Differential Query Semantic Analysis: Discovery of Explicit Interpretable Knowledge from E-Com Search Logs

Sahiti Labhishetty*, ChengXiang Zhai*, Min Xie‡, Lin Gong‡, Rahul Sharnagat‡, Satya Chembolu‡
*University of Illinois Urbana-Champaign, USA. {sahitil2,czhai}@illinois.edu
‡WalmartLabs,USA.xiemin.ruc@gmail.com, {lin.gong,rahul.sharnagat,satya.chembolu}@walmartlabs.com

ABSTRACT

We present a novel strategy for analyzing E-Com search logs called Differential Query Semantic Analysis (DQSA) to discover explicit interpretable knowledge from search logs in the form of a semantic lexicon that makes context-specific mapping from a query segment (word or phrase) to the preferred attribute values of a product. Evaluation on a set of size-related query segments and attribute values shows that DQSA can effectively discover meaningful mappings of size-related query segments to their preferred specific attributes and attributes values in the context of a product type. DQSA has many uses including improvement of E-Com search accuracy by bridging the vocabulary gap, comparative analysis of search intent, and alleviation of the problem of tail queries and products.

CCS CONCEPTS

• **Information systems** → *Web mining*; *Query log analysis.*

KEYWORDS

Query word lexicon; E-Com Search Logs; Query difference analysis

ACM Reference Format:

Sahiti Labhishetty*, ChengXiang Zhai*, Min Xie‡, Lin Gong‡, Rahul Sharnagat‡, Satya Chembolu‡. 2022. Differential Query Semantic Analysis: Discovery of Explicit Interpretable Knowledge from E-Com Search Logs. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498503

1 INTRODUCTION

Search logs contain rich information about a user's search behavior and preferences and are thus valuable resources for discovering useful knowledge about users and their search intents. Much work has been done on discovering and exploiting such knowledge. However, most existing works attempt to exploit the knowledge buried in the search logs indirectly, often by using search logs to construct user-engagement features for use in a learning to rank algorithm without explicitly articulating the knowledge exploited (see e.g., [12]). The non-interpretability and the lack of explainability are the major deficiencies of such a strategy and make it inherently non-robust (e.g., the accuracy of learning to rank may even be worse than a single best feature such as BM25 score [13]). It would be much more beneficial to discover explicit interpretable knowledge from search logs since this would allow us to evaluate the reliability of the discovered knowledge and further apply the discovered knowledge to many downstream applications. However, how to design general log-mining methods to turn the raw search log data into such useful interpretable knowledge is a difficult challenge.

In this paper, we tackle such a challenge and study how to mine E-Com search logs to discover interpretable knowledge about the meaning (intent) of those query words and phrases that express a user's preference of specific attribute values of products. The envisioned knowledge is in the form of a semantic lexicon that makes context-specific mapping from a query word or phrase to the preferred attribute values of a product. For example, the meaning of the word "small" in the query "small pot" is small values of the diameter attribute of a pot, whereas the word "kids" in the query "kids pool" refers to dimensions of the pool. We want to discover such knowledge automatically from E-Com search logs, which can enable many applications including, e.g., bridging the vocabulary gap in E-Com search, query understanding and more.

To see more clearly the value of such knowledge in improving E-Com search, consider an example query "big pot for cooking." The token "big" in this query is unlikely mentioned in any description of cooking pots since a cooking pot tends to be described only using the size or diameter of the pot. Clearly, in order to work well on such queries, a search engine must be able to understand the meaning of terms such as "big" in the given context of the query and map them to preferred ranges of product attribute values.

The general question is how to interpret an intent-carrying segment (word or phrase) in a query accurately. Note that this is a quite challenging problem that goes beyond representing the dictionary meaning of a word. Consider a word like "kids" in a query, the literal meaning of "kids" from English dictionaries does not help identify relevant products. Instead, we need an interpretation of the intent of the word "kids" in terms of product attribute values in the *context* of finding a particular type of products.

In this paper, we show that it is feasible to use a computational approach to automatically acquire such knowledge about the meanings of word/phrase segments such as "kids" and "small" in different contexts (e.g., different product types) and construct a Contextualized Query Segment Intent Lexicon (CQSIL) that can also adapt to the changing interpretations of query segments over time.

Specifically, we propose a novel query log analysis method called Differential Query Semantic Analysis (DQSA), which enables us to infer both the attribute of a product referred to by a query segment (which we refer to as an *reference attribute*) and the specific preferred values of the reference attribute. In DQSA, we would

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498503

analyze the difference between a pair of comparable query strings (e.g., "big cooking pots" and "cooking pots") and associate the difference in the queries (e.g., "big") with the corresponding difference in the attribute values of the products that the users have engaged with. This allows us to infer the semantics of the differentiating query segment (i.e., the query words differentiating the two queries) and thus construct a contextualized lexicon consisting of differential query segments, where the context is obtained through the query. We propose and study multiple probabilistic approaches and information-theoretic measures to implement the idea of DQSA.

The CQSIL has many uses. First, it can be used to directly bridge the vocabulary gap to improve results for queries like "big pots for cooking" and also retrieve tail products that are missed due to the knowledge gap. Note that the discovered knowledge from the head queries (e.g., meaning of "big") can be used directly to interpret any tail query where the same query word or phrase is mentioned. Because the discovered knowledge is explicit and interpretable, it also enables many other applications such as enrichment of product description/title, analysing the intent of a word over time as we will further discuss in the paper.

We used Walmart E-Com search log data to perform DQSA to generate a CQSIL, which is evaluated using manual annotations. The results show that DQSA performs well in detecting both reference-attribute and attribute values of a query segment in different contexts, providing a more accurate understanding of the intent of a query segment than conventional correlation analysis. We further show that DQSA can reveal different meanings of the same query segment in different contexts and the constructed lexicon can be used to improve retrieval accuracy and bring up tail products in search results.

2 RELATED WORK

Search logs have been used for improving ranking (e.g., [3, 12, 16, 19, 28]), but such approaches cannot generate explicit interpretable knowledge from search logs. Other work on search log analysis has addressed problems such as query understanding[2, 6, 20, 25, 29], personalization[15, 24], and product relations[27]. The proposed DQSA method differs from this line of previous work in that it enables a more detailed and context-sensitive understanding of a component word or segment in a query; in contrast, the previous work can only analyze the intent of a whole query. The learned knowledge of our approach is thus more generalizable and can be transferred to different queries, including many tail queries. The work [22] proposed an algorithm to discover query context-specific substitutions of query terms or expansion with additional terms. However, there is no explicit context-sensitive semantic representation of query terms, which we attempt to achieve.

