Rewriting Null E-Commerce Queries to Recommend Products

Gyanit Singh[1], Nish Parikh[1] and Neel Sundaresan[1]
[1]eBay Research Labs, 2065 Hamilton Ave, San Jose, CA, USA
[1]{gysingh, nparikh, nsundaresan}@ebay.com

ABSTRACT

In e-commerce applications product descriptions are often concise. E-Commerce search engines often have to deal with queries that cannot be easily matched to product inventory resulting in zero recall or null query situations. Null queries arise from differences in buyer and seller vocabulary or from the transient nature of products. In this paper, we describe a system that rewrites null e-commerce queries to find matching products as close to the original query as possible. The system uses query relaxation to rewrite null queries in order to match products. Using eBay as an example of a dynamic marketplace, we show how using temporal feedback that respects product category structure using the repository of expired products, we improve the quality of recommended results. The system is scalable and can be run in a high volume setting. We show through our experiments that high quality product recommendations for more than 25% of null queries are achievable.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*search process, query formulation*

General Terms

Experimentation, Human Factors, Measurement

Keywords

Zero recall, e-commerce search, query rewrite, temporal feedback, null query

1. INTRODUCTION

The challenges faced by e-commerce search engines are different from those faced by typical search engines. Product descriptions are shorter and less abundant and less redundant as compared to other web documents. Further, transience of the content (as products sell out they may not be available for search) and difference in vocabulary between the sellers or retailers and potential buyers can result in null results. Consider the following example. A seller selling a clock key that works with many different collector clocks might describe it as a *"universal clock key"*. A buyer needing a 3.5 mm clock key for her clock might search for *"size 5 clock key"*. Another example is that of a recent search on eBay for *"bobby engram rookie class"* which returned null results. The query *"abu garcia hoodie"* on Amazon and the query "large animal enclosures" on Craigslist returned null results. These indicate expired inventory or discontinued products. Such situations cause significant buyer frustration implying potential loss of revenue (more details about this are provided in Section 3.4). For online businesses this means potential loss of customers. As individual null queries are mostly unique, most engines do not possess enough intelligence to handle them and the user experience for such queries is not optimal. The search engine may not be able to find the right inventory matching such queries. Also other features like query recommendations or item recommendations based on per-query behavioral data may not be of much value.

Most solutions to this problem entail rewriting a user's original query to one that would return products close to the user interest. Arguably, the users would prefer to see results that are close to their original query instead of looking at an empty page with no results because they did not know the appropriate vocabulary while typing the query. Query relaxation approaches which drop terms from the original query can be seen deployed in practice. However, naïve dropping of terms might lead to increased recall but provide no quality guarantee. As users typically expect products to be returned within a fraction of a second, run-time efficiency of the algorithm is an important concern. The infrequency of the space of such queries and unstructured nature of the inventory make this task even more challenging. Hence, our goal was to build a system that would be:

- Efficient to run on production for a high volume site like eBay (eBay gets more than 100 million queries a day, a non-trivial portion of these queries are null, and the user expectation is to see a useful output with guaranteed performance).
- Able to preserve the fidelity of the rewrite, so that the items retrieved are as close to the user's original goal as possible.

