching methods, benefiting from the modeling of relevance at the attribute level. It was shown that with the combined use of the advanced estimates and the associated text data, the accuracy of entity retrieval can be significantly improved from the simple estimates. A robust estimate based on the interpolation of the attribute level language model and the entity level language model achieved the best performance in the experiments. The probabilistic model we proposed also serves as a general approach for modeling structured entity data associated with unstructured text data, as its usability was demonstrated in several novel applications including facet generation and review annotation.

Our work is a first step toward exploring a new family of retrieval models for ranking entities based on keyword preference queries. An obvious future work is to further study the generality of the model and explore the use of the model on different kinds of entities. Our work also opens many doors for further optimization of product search. It would be interesting to see if other IR models can be adapted to achieve even better performance. Another interesting direction is to study how to capture term dependencies, which has been shown to be an important factor in IR, to enhance our model and optimize product search. Finally, as external knowledge bases have matured over the years, an interesting follow-up work would be to study how such data can be incorporated in our model to further improve search accuracy.

7. REFERENCES

[1] S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In *Proceedings of ICDE 2002*, 2002.

[2] K. Balog, Y. Fang, M. de Rijke, P. Serdyukiv, and L. Si. Expertise finding. *Foundations and Trends in Information Retrieval*, 6(3), 2012.

[3] K. Balog, P. Serdyukov, A. P. D. Vries, P. Thomas, and T. Westerveld. Overview of the trec 2009 entity track. 2009.

[4] A. Berger and J. Lafferty. Information retrieval as statistical translation. In *Proceedings of SIGIR*, pages 222–229, 1999.

[5] G. Bouma. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40, 2009.

[6] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic information retrieval approach for ranking of database query results. *ACM Trans. Database Syst.*, 31(3):1134–1168, 2006.

[7] T. Cheng, H. W. Lauw, and S. Paparizos. Entity synonyms for structured web search. *IEEE Transactions on Knowledge and Data Engineering*, 24(10):1862–1875, 2012.

[8] T. Cheng, X. Yan, and K. C.-C. Chang. Entityrank: Searching entities directly and holistically. In *VLDB*, pages 387–398, 2007.

[9] E. Demidova, X. Zhou, and W. Nejdl. A probabilistic scheme for keyword-based incremental query construction. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):426–439, 2012.

[10] N. Fuhr. A probabilistic framework for vague queries and imprecise information in databases. In *VLDB*, pages 696–707, 1990.

[11] N. Fuhr. A probabilistic relational model for the integration of ir and databases. In *SIGIR*, pages 309–317, 1993.

[12] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

[13] K. Ganesan and C. Zhai. Opinion-based entity ranking. *Inf. Retr.*, 15(2):116–150, 2012.

[14] H. He, H. Wang, J. Yang, and P. S. Yu. Blinks: ranked keyword searches on graphs. In *Proceedings of SIGMOD*, pages 305–316. ACM, 2007.

[15] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient ir-style keyword search over relational databases. In *Proceedings of the 29th VLDB conference*, pages 850–861, 2003.

[16] V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In *Proceedings of VLDB 2002*, 2002.

[17] A. Hulgeri and C. Nakhe. Keyword searching and browsing in databases using banks. In *Proceedings of the 18th ICDE conference*, 2002.

[18] J. Lafferty and C. Zhai. Probabilistic relevance models based on document and query generation. 2003.

[19] M. Lalmas. *XML Retrieval (Synthesis Lectures on Information Concepts, Retrieval, and Services)*. Morgan and Claypool, 2009.

[20] X. Li, Y.-Y. Wang, and A. Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proceedings of SIGIR*, pages 572–579, 2009.

[21] F. Liu, C. Yu, W. Meng, and A. Chowdhury. Effective keyword search in relational databases. In *Proceedings of SIGMOD*, pages 563–574. ACM, 2006.

[22] J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the ACM SIGIR*, pages 275–281, 1998.

[23] J. Pound, S. Paparizos, and P. Tsaparas. Facet discovery for structured web search: a query-log mining approach. In *Proceedings of SIGMOD*, pages 169–180, 2011.

[24] S. E. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.

[25] A. P. D. Vries, A. marie Vercoustre, J. A. Thom, M. Lalmas, and I. rocquencourt Le Chesnay Cedex. Overview of the inex 2007 entity ranking track. In *In INEX 2007*, pages 245–251. Springer-Verlag, 2008.

[26] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR'2001*, pages 334–342, Sept 2001.