Query understanding in E-commerce remains a difficult task [12, 21, 26], which has been tackled using probabilistic models[6, 7], text categorization [2, 20, 29], named entity recognition [8] and embedding approaches[4, 16, 30]. In this line, existing work has often leveraged search logs with a focus on understanding users' shopping intent with respect to different attributes of items, such as the brands, categories, and product types [2, 25, 29]. Other work [10, 11, 17, 23, 26] focused on applications such as query rewriting and query suggestion. Semantically related products from the catalog for given customer queries via distributed representations is explored in [18]. However, most of the analysis in previous works is done only at query level and not at word or phrase level intent understanding. Further, the interpretability of such methods is often a concern as the latent representation does not enable understanding the intent of the query words or allow further downstream applications.

3 DIFFERENTIAL QUERY SEMANTIC ANALYSIS (DQSA)

The key idea of DQSA is to infer the meaning of a word like "small" by analyzing the difference in the user engagement data of two comparable queries that differ only in our target word, i.e., Q and *Q+small* where *Q+small* is the query Q concatenated with word *small*. By comparing the preferred attribute values of products engaged in both queries, we should expect *smaller* attribute values for size are preferred strongly by users of the query *Q+small* compared to the users of the query Q. However, if we make a similar comparison of the distribution of preferred values of another attribute unrelated to size (e.g., color) for the two queries, we would not expect to see so much difference since for both Q and *Q+small*, users would likely have similar preferences for specific colors. This means that we can estimate the likelihood that a particular attribute is the one referred to by *small* based on the divergence between the two empirical distributions of values of the attribute in the engagement data corresponding to the two queries; the larger the divergence, the more likely that the attribute is the reference attribute. Once we identify the reference attribute of a query segment, we can further examine the preferred values of this attribute by the users who entered the query *Q+small* and assume that the most preferred values (e.g., the values of those products which are either most frequently clicked or most frequently added to the cart or most frequently purchased by users) represent the meaning or intent of *small*. This way, we can analyze many such comparable pairs to infer the meanings of many query segments and construct a detailed contextualized intent lexicon automatically.

We now introduce some definitions to facilitate the formal presentation of the DQSA algorithm.

Definition 1. Comparable Query Pair, Query Segment, Base Query, Expanded Query. Two queries Q and $Q \cup X$ is called a *comparable query pair*, where X is a non-empty set of words and is called a *query segment*. Q and $Q \cup X$ are called a *base query* and an *expanded query*, respectively. For example, queries "cooking pot" and "large cooking pot" form a comparable query pair, where the base query is "cooking pot," the expanded query is "large cooking pot," and "large" is a query segment. Note that a query segment can also contain multiple words (e.g., "travel size").

Definition 2. Engagement Products. Given a search log, the *engagement products* of a query Q, denoted by $E(Q)$ is the bag of products that the users who entered query Q have engaged as reflected in the search log data, where the engagement can be based on *clicks*, *add to carts* or *orders*. We also count the number of times a product e in $E(Q)$ is engaged with and denote it by $c(e, E(Q))$.

Definition 3. Product Types and Attributes of a Product Type. In an E-Com search application, all the products are tagged with a particular *product type* from a pre-defined finite set of product types $\mathcal{T}$. We denote a product type by $pt \in \mathcal{T}$. The set of *attributes* of a product type pt consists of all the attributes used to describe a product of type pt, and is denoted by $\mathcal{A}(pt)$.

Definition 5. Attribute Values of a Product. A product (entity) e of product type pt is specified with a set of attribute values denoted by $\mathcal{V}(e) = \{f_v(e,a)|a \in \mathcal{A}(pt)\}$, where $f_v(e,a) \in \mathcal{V}(a)$ is the value of attribute a for product e and $\mathcal{V}(a)$ is the set of all possible values of attribute a.

Definition 6. Engagement Value Distribution. Let $a \in \mathcal{A}(pt)$ be an attribute of product type pt and $E(Q)$ be the set of engagement products for query Q. The *engagement value distribution* of attribute a for query Q, denoted by $p(.|Q,a)$ is the distribution of values of a associated with the products in $E(Q)$.

Intuitively, the engagement value distribution $p(.|Q,a)$ reflects the preferred values of an attribute a by users who entered query Q. The distribution would be close to uniform if the users who entered Q do not have a strong preference for any particular value of attribute a, but may also be heavily concentrated on a particular value or a small set of values if such users strongly prefer a certain kind of values. The main idea of DQSA is to compare the engagement value distribution of the base query with that of the expanded query, i.e., comparing $p(.|Q,a)$ with $p(.|Q \cup X,a)$. If the segment X conveys a preference on a particular attribute $a(X)$, we would expect $p(.|Q \cup X, a(X))$ to be more skewed than $p(.|Q,a(X))$ (thus large difference between the two distributions) but for another attribute b unrelated to X, $p(.|Q,b)$ and $p(.|Q \cup X,b)$ would not have so much difference. Furthermore, once we identify $a(X)$ as the reference attribute of X, we can further assume that the difference between $p(.|Q,a(X))$ and $p(.|Q \cup X,a(X))$ can be attributed to the segment X which is the only difference between the two queries, allowing us to not only infer which attribute X refers to but also the preferred values of that attribute implied by X. In general, we may consider the preferred values are those with the highest probability values given by $p(.|Q \cup X,a(X))$, but we may also want to consider some normalization strategies to bring in other information such as the background value distribution (which can be estimated based on all the products in the database) for the purpose of penalizing some generally popular values.

Note that it is possible that a segment X may refer to multiple attributes. For example, "small" can refer to any of the three dimensions of product size (e.g., height, length, and width of a table). The multiple reference attributes can be obtained by taking multiple top-ranked attributes when we rank those attributes based on their respective difference between the two engagement value distributions of the base and expanded queries, respectively.