We have designed a system that provides high quality product recommendations for null queries. The components of the system are such that they can be used for low recall scenarios as well to provide more options to the end user. We tackle the problem of efficiency by using a simple algorithm that works well in practice. The algorithm is able to do several evaluations in parallel, thus keeping the response time of the system acceptable for the end user. We make use of historical products and the eBay taxonomy tree to improve the fidelity of the rewrite. Most commerce sites have a product taxonomy [eBay taxonomy[1]] and the sellers list the inventory using this taxonomy at a particular node in the tree. All the products in a given sub-tree tend to be focused on a theme and as we move to the lowest level nodes (leaf categories) in the tree, these become more specific. Products are ephemeral and dynamic. A query may not be matched to any inventory today, because all relevant inventory might have been sold. However, it is likely that the query would have matched some inventory if it was run yesterday or a week or month before. By looking at a historical database of billions of products and their preferential attachment to specific nodes in the tree, we are able to infer the category affinity for null queries as we receive them. By constraining the relaxed rewrite versions of the null queries to match products only in particular sub-trees of taxonomy we improve the precision of the retrieval. Consider the following example. The query *"hk 450gt pro"* was recently a null query on eBay. A system that runs relaxed rewritten queries might find items matching *"hk pro"* or other subset queries. However, a query like *"hk pro"* would lead to matching inventory in various categories like Motors, Consumer Electronics, Collectibles, Computers & Networking and Sporting Goods disrupting the original intent of the user. However, our rewrite engine consults a database of historical expired items which matches the query *"hk 450gt pro"* to a particular node in the taxonomy (*Toys & Hobbies* $\rightarrow$ *Radio Control & Control Line* $\rightarrow$ *Radio Control Vehicles* $\rightarrow$ *Airplanes & Helicopters*). By running relaxed versions of original query, for e.g. "hk pro" but constraining them to the discovered node in the taxonomy tree, products that more closely align with user's original intent are matched. In this case, other similar models of helicopter kits were found. We test our system for performance and scalability. The system results are also available via an API (see section 6). Evaluation of the system results is also done by multiple judges to see if the products retrieved are interesting, engaging and related to the original user intent (see section 7).

The rest of the paper is organized as follows. In the next section we describe related work. Characteristics of null queries and their impact on users for eBay marketplace are described in section 3. Datasets used for our experiments and terminology used to describe our algorithm are described in section 4. Section 5 describes our system and algorithm (query relaxation and temporal taxonomical feedback) in detail. In section 6, we discuss the scalability aspects of our system and what makes it practical to use it for a high-volume industrial context like eBay. In section 7, we discuss the evaluation of product recommendations from our system. Finally, we discuss future work and conclude in section 8.

2. RELATED WORK

Extensive research has been done in information retrieval around handling difficult queries. Verbose queries are one class of queries which have posed a challenge in information retrieval. Analysis of long verbose queries is described in [3]. Selecting a subset of the original query (a "sub-query") has been shown to be an effective method for improving these queries. In [25], sub-query selection is considered as a sequential labeling problem, where each query word in a verbose query is assigned a label of "keep" or "don't keep" and a Conditional Random Field model is proposed to generate the distribution of sub-queries. One of the main issues in processing these queries is identifying the key concepts that will have the most impact on effectiveness. [16] also describes how long queries frequently contain many extraneous terms that hinder retrieval of relevant documents. In the paper, the authors present techniques to reduce long queries to effective shorter ones that lack those extraneous terms. They also transform the reduction problem into a problem of learning to rank all sub-sets of the original query (sub-queries) based on their predicted quality, and select the top sub-query. [2] describes a technique that uses query-dependent, corpus-dependent and corpus-independent features for automatic extraction of key concepts from verbose queries. The technique achieves higher accuracy in the identification of key concepts than standard weighting methods such as inverse document frequency. Most approaches focus on weighting the important words or phrases and then selecting the best subset query. The former does not consider how words and phrases are used in actual subset queries while the latter ignores alternative subset queries. In [24], an approach is described wherein verbose queries are represented as distribution of subset queries. Evaluating an exponential number of sub-queries for goodness is an expensive operation and may not work in practice. In [5] the authors describe a randomized reduction technique that samples only a small number of candidate sub-queries and yet efficiently explores the sub-query space.

Methods to predict the importance or necessity of terms in queries have been explored. [26] describes several factors that affect term necessity. Search query logs contain traces of users' search modifications. One strategy users employ is deleting terms, presumably to obtain greater coverage. It is described in [13], that it is useful to model and automate term deletion so that queries with no matches can have words deleted till a match is obtained. Algorithms are provided which perform substantially better than the baseline in predicting which word should be deleted from a reformulated query, for increasing query coverage in the context of web search on small high-quality collections.