In general, the engagement value distribution $p(.|Q,a)$ can be estimated by aggregating the observed values of attribute a in $E(Q)$. As optimization of this estimate is out of the scope of this paper, we simply estimate $p(.|Q,a)$ by using the maximum likelihood estimator as follows:

$$p(v|Q,a) = \frac{\sum_{e \in E(Q), f_v(e,a)=v} c(e,E(Q))}{\sum_{v' \in \mathcal{V}(a)} \sum_{e \in E(Q), f_v(e,a)=v'} c(e,E(Q))}$$

$p(v|Q \cup X,a)$ can be computed similarly by using the engagement products of $Q \cup X$, i.e., $E(Q \cup X)$.

Given $p(.|Q,a)$ and $p(.|Q \cup X,a)$, we can use any reasonable way to measure their difference so as to quantify the impact of adding X to query Q on the engagement value distributions. We will use δ to denote such a deviation measure, thus we can estimate the likelihood that a candidate attribute a is a reference attribute of segment X based on the reference attribute scoring function in the context of query Q:$s(a,X,Q) = \delta(p(.|Q,a), p(.|Q \cup X,a))$. The attribute whose engagement value distribution is impacted most can be assumed to be the most likely reference attribute of X. That is, we can assume those attributes with the highest $s(a)$ scores (i.e., largest deviations) are the reference attributes of X.

There are potentially many different ways to instantiate δ to measure the difference between the two engagement value distributions. In this paper, as an initial study of DQSA, we will explore the following three information-theoretic measures, in hope of establishing some baseline results of the new idea of DQSA, leaving a full exploration of how to optimize this measure as the future work.

1. Entropy Difference: Our first idea is to directly quantify the intuition that the engagement value distribution (EVD) of a reference attribute referred to by segment X (i.e., $a(X)$) in the expanded query is expected to be more skewed as compared with that in the base query. Since entropy is a natural choice for quantifying the randomness of a distribution, we can score an attribute a based on the following Entropy Difference measure:

$$s_{ED}(a,Q,X) = H(p(.|Q,a)) - H(p(.|Q \cup X,a)).$$

2. KL-divergence: Our second idea is to use the KL-divergence [5]. Since an expanded query is more specific than its base query (due to the addition of segment X), we can expect the preferred values of a reference attribute would have increased probabilities according to the EVD of the expanded query, and it is desirable to place more weights on such preferred values. This can be achieved by using $p(.|Q \cup X,a)$ as the first argument of the KL-divergence, leading to the following scoring function:

$$s_{KL}(a,Q,X) = D(p(.|Q \cup X,a)||p(.|Q,a)).$$

3. Jensen-Shannon (JS) Divergence: A disadvantage of the KL-divergence is that it is not bounded and is not comparable between different attributes and across different query pairs. We thus further use the Jensen-Shannon (JS) divergence, which is a smoothed and normalized version of KL divergence for computing the statistical distance [14], defined as follows:

$$s_{JS}(a,Q,X) = [D(p(.|Q,a)||p_m(.)) + D(p(.|Q \cup X,a)||p_m(.))]/2$$

where $p_m(v) = (p(v|Q,a) + p(v|Q \cup X,a))/2$.

Expansions of Basic DQSA: The DQSA algorithm described above is defined for a comparable query pair (Q, Q+X) and it can be extended to aggregate over multiple queries in different ways for broader applications.

1. Multi-Query Analysis. While DQSA operates on a pair of two comparable queries (i.e., Q and $Q + X$), it can be easily generalized to compare multiple queries. For example, we may use DQSA to simultaneously analyze a set of comparable queries $\{Q, Q + X_1, Q + X_2, ..., Q + X_k\}$ by iteratively applying DQSA to each pair (Q, $Q + X_i$), where $i \in [1,k]$. This allows us to compare the meanings of $X_1, ..., X_k$ in the context of query Q. As a specific example, if X_1, ..., X_k are size indicators such as "mini", "small", "medium", "large", "extra large", the learned attribute value distribution for each X_i can be used to further learn a range of "typical" values associated with each of the size indicators for the intended product type of Q. As another example, x_i's can also be different brands, in which case, DQSA would enable us to learn *winning* attribute value of each brand. Such knowledge can be very useful for business intelligence.

2. Aggregated Query Analysis. We may aggregate queries that

are similar to induce a more general category of queries (i.e. a generalized query context), and apply DQSA to each pair in the general category and then aggregate the semantic representations obtained by DQSA from each specific pair. This allows us to use DQSA to interpret a query segment X in a *broader context* that includes a set of queries $C = \{Q_1, ..., Q_n\}$. For example, a general context for a set of queries in E-Com can be the product type associated with the queries, thus by aggregating, we can infer the meaning (intent) of X with respect to a product type/category. Such knowledge is presumably more general than that at the level of a query and thus more useful. Naturally, we can also construct a hierarchy of query context by further combining sub-types into a more general type. In this paper, we explore this line of expansion where we aggregate queries to infer meanings of query segments in the context of different product types. Below we present detailed methods for supporting this kind of inference.

Aggregated Analysis: Given any meaningful context C defined by a set of queries $C = \{Q_1, ..., Q_n\}$, we can score an attribute in context C by aggregating the query level scores as follows: $s(a, C, X) = g(\{s(a, Q, X)|Q \in C\})$, where g is an aggregation function that can be defined in any reasonable way. In our experiments, we primarily focus on learning intent at the level of product types, so our context is defined as all the queries looking for the products of the same type. The product type of a query can be determined by using any existing product type tagger or using the majority voting of the top-ranked products returned by the query.

The combining function g can vary depending on the divergence measures. If a divergence measure is not smoothed or scaled between a fixed range, we can use the rank of an attribute (determined by using a measure to rank attributes) and combine the ranks of an attribute across query pairs to ensure comparability. If a measure is smoothed and scaled between 0 and 1, then the attribute scores can be directly used for the combining function. The scores or ranks are combined as a weighted combination where weights are proportional to the amount of engagement (e.g., number of clicks) of the expanded query because the engagement of the query provides confidence in the divergence scores and thus used as weights. Therefore, the attribute scoring function for segment X in the context of a product type pt is given by one of the following formulas depending on whether the deviation measure is normalized (s_{ED} and s_{JS} are normalized, where as s_{KL} is not):

Rank-based combining function:
$s(a, C, X) = \sum_{i=1}^{n}(1/rank(a, Q_i, X)) * |E(Q_i \cup X)|$, where $rank(a, Q_i, X)$ is the rank of attribute a for the query pair $(Q_i, Q_i \cup X)$ and $|E(Q_i \cup X)|$ is the size of the set of engagement products of query $Q_i \cup X$.

Score-based combining function:
$s(a, C, X) = \sum_{i=1}^{n} s(a, Q_i, X)) * |E(Q_i \cup X)|$ where $s(a, Q_i, X)$ is the score of attribute a for the query pair $(Q_i, Q_i \cup X)$.

Inference of preferred attribute values: Once we identify a reference attribute for a query context Q, we can further identify the preferred values of an identified reference attribute for segment X by analyzing the distribution of the attribute values.

1. Expanded Query distribution (Qe dist): In this approach, once a reference attribute a is identified for segment X, the preferred values of the attribute a would be obtained from the engagement value distribution $p(.|Q \cup X, a)$ where we would select the top-K values with the highest probabilities according to $p(.|Q \cup X, a)$).

2. Pointwise KL-Divergence: As the segment X contributes to the difference between Q and $Q \cup X$, the preferred values of a reference attribute may be more accurately captured by the two EVDs, $p(.|Q \cup X, a)$ and $p(.|Q, a)$. So in this approach, we compute a pointwise KL-divergence score for each value:

$$s(v) = p(v|Q \cup X, a) \log \frac{p(v|Q \cup X, a)}{p(v|Q, a)}$$

and the scores can be used for sorting the values to select the top-K values as the inferred preferred values of attribute a.

To generalize either of these two combining functions to infer the preferred values for a broader context defined by a set of queries, we can compute a weighted average of the scores of each value derived from each query in the context, where the weight of each query Q_i is set to be proportional to the size of the engagement product set of the expanded query $Q \cup X$, and all the weights are normalized to sum up to one to ensure comparability.

Construction of CQSIL: With the methods above, we can use DQSA to construct a CQSIL that maps a segment X and a product type $pt \in \mathcal{T}$ to a reference attribute $a \in \mathcal{A}(pt)$ and the preferred values of attribute a implied by X when used in a query for finding products of type pt. Such an entry can be interpreted as knowledge discovered from the search log about the intent (meaning) of segment X in the context of queries for finding products of pt and can be useful in many ways including improving search accuracy as we will show later. Since a segment often implies constraints or preferences on multiple correlated attributes, we also allow a pair (X, pt) to be mapped to multiple reference attributes. For each reference attribute, we also allow multiple preferred values to reflect the fact that the mapping from a segment to attribute values is generally one to many (the product specifications tend to use very fine-grained values of attributes).

Process-wise, we first identify all the comparable query pairs, from which we can identify the segments that can be analyzed using DQSA. We can then aggregate query pairs of the same product type $pt \in \mathcal{T}$ to form a context of product type $C(pt)$. Next, for each segment X and a context $C(pt)$, we would use $s(a, C(pt), X)$ to obtain a ranked list of all the attributes of product type pt (i.e., $\mathcal{A}(pt)$) to select the reference attributes (e.g., using a threshold). For each reference attribute, the preferred attribute values are then selected using the value scoring function $s(v)$ in its aggregated form based on context $C(pt)$.

Applications of DQSA: The DQSA algorithm is general and can be used to analyze any Q and X as long as we have a pair (Q, Q+X) in the search log. In many ways, DQSA is similar to those general pattern discovery algorithms in data mining that have found widespread applications [9]; we thus expect DQSA to be a powerful new general tool for analyzing search log data to discover useful semantic knowledge that enables many interesting applications. A thorough exploration of all such applications is out of the scope of this paper, but we briefly discuss some specific applications here to highlight the potential impact of the proposed new algorithm.

1. Deep intent analysis. The query segment to be analyzed can be any token, making DQSA a general method for inference about the intent of *any* segment X in the context of *any* query Q as long as we have sufficient engagement data for queries that form a pair of the form $(Q, Q + X)$. For example, X can be a word such

as "trendy" whose meaning clearly vary according to the query context (e.g., "trendy" means different preferred attribute values when Q is "shoes" vs. when Q is "office table"), and DQSA would allow us to understand such differences of the meaning of "trendy" in different query context, enabling a search engine to leverage this understanding to not only improve product matching and search ranking but also improve faceted browsing by showing the ranges of the inferred attribute values associated with the word "trendy". Furthermore, we may compare the inferred attribute values associated with a word such as "trendy" using engagement data from different time periods to reveal how the meaning of "trendy" might have changed over time as users shift their preferences.

3. Utility for tail queries and tail products: Using DQSA, the knowledge learned from head and torso queries (e.g., meaning of segment X in product type T) can be transferred to tail queries to help a search engine better understand the intent of a fresh new query that uses X but has never been seen by the search engine in the past. We can improve the engagement statistics for tail products that may not be initially retrieved due to the knowledge gap.

4. Improvement of product descriptions. DQSA can be used to suggest improvement of descriptions of some products. Specifically, it is straightforward to design an algorithm to leverage the inferred preferred attribute values of a query segment X to improve the description of products that match the preferred attribute values by adding X (and its synonyms) to the description to enable the product description to directly match X in a user's query, effectively addressing the vocabulary gap.

5. Improvement of result relevance prediction. DQSA can be used to improve the search accuracy by leveraging the preferred attributes and attribute values of a segment 'X' to understand a query and use it to expand or re-write query or can be utilized to add features for matching a query to the item attributes and thus improve the prediction of the relevance of a product item as is shown in our experiment results.

4 EVALUATION

To examine the effectiveness of DQSA, we evaluate the lexicon constructed with an E-Com search log data set from the Walmart search engine. We primarily aim to answer the following research questions: **RQ1**. How effective is DQSA for discovering the reference attributes of a query segment and the preferred values of that reference attribute implied by the query segment? **RQ2**. Can the automatically constructed CQSIL be leveraged to improve search results in E-Com? Further, we also study the effectiveness of the different divergence scoring methods and different attribute value inference methods and compare DQSA with a baseline method based on correlation analysis without doing differentiation of queries

In the rest of this section, we will first describe the data set, baseline and selection of the comparable query pairs and then discuss the experiment results. We used independent two-sample t-test for testing statistical significance in all cases.