Research also has been done around null queries. This is another class of difficult queries which lead to zero matches not only because of verbosity but also because of difference between searcher and publisher vocabulary, unavailability of documents (e.g. in the case of product search, some products might have been completely sold out and hence unavailable) and inability of naïve users to form appropriate queries. In [20], the author describes how a shortcoming of existing library systems is the generation of a large percentage of zero-hits, i.e. no library records are retrieved for the corresponding library patron's queries. In [18], the authors describe how product searches can often return null results due to buyer-seller vocabulary mismatch or over-specialization of queries. They propose the use of a Semantic Query Network to help the users reformulate their query or to recommend queries which are likely to return inventory and are related to the user's intent. In [4], the authors describe rare queries. They mention that rare web query sets contain 8-15% of queries which lead to zero recall. In [21], we describe a comprehensive large-scale study of null searches

[1]`http://developer.ebay.com/DevZone/xml/docs/Reference/ebay/GetCategories.html`

on an e-commerce marketplace using eBay as an example. We describe how e-commerce null queries are a non-trivial amount of total commerce traffic and can have a detrimental impact on users and consequently company revenue.

A common mechanism used to improve the relevance in information retrieval is query rewriting. At a high level query rewriting outputs a list of queries [17] (referred to as rewrites) that are related to a given query. Query rewriting is a well studied problem with significant work has been done in generating relevant rewrites with respect to the original query using a variety of methods such as mining query logs and user sessions. Rewriting user queries to broaden coverage is a common technique employed by all search engines. For example, search engines routinely make spelling corrections to queries when retrieving results. In the context of search over structured data sources, textual similarity approaches that treat the query as a bag of words will perform poorly. Past approaches based on log mining [14, 1, 13] discover relationships between terms but they do not address how such knowledge can be exploited in conjunction with statistics of the documents to come up with good rewrites of the queries that preserve fidelity and ensure coverage. [7] proposes a method to relax text queries using taxonomies. Their approach can also be viewed as rewriting queries taking advantage of a taxonomy created by experts. [8] describes a way to efficiently rewrite structured web queries. It shows how null product queries with a structured nature can be rewritten in a principled manner, to surface at least the required number of results while satisfying a low-latency constraint.

There have been extensive studies around relevance feedback as well [9, 12]. The basic idea behind relevance feedback is to take the results that are initially returned from a given query and to use information whether or not those results are relevant to perform a new query. Temporal aspects and their impact on search have also been studied [6, 15]. Temporal aspects are prominent in time-sensitive short document search like in microblogs or e-commerce.

Although many approaches to tackling difficult queries have been studied, no practical approach which can work at industry scale for unstructured document collections has been described. Our work focuses on bridging this gap. We borrow from research work around query rewriting and relevance feedback. We first relax null e-commerce queries. Then we modify the rewrite for better precision. In Rocchio relevance feedback [12], for a given user query, some documents are retrieved. Based on these initially retrieved documents, the query is modified, and documents are retrieved based on the modified query. In many cases, this increases precision as well as recall. Our approach is along these lines but takes advantage of the temporal factor. We take the query and retrieve older expired items. Then based on the meta-data attached to retrieved items from the expired database, we augment the current query with a constraint (e.g. add a taxonomical constraint). We observe that queries rewritten in this way preserve the fidelity of the rewrite better. More specifically our contributions are:

- Design of a scalable rewrite system which can be deployed in practice on a large-scale.
- Inclusion of temporal taxonomical feedback to improve fidelity of rewrites, so that products which match as closely as possible to user's original intent can be retrieved.

3. CHARACTERISTICS OF NULL QUERIES

In this section we describe the characteristics of null queries and their impact on eBay users. A linguistic analysis of null queries is described in section 3.1. General first order characteristic of null queries and comparison with queries that are not null is presented in section 3.2. Uniqueness and temporal volatility of null queries is also discussed in section 3.2. In section 3.3 we discuss the mechanism of generation of null queries and in section 3.4 we show how encountering null queries is detrimental for users and consequently the marketplace. Further details can be found in [21].

3.1 Linguistic Analysis

The quality of experience that can be provided to users is dependent on the amount of understanding about queries the engine possesses. We studied characteristics of null queries along different dimensions and the information around extracted attributes is shown in Figure 1.