Data Set: Our E-Com search log data set is composed of tuples *(query, item, item product type, impressions, clicks, add-to-carts, orders)* and each item has a set of attribute tuples like (attribute, attribute value). The attributes in the data are fine-grained; some examples of attributes are *height*, *length*, *width*, *weight*, *color*, *age group* and so on. Note that not all attributes are present for each item/product. Using DQSA and other lexicon construction methods, we differentiate between the fine-grained attributes. We aggregate the engagement data of the same product type and obtain the corresponding query engagement in that product type for each attribute and attribute value. To limit the space of attributes, for each product type, we only consider the attributes that are present in most of the items of that product type (i.e., most popular). In our experiments, we selected *top 20* most popular attribute types in each product type. Additionally, to ensure the reliability of the discovered knowledge, we use a threshold on click engagement to filter any attribute value which has very low engagement (the threshold was set to 50). We selected the most popular $230k$ queries during a one-year window to construct a lexicon since they have sufficient engagement to infer the meaning of a segment confidently.

Selection of Comparable query pairs: When selecting query pairs, we selected pairs such that the segment X is added as a whole to the base query Q, and the segment is added either in the beginning or end. Thus if "2 person" is a segment, neither "white 2 person table", nor "2 tables for a person" would be treated as an Expanded query. A query can belong to multiple product types as it can have engagement items belonging to different product types. To ensure consistency among the aggregated queries, we consider only the dominant product type of a query (i.e., the product type to which most engaged items belong) and consider the engagement of that product type alone. Finally, we used only those query pairs where both the base query and the expanded query have the same dominant product type. Our algorithm to generate query pairs has a time complexity of O(n*k) where n is the number of queries and k is the number of segments. An NLP template matching algorithm can also be used to identify target queries($Q + X$), exploration of it is out of the scope of this paper.

PMI Baseline: To examine the effectiveness of DQSA, we compare it with a strong baseline method based on Pointwise Mutual Information (PMI) between a query *segment* and an *attribute value* as computed using the same search log data. PMI also uses the historical engagement between queries and products. As the lexicon is constructed separately for every product type, we compute the PMI between segments and attribute values separately in every product type. Specifically, all the queries in the product type are used to make the joint distribution $p(X, v)$ where X is a segment and v is an attribute value. The PMI score for an attribute value in a product type is computed as $PMI_{pt}(X, v) = (P(X, v|pt)/P(X|pt)P(v|pt))$. For an attribute a, we compute its PMI score for segment X in product type pt as the average PMI of the attribute values of a as follows, $PMI_{pt}(X, a) = \frac{1}{|\mathcal{V}(a)|} \sum_{v \in \mathcal{V}(a)} PMI_{pt}(X, v)$, where $|\mathcal{V}(a)|$ is the total number of attribute values of a. Ranking attributes based on $PMI_{pt}(X, a)$ score allows us to select the reference attributes in the same way as we do for DQSA. Finally, for each attribute, the attribute values are ranked by their PMI values, thus constructing a lexicon by mapping a (segment,product-type) pair (X,pt) to (reference attribute, attribute values).

4.1 Evaluation of reference attribute inference

To study the effectiveness of different methods for inferring reference attribute, we need to have the ground truth of the reference attributes of the query segments. We address this challenge by

having human annotators manually assign scores to candidate attributes for each (segment, product-type) pair. To avoid ambiguity and reduce variances in human annotations, we have chosen a subset of "size" related segments to focus on annotating size-related query segments, including words such as "big", "small", "huge", "large scale", "travel size", "small space", etc. and query pairs are also generated only for these segments. Evaluating with more types of query segments will be an important future work.

Specifically, for a pair (X,pt), we provide a list of attributes and each attribute is labelled with relevance labels 0, 1 or 2, with 2 indicating that the X in the context of pt implies a preference on the attribute, 1 indicating that the inferred attribute may not be perfect but is related, and 0 meaning the attribute is unrelated to X.

To construct the dataset, we used one of the DQSA methods to get the top 10 attributes and used it as the list of candidate attributes for each (X,pt) where X is a size-related segment. We used a heuristic approach to select such words from the search log. In total, we obtained annotations for around 460 (segment, pt) instances with the help of an internal data tagging team.

This annotated data set was then used to evaluate the ranking accuracy of a reference attribute discovery method in a similar way to retrieval evaluation. We used standard evaluation metrics such as NDCG@10, MRR, Average Rank, which is the average rank of the very first relevant (label 2) attribute in the ranked list.

We compare the three scoring methods for inferring reference attributes (i.e., s_{ED}, s_{KL}, and s_{JS}) of DQSA with the baseline PMI using NDCG@10, MRR, and Average rank. As shown in Table 1, all the DQSA methods outperform the PMI baseline, indicating that differential analysis of queries is indeed more effective than the conventional correlation analysis. One of the major challenges in query understanding is the bias towards popular categories, item attributes and values in engagement data, the DQSA method exactly addresses this challenge through differential analysis and thus performs better than conventional correlation analysis. Among the three DQSA methods, the JS divergence (s_{JS}) performed best. The JS divergence is a smoothed and normalized value which makes the aggregating function over multiple query pairs more robust compared to only KL divergence which is not quite comparable between query pairs. While using ranks, the Average rank of JS divergence is 2.85 indicating that on average the correct attribute can be found within the top 3 attributes types of the ranked list, suggesting that DQSA is quite effective for inferring the reference attributes.

Table 1: Attribute evaluation.

Methods	NDCG@10	MRR	Avg. Rank
KL div	0.7174	0.6781*	2.9425*
JS div	**0.7523***	**0.7039***	**2.853***
Entropy diff	0.6707	0.6260*	2.9764
PMI (baseline method)	0.6902	0.5983	3.3442

∗: Impr. over *PMI_baseline* is statistically significant (p< 0.1)

4.2 Evaluation of preferred attribute values

To study the effectiveness of methods for inferring preferred attribute values, we also need a ground truth. Unfortunately, the number of attribute values in a single attribute is prohibitively huge to obtain manual labels. To address this challenge, we opted to use the *order* engagement data (i.e., purchased products) from the search log of a query as an approximate gold standard. For a query, we can intuitively assume that a purchased product is likely a product with the preferred attribute values based on the query. Therefore for a given query, in the context of a product type pt, we collected all purchased products which are tagged with the same product type and aggregated the engagement statistics for each attribute value. Finally, the number of orders of an attribute value is converted to relevance labels between 0-3. This is done by normalizing the order engagement of an attribute value to form a probability distribution and the attribute values of top 50% of the probability mass are labelled 3, the next 30% are labelled 2 and the rest 20% are labelled 1. The attributes values with zero orders are given label 0. We then evaluated the ranked list of preferred attribute values given by each method in the same way as we evaluate a ranked list of search results using NDCG@10.