In order to detect adjectives, conjunctions, prepositions and articles we used the Stanford NLP Group's POS Tagger trained on an English language corpus [23]. We also used human generated lists for celebrity and brand names. As the tagger was trained on a different corpus and the extraction of attributes was done based on simple string match algorithms, the results are conservative estimates of the actual percentages. All possible nick names of the celebrity or all abbreviations of the brand may not be present in the list hence making these percentages conservative estimates. If the brand or celebrity name to which the query pertains can be known, it can be effectively used for query suggestions and merchandising (even if the query cannot be matched to any relevant inventory).

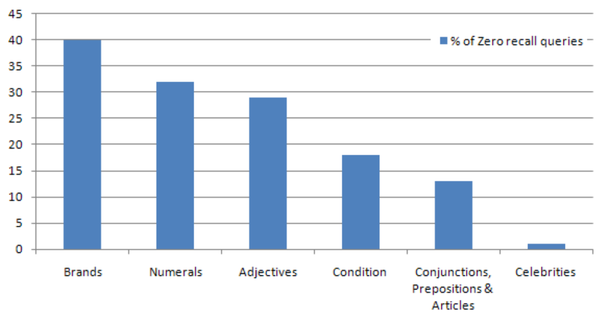


Figure 1: Various attributes found in Zero recall (Null) searches.

Being able to extract properties such as numerals, adjectives, conjunctions from queries can be useful in devising a strategy to predict the necessity of a query term to search for desired items along the lines of work described by Zhao and Callan in [26].

3.2 First Order Analysis, Uniqueness and Temporal Volatility

Term length analysis of queries is provided in Table 1. Null queries tend to be longer. Number of terms used on average is 3.8 compared to 2.3 terms used for non-null queries.

Table 1: First order analysis of null queries and non null queries with respect to (a) Number of terms. (b) Number of characters

Terms per query		Null	Non Null
All searches	Avg	3.83	2.37
	Median	3	2
	StdDev	2.56	1.82
	Max	1971	1286
Unique queries	Avg	4.65	3.8
	Median	3	3
	StdDev	3.47	2.74
	Max	1971	1286

Characters per query		Null	Non Null
All searches	Avg	21.68	13.24
	Median	21	13
	StdDev	13.7	11.9
	Max	4484	4292
Unique queries	Avg	21.3	21.5
	Median	20	20
	StdDev	14.4	15.3
	Max	4484	4292

Null queries are close to being unique. As shown in Table 2 null queries have a repetition factor of 1.4. Non-null queries on the other hand have a repetition factor of 20. Even the most popular null queries do not repeat more than tens of thousands of times within a month. But the most popular non-null query repeats more than millions of times.

Table 2: Repetition factor for null queries and non-null queries. Null queries are almost unique (1.45 repetition factor); whereas non-null queries have a heavy head.

	Repetition factor
Null queries	1.45
Non-null queries	19.57

Figure 2(a) describes the quantity of search traffic covered by only top-k most popular queries. Top 10% most popular queries only cover 30% of null search traffic. But for non-null queries top 10% most popular queries cover 90% of the search traffic. This shows that null search traffic has extremely thin head and long tail. Further more Figure 2(b) describes day over day overlap of null queries. For this all null queries that appeared in a single day were taken. Then overlap was computed over historically occurring queries. It was observed that any two separate days have only 7% null queries in common. When the historical null queries time period is increased to 32 days then also the overlap increases to 28%. As can be seen in the figure that adding more historical data has diminishing returns after some days with respect to amount of queries overlapping.

Null queries have a long tail and do not repeat often over time. Hence it is not practical to apply techniques that rely on query specific models to null searches.

3.3 Generation of Null Searches

Null results are caused because of various reasons. Null queries are much longer than non-null queries. In e-commerce domain as the documents are shorter and far less than web domain, query verbosity is one of the reasons for search being a null search. For example the following query *"aquasource kitchen faucet with sprayer 2 chrome handle"* likely leads to no results on major commerce search engines. The faucet might match but the color chrome may not match leading to a null search.

Another reason for null searches is the temporal volatility of the document space. As the items are being sold they are being removed from the marketplace. Non-null queries may turn into null queries due to this reason. Some null recall queries also transition to non-null recall queries. Queries have seasonal (for example Christmas, Thanksgiving) and buzz [19] (suddenly rising in popularity) properties. Just like queries item corpora (supply) in the e-commerce market place also has seasonal and buzz properties. That is items being sold in holiday season may not be available during summer, Valentine's day based inventory is only available close to valentine's day. The queries (demand) are also seasonal in their peaks. But many queries do not go down to zero in off season. At that time it can lead to null recall searches.