While evaluating a lexicon, for a (segment,product-type) pair we consider the top attribute's attribute value distribution and treat it as a ranked list of attribute values. A set of test queries is chosen (whose purchased products are used to approximate ground truth), while the remaining queries are used for constructing a lexicon. For each (X,pt), we select the queries which have the segment and also correspond to type pt, and use the *orders*-based relevance labels of the attribute values as the approximate gold standard, the attribute value distribution corresponding to (X, pt) is evaluated against this using NDCG. The average NDCG of all queries matching (X,pt) is taken as the performance for (X,pt), and the average of the average NDCGs for all lexicon entries gives the performance of the whole dictionary/lexicon. Note that each test query might be used more than once if it has more than one segment in it, in which case, different segments may correspond to different attributes and are evaluated accordingly. In this evaluation, we want to evaluate the (segment, pt) representation according to the lexicon and are not evaluating its ranking performance for a query.

We compare the proposed two methods for inferring the preferred attribute values: 1.*Pointwise KL divergence based distribution* (**Pointwise KL div**); 2. Expanded query engagement value distribution (**Qe dist**). These combined with the reference attribute discovery methods give rise to a total of 6 DQSA methods. In order to only evaluate the attribute values, we consider the cases where the top attribute obtain from the lexicon for the (segment, pt) is correct (annotated 2 as discussed in the manual tagging task). The results of the evaluation are shown in Table 2, where it can be seen that **Qe dist** outperforms **Pointwise KL div** and is also consistent across different attribute methods. In addition, to exclusively evaluate the attribute values, we also perform an overall evaluation for all (segment, pt). In this cases, if the predicted (top) attribute is not correct (not labelled 2), then the NDCG will be 0, whereas if the attribute is labelled 2, then the attribute values are evaluated and

Table 2: Attribute value evaluation.

Methods	NDCG@10	NDCG@20	NDCG@50
KL div, Pointwise KL div	0.5145	0.5430	0.5750
JS div, Pointwise KL div	0.5224	0.5538	0.5845
Entropy diff, Pointwise KL div	0.5094	0.5321	0.5604
KL div, Qe dist	0.5772*	0.6072*	0.6345*
JS div, Qe dist	**0.5799***	**0.6106***	**0.6407***
Entropy diff, Qe dist	0.5627	0.5902*	0.6126

∗: Impr. over Qe dist. is statistically significant (p < 0.1)

NDCG is computed for the ranked list same as the previous test scenario. This evaluation gives an overall performance of the method in identifying the (attribute, attribute values) for a (segment,pt). Based on Table1 and 2, *JS div* combined with *Qe dist* gives the best performance overall, which is confirmed by the results shown in Table 3. We also see that once again, the overall performances of multiple DQSA methods are much better than PMI.

Table 3: Attribute and Attribute value evaluation.

Methods	NDCG@10	NDCG@20	NDCG@50
KL div, Pointwise KL div	0.2952*	0.3115*	0.3298*
JS div, Pointwise KL div	0.3305*	0.3502*	0.3697*
Entropy diff, Pointwise KL div	0.2335*	0.2439*	0.2568*
KL div, Qe dist	0.3295*	0.3466*	0.3621*
JS div, Qe dist.	**0.3647***	**0.3840***	**0.4030***
Entropy diff, Qe dist	0.2487*	0.2609*	0.2709*
PMI	0.1104	0.1363	0.1611

∗: Impr. over *PMI_baseline* is statistically significant ($p < 0.1$)

4.3 Applications of DQSA for search

There are many ways to use DQSA to improve search results and faceted browsing, but optimizing the way of using the lexicon is out of the scope of our paper. Here we use the lexicon to add an additional *boost parameter* [1] into a production search engine based on Solr, where the value of this parameter depends on how close an underlying product is to our top-ranked attribute values. We consider the Solr-based production search engine without our additional boost parameter as our ranking baseline. This baseline uses BM25F for product retrieval and scoring, where the products are ranked based on features like text matching on product title and description, category matching between query and product, and boosting by prior engagement based product popularity.

Boosting method: An E-Com query usually contains information around a specific product type related to users' shopping intent. In this work, we leverage an existing text categorization work that classifies each query into a set of relevant product types. Given a user query, we first decide if it has any size-related segment; if it has multiple size-related segments, we only choose the most popular segment (popularity is obtained by the number of query pairs with that segment in the data). Then for each relevant pt identified by the product type classifier, we use the lexicon for that ($segment$,pt) representation and boost the top 3 attribute values of the predicted (top) attribute. We set the boost score as a constant value of 0.5 in our experiment. Note that the boost score can be made flexible for each ($segment$,pt), but we will leave this for future exploration since our goal is not to optimize the impact on retrieval, but to verify the benefit of the lexicon. For items that match both the pt predicted by the classifier and the (attribute, attribute value) pair, a score of $(boost\_score) * (pt\_score)$ is applied to it in the retrieval scoring function, where pt_score is the score of the pt from the underlying query product type classifier. If (attribute, value) doesn't match, then the score will be zero. As in the case of evaluating preferred values, here a test query is also chosen randomly from the set of all queries which have a size-related segment and the remaining queries are used for constructing the lexicon. We repeated the process 5 times, and the performance is averaged over the 5 runs. We evaluated 1187 test queries overall.

The relevance labels for a (query, item) pair are obtained using the number of clicks which are scaled between 0-3. We also evaluated using manually annotated relevance labels but only for some queries for which they are available, where the labels are between 0-4. In Table 4, we compare the NDCG performance of different methods. The Solr-based ranking baseline is referred to as *Ranking baseline* and the boost parameter can be added using each of the different DQSA methods along with the PMI method. The ranking baseline along with the boost parameter is referred to as *Rankingbaseline*+ $<$ *lexiconmethod* $>$ (+ $<$ *lexiconmethod* $>$ for short) where the $<$ *lexiconmethod* $>$ will be one of the DQSA or PMI methods used to compute the boost parameter.