Figure 2(b) shows the magnitude of volatility among null recall queries. To compute the volatility in the inventory-document space we conducted an experiment. Instead of performing search over products currently being sold in the market place we performed the search over inventory that was available even once in previous year. For this we indexed historical inventory from previous year. We randomly sampled 500k queries from the set of current null queries. We ran these queries on the historical cumulative index and found that 22% of queries became non-null recall.

Another reason for zero recall searches is difference in vocabulary between seller and buyers. As we show in [21] both power and novice users on the marketplace encounter null recall situations.

3.4 Effect of Null Searches on Users

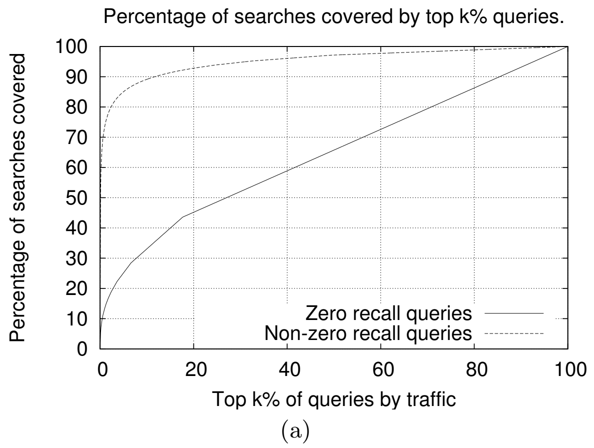
Users tend to react differently to null queries. Some users end their searching as soon as they experience null searches. Others tend to reformulate their query to search for the relevant items. In this section we discuss the effect of null recall on users.

Table 3: For a user segment the table compares degradation of purchase rate for zero recall (null) search trails from non-zero recall (non-null) search trails. Purchase rate for all classes of users is lowered when null recall situations are encountered on their trails

User segment	$\frac{Purchase\ rate\ for\ ZRST}{Purchase\ rate\ for\ NZRST}$
All users	0.63
Power users	0.68
Novices	0.61

We notice that the purchase rate drops when the users experience a null search in their search trail. A user with null search in her search trail has a purchase rate which is 0.63 times the purchase rate of a user who did not experience a null search. Table 3 describes this factor for various user segments.

Figure 3 shows a trail starting with a null search *"heli-*



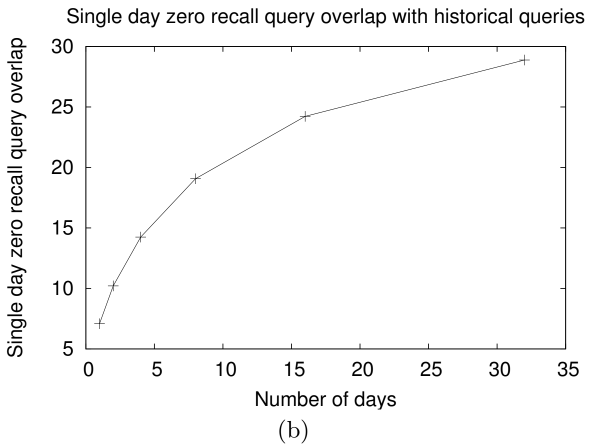


Figure 2: (a)Search traffic coverage of top k% most popular queries. Only 30% of null query traffic is covered by 10% most popular null queries. But 90% of non-null query traffic is generated by 10% most popular queries. (b) Overlap of single day null queries with historical queries.

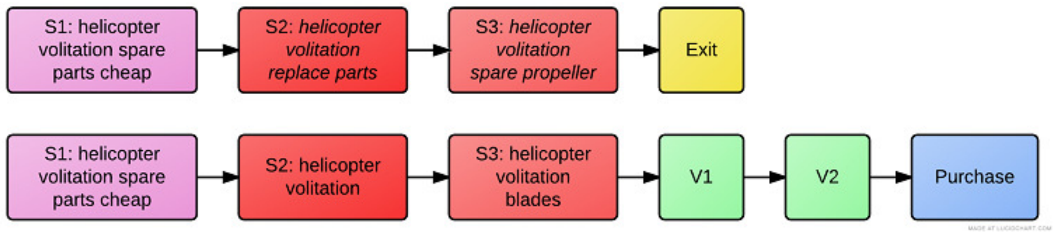

S1: helicopter volitation spare parts cheap
S2: helicopter volitation replace parts
S3: helicopter volitation spare propeller
Exit
S1: helicopter volitation spare parts cheap
S2: helicopter volitation
S3: helicopter volitation blades
V1
V2
Purchase

Figure 3: Figure shows two trails starting with same null search. In the first trail user is searching for volitation brand toy helicopter. User reformulates the query twice but still gets the null result and exits the site. In the second trail user is able to reformulate the query successfully. The V's are product description page view.

copter volitation spare parts cheap". The user is searching for spare parts for *Volitation* brand toy helicopter. The user reformulated the query to *"helicopter volitation replace parts"* ($S2$) which also returned no results. This user did another query ($S3$) which returned zero results again and at this time user gave up.

4. DATASET AND TERMINOLOGY

In this section we define the terminology used in the rest of paper. We also describe the dataset used along with some key data properties.

4.1 Terminology

For search engines built over ephemeral documents, documents are consistently being added and removed. As the item expires search engine removes it from the searchable documents and it is not to be shown to the user. The document corpora on which search engine searches over is also dynamic and users at different time will see different results. Hence the search engine becomes dynamic, volatile and time sensitive.

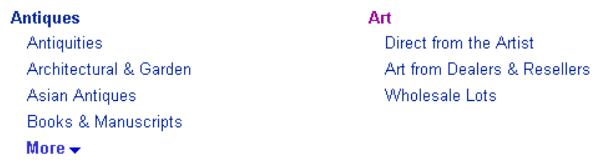
For any time stamp t, we define the set of all documents that may be shown as a response to a search query at that time S_t. These items are considered of value to user at that particular time and hence are shown to user. In other words the recall set of every query will be a subset of S_t. We will also refer to items in S_{now} as *live items.* Different systems expire documents differently. In e-commerce, items that get purchased expire. In social media, stale or old documents are considered to be expired or irrelevant. For example, `http://twitter.com` expire any tweet which is more than few weeks old. Ticketing or invitation management systems expire tickets or invitations after the event has happened.

We also provide a notation to represent the set of all documents which were *live* in recent past. We will denote this set as U_t. Many items in this set are not live any more. Hence the search engine will not search over all of these items. More formally

$$U_t = \cup_{t1} S_{t_1};\ where\ \ now - t \leq t_1 \leq now$$

That is U_t is the set of all items that were live at any point since last t time stamps. For example $U_{6months}$ is set of all items which were live at any point in last 6 months. It is very easy to see that $U_0 = S_{now}$. And $S_{now} \subset U_t\ \forall t;\ t \geq 0$. Metadata space is defined as space over which documents are clustered or labeled. This is engineered into many systems, either provided by the editor or users or inferred by the system. For example, a taxonomy class or tags over documents are a good example of Metadata space. For example, sites like wikipidea provide a rich taxonomical classification of document. Same is true for many other domains like question-answer sites or e-commerce site. In other domains like social media, some tags are user provided with special syntax (# for twitter) while other tags are to be inferred from the documents.

We will refer to taxonomy used in eBay as $C = \{l_1, l_2, \ldots, l_n\}$ where l_i is taxon used in the taxonomy. The taxonomy used in eBay is a laminar family. That is given two taxa l_1 and l_2 either $l_1 \cap l_2 = \phi$ or $l_1 \subseteq l_2$ or $l_2 \subseteq l_1$. For our purposes we will only deal with the leaves of this taxonomy tree. Figure 4 shows a small snapshot of eBay taxonomy. This taxonomy tree is maintained and generated by domain experts and is of high quality.