The results in Table 4 show that the additional boost from our lexicon methods results in better performance than *Ranking baseline* for both click-based and manual relevance label-based evaluations, and we note that the difference is also statistically significant for the click-based evaluation. This suggests that the attributes and attribute values identified by the lexicon for a segment indeed help in identifying relevant products. The improvement from the DQSA methods is more significant than the PMI method, indicating the quality of the lexicon/dictionary created by the DQSA method is better than that by the baseline. The Manual evaluation is computed based on a much smaller benchmark with 200 queries; the small size may explain why the results are not statistically significant, but we can still see a consistent improvement of NDCG from the DQSA methods when compared to ranking baseline methods. As the manually annotated labels are very reliable, this provides more confidence in the performance of DQSA methods. Overall, we can conclude that the DQSA methods improve the search performance over a standard baseline with text match and popularity features. As we have only experimented with a simple way to leverage the lexicon, it is reasonable to assume that more sophisticated methods can potentially lead to even more improvement using our lexicon.

We further looked into the per-query performance improvement of + $<$ *lexiconmethod* $>$ over the *Ranking baseline*. A scatter plot (not shown due to space limit) of NDCG@10 shows that the improvement is quite consistent on the queries where the baseline particularly did not perform well (presumably due to vocabulary gap). To drill down into the area where baseline performs poorly, we considered $NDCG < 0.2$) as low, $0.2 < NDCG <= 0.6$ as medium, and $NDCG > 0.6$ as high. We can observe in our experiment that the increase in average NDCG performance for *Ranking baseline* + *(JS div, Qe dist)* for low NDCG is the most (+0.05), this is compared to +0.03 for medium NDCG, and 0.003 for high NDCG. The increase is also statistically significant in low NDCG cases compared to high NDCG cases. Our hypothesis is that the low NDCG score cases for baseline are likely correlated with more vocabulary gap issues, and thus our method is able to help more. Overall, these results confirm our hypothesis that DQSA can improve search accuracy by mitigating the vocabulary gap.

Qualitative Analysis: We now take a closer look at some of the lexicon entries and some of the ranking results improved by the lexicon. Some size-related segments indicate scale on a dimension. For example *small*, *medium*, and *large* tables can have attribute value distributions corresponding to ranges adjacent to each other. In the following, we show one such example using (JS div, Qe dist) as the lexicon method. For a ($segment$, pt), we only show the top attribute and top-3 attribute values in Table 5. For product type "Freezers," the segments "deep" and "small" are mapped to the attribute *product*

Table 4: Ranking evaluation results.

Methods	Click based evaluation			Manual relevance labels based evaluation		
	NDCG@10	NDCG@20	NDCG@50	NDCG@10	NDCG@20	NDCG@50
Ranking baseline	0.5468	0.5411	0.5278	0.5853	0.5063	0.4698
+KL div, Pointwise KL div	0.5604	0.5576*	0.5467*	0.6015	0.5201	0.4816
+JS div, Pointwise KL div	0.5629*	**0.5601***	**0.5495***	**0.6022**	**0.5205**	**0.4818**
+Entropy diff, Pointwise KL div	0.5595	0.5564*	0.5458*	0.5996	0.5186	0.4805
+KL div, Qe dist	0.5609	0.5584*	0.5476*	0.6007	0.5192	0.4810
+JS div, Qe dist	**0.5631***	0.5600*	**0.5495***	0.6018	0.5198	0.4812
+Entropy diff, Qe dist	0.5619*	0.5587*	0.5482*	0.5992	0.5180	0.4799
+PMI	0.5598	0.5574	0.5466	0.5990	0.5182	0.4806

∗: statistical significance between *Ranking baseline* and *Rankingbaseline+ < lexiconmethod >* for the corresponding *lexicon method*.

height, which makes perfect sense. The attribute values identified are also quite reasonable where "deep" is associated with large height with the preference decrease as the height decreases with "33.5" as the most preferred value, whereas the most preferred value for "small" is "18.5" with larger values less likely. However, we also notice that "33.5" was incorrectly regarded as more preferred to "24.5" for "small". It may be because "33.5" might be generally more preferred among "Freezers" and thus has higher engagement compared to "24.5". This indicates that there is still much room for further improving the DQSA methods we have evaluated. Another example, shown in Table 5, is "small" and "tall" in End tables, which shows how different segments are related to a different type of size dimensions and how they can be identified with the DQSA method automatically; here "small" is associated with "length" or maybe "width" but "tall" only makes sense for "height"; the attribute values identified are also reasonable. As "End tables" are not too tall, the most preferred height is given as "27.72".

Table 5: Sample query segment intent lexicon entries.

Segment	Pt	Attribute	Attribute values
deep	Freezers	product height	'33.5', '21', '18.5' inches
small	Freezers	product height	'18.5', '33.5', '24.5'

Segment	Pt	Attribute	Attribute values
small	End Tables	product length	'17.375', '20', '15.5'
tall	End Tables	product height	'27.72', '21', '21.0'

The lexicon reveals different meanings of a word in different contexts. In Table 6, we show some different attributes associated with the word "mini" in different product types. Such a detailed context-specific meaning/representation of a word is much more useful knowledge than any global representation of the word in the product attribute space domain.

Finally, we look into a specific case where DQSA has improved ranking accuracy to examine whether the improvement is indeed from the use of knowledge in our learned intent lexicon. The query is "tall shelf", where the NDCG@50 for *Ranking baseline* and our method ((Ranking baseline + (JS div, Qe dist) is 0.3392 and 0.5277, respectively. In Table 7, we show the top-three items that our method has retrieved, but are missed by the baseline (they are not in the top 25 of the baseline). From the product titles shown in the table, none of them matches the word "tall", which may explain why the baseline has missed them. However, with our approach, the learned mapping of "tall" to some dimensional attributes (such as "height") has enabled us to use the "boosting" strategy to essentially "match" the word "tall" with large values of preferred values of height, thus bringing up those products which all seem to have a large value

Table 6: Different meanings of "mini" in different contexts

Product Type	Attribute	Attribute values
Refrigerators	productHeight	10.5", 33", 24.8"
Skateboards	productWidth	6",7.5", 6.00"
Laptop Computers	HardDriveCapacity	32GB, 16 GB, 80GB
'Trampolines'	MaximumWeight	220, 200 lbs, 250 lbs
Electric Household Fans	productlength	8.5", 5.75", 10.63"

Table 7: Products missed by a baseline, but retrieved using DQSA boosting (query = "tall shelf").

Rank	Rel. Label	Product & Dimensions
17	2	Mainstays No Tools 5 Shelf Standard Storage Bookshelf... 29.61"W x 15.55"D x 56.10"H
20	2	Best Choice Products 5-Tier Rustic Industrial Bookshelf... 31.5"(L) x 14"(W) x 62.75"(H)
25	3	Better Homes Gardens 4 Cube Storage Organizer... 15.35" L x 17.17" W x 57.83" H

of "height", suggesting that the inferred values are mostly in the correct range. In the lexicon, "tall" in the context of "Bookcases" is correctly mapped to the attribute "productHeight" with attributes values: 71.46",...69.764",...72",..,57.7,..,52",.. 60" and so on. Thus product items matching the height attributes are given a boost in the ranking results, being pushed up correctly. The lexicon can not only bridge the vocabulary gap but also help provide a meaningful explanation to users as to why a product has been retrieved, an important benefit missing in most existing learning methods.

5 CONCLUSIONS

We presented a novel general framework named Differential Query Semantic Analysis (DQSA) for performing semantic analysis of E-Com search logs to obtain explicit interpretable knowledge of the intent of query segments. The main idea of the proposed framework is to mine product engagement log of query pairs that differ just by one query segment and measure the correlation between the differentiating token in the query pair and attribute value distribution differences of the engaged products. We proposed multiple measures to instantiate the framework and evaluated them on an E-Com search log. The results show that DQSA is quite effective for inferring both the reference attributes and the preferred values of a reference attribute. We also show that the learned intent lexicon can be leveraged to improve search by addressing the vocabulary gap issue in E-Com search via translating hard-to-match query tokens into the preferred attribute values, thus enabling direct matching of the query with product specifications. DQSA enables many new applications and opens up an interesting new research direction.

REFERENCES

[1] 2020 (accessed Oct 19, 2020). *The DisMax Query Parser.* https://lucene.apache.org/solr/guide/6_6/the-dismax-query-parser.html

[2] Ali Ahmadvand, Surya Kallumadi, Faizan Javed, and Eugene Agichtein. 2020. JointMap: Joint Query Intent Understanding For Modeling Intent Hierarchies in E-commerce Search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1509–1512.

[3] Qingyao Ai, Yongfeng Zhang, Keping Bi, and W Bruce Croft. 2019. Explainable product search with a dynamic relation embedding model. *ACM Transactions on Information Systems (TOIS)* 38, 1 (2019), 1–29.

[4] Stéphane Clinchant and Florent Perronnin. 2013. Aggregating continuous word embeddings for information retrieval. In *Proceedings of the workshop on continuous vector space models and their compositionality*. 100–109.

[5] Thomas M Cover. 1999. *Elements of information theory.* John Wiley & Sons.

[6] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. A probabilistic mixture model for mining and analyzing product search log. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2179–2188.

[7] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. Supporting keyword search in product database: a probabilistic approach. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1786–1797.

[8] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 267–274.

[9] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. 2007. Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery* 15, 1 (2007), 55–86.

[10] Yeye He, Kaushik Chakrabarti, Tao Cheng, and Tomasz Tylenda. 2016. Automatic discovery of attribute synonyms using query logs and table corpora. In *Proceedings of the 25th International Conference on World Wide Web*. 1429–1439.

[11] Jyun-Yu Jiang and Wei Wang. 2018. RIN: reformulation inference network for context-aware query suggestion. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 197–206.

[12] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. 2017. On application of learning to rank for e-commerce search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 475–484.

[13] Saar Kuzi, Sahiti Labhishetty, Shubhra Kanti Karmaker Santu, Prasad Pradip Joshi, and ChengXiang Zhai. 2019. Analysis of Adaptive Training for Learning to Rank in Information Retrieval. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2325–2328.

[14] Jianhua Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory* 37, 1 (1991), 145–151.

[15] Zitao Liu, Gyanit Singh, Nish Parikh, and Neel Sundaresan. 2014. A large scale query logs analysis for assessing personalization opportunities in e-commerce sites. *WSCD '2014 New York, New York USA, ACM-2014* (2014).

[16] Alessandro Magnani, Feng Liu, Min Xie, and Somnath Banerjee. 2019. Neural Product Retrieval at Walmart. com. In *Companion Proceedings of The 2019 World Wide Web Conference*. 367–372.

[17] Aritra Mandal, Ishita K Khan, and Prathyusha Senthil Kumar. 2019. Query Rewriting using Automatic Synonym Extraction for E-commerce Search.. In *eCOM@ SIGIR*.

[18] Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2876–2885.

[19] Xichuan Niu, Bofang Li, Chenliang Li, Rong Xiao, Haochuan Sun, Hongbo Deng, and Zhenzhong Chen. 2020. A Dual Heterogeneous Graph Attention Network to Improve Long-Tail Performance for Shop Search in E-Commerce. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3405–3415.

[20] Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2006. Building bridges for web query classification. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 131–138.

[21] Parikshit Sondhi, Mohit Sharma, Pranam Kolari, and ChengXiang Zhai. 2018. A Taxonomy of Queries for E-commerce Search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1245–1248.

[22] Xuanhui Wang and ChengXiang Zhai. 2008. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 479–488.

[23] Zhen Wang, Xiang Yue, Soheil Moosavinasab, Yungui Huang, Simon Lin, and Huan Sun. 2019. SurfCon: Synonym Discovery on Privacy-Aware Clinical Data. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1578–1586.

[24] Chen Wu, Ming Yan, and Luo Si. 2017. Ensemble methods for personalized e-commerce search challenge at CIKM Cup 2016. *arXiv preprint arXiv:1708.04479* (2017).

[25] Chao-Yuan Wu, Amr Ahmed, Gowtham Ramani Kumar, and Ritendra Datta. 2017. Predicting latent structured intents from shopping queries. In *Proceedings of the 26th International Conference on World Wide Web*. 1133–1141.

[26] Rong Xiao, Jianhui Ji, Baoliang Cui, Haihong Tang, Wenwu Ou, Yanghua Xiao, Jiwei Tan, and Xuan Ju. 2019. Weakly Supervised Co-Training of Query Rewriting and Semantic Matching for e-Commerce. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 402–410.

[27] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Product knowledge graph embedding for e-commerce. In *Proceedings of the 13th international conference on web search and data mining*. 672–680.

[28] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR Meets Graph Embedding: A Ranking Model for Product Search. In *The World Wide Web Conference*. 2390–2400.

[29] Jiashu Zhao, Hongshen Chen, and Dawei Yin. 2019. A dynamic product-aware learning model for e-commerce query intent understanding. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1843–1852.

[30] Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 575–584.