Antiques
- Antiquities
- Architectural & Garden
- Asian Antiques
- Books & Manuscripts
- More

Art
- Direct from the Artist
- Art from Dealers & Resellers
- Wholesale Lots

Figure 4: shows a snapshot of taxonomy used in eBay. Here the top taxa Antique and Art are expanded into lower taxa for example Direct from the Artist or Antiquities.

Query is the keyword typed in by the user to perform a search. Some of the search engines can take other forms of input along with the query to perform the search. Just to name a few, taxonomical constraints, tag constraints or time constraints could be used. A query for which the search engine does not match any document is referred to as *null query*.

4.2 Dataset

For this paper we used items listed on `http://www.ebay.com`. Items are submitted by sellers with intention to find the right buyers. One of the key methods for item discovery is search by the buyers. As items get sold and out of stock they are removed from the inventory to be searched upon. We use two datasets $U_{6months}$ and $U_{12months}$. Table 4 gives normalized size of these datasets when compared with S_{now}. That is size of all the live items at a given instant. The dataset with previous 6 months items is 7.1 times bigger than the live dataset. The dataset for 12 months is 27.8 times larger than the live dataset.

Table 4: Relative size of $U_{6months}$ and $U_{12months}$ when compared to live items at an instant.

Dataset	Normalized size
S_{now}: *live* items at a given instant	1
$U_{6months}$: items that were *live* at some point in last 6 months	7.1
$U_{12months}$: items that were *live* at some point in last 12 months	27.8

It can be seen from Table 5 that when current zero recall searches are performed on $U_{6months}$ (all items that were live in last 6 months) then 28% of these searches become non-null searches. These queries now match items which have expired. Although these items cannot be shown to the users, they provide usefull metadata information on user's intent. This number increases to 63% when null searches are performed over $U_{12months}$.

Table 5: Percentage of zero recall searches that are not zero recall when done on U_t. That is when zero recall searches are performed on a expired items also then what percentage do not remain zero recall.

Dataset	Percentage of zero recall searches that do not remain zero recall over U_t
$U_{6months}$	28%
$U_{12months}$	63%

We use 4% sample of searches on eBay for simulation of live traffic. This is used for simulation live traffic for the algorithm. The query logs were used for this sampling.

We also tracked power buyers ([21]). We extracted sessions where power buyers performed a null search but then later on recovered and made a purchase in the same session. We are using this dataset as one of the ground truth for algorithm evaluation. This dataset comprises of 100,000 null query and purchased item pair.

5. ALGORITHM

In this section we describe the algorithm for query relaxation while keeping high fidelity. We use taxonomical feedback inferred from the database of expired items ($U_{12months}$). The taxonomical feedback is used as a proxy for user intent with null query. When the query is relaxed the taxonomical feedback help keeps high fidelity.

Table 6: Algorithm for better query relaxation using historical feedback.

Input:	Zero recall query q of term length k. That is q is performed on S_{now} returns an empty set.
1.	Perform q on U_t (for t = 6months or 12months) to match $I_e = \{i_{e1}, i_{e2}, \ldots, i_{en}\}$.
2.	Induce a distribution over C from I_e. That is $Prob[user\ intended\ to\ find\ items\ in\ taxon\ l]$ $= \frac{\lvert\{i \mid i \in I_e \text{ and } i \text{ is associated with taxon } l\}\rvert}{\lvert I_e \rvert}$
3.	Infer top few taxa from the distribution. We use the head of the distribution. $l \mid p[l] > avg_c inC(p[c]) + smoothingconstant$
4.	Generate subset queries of q in order of length.
5.	Run subset query with taxa constraints on S_{now} and return that as output.

Table 6 describes the algorithm formally. The input to the algorithm is a null query q that is a query which returns zero items when searched over S_{now}. We perform this query over U_t (for $t = 6$ months or 12 months). The query matches some of the expired items. Let's call this set of items $I_e = \{i_{e1}, i_{e2}, i_{e3}, \ldots, i_{en}\}$. Each of the item i_{ek} belongs to the taxon in the taxonomy class. We can induce a probability distribution over taxonomy class C from I_e. Probability [User wanted items from taxon l] is defined as fraction of items from taxon l which matched query q in historical database. Some smoothing needs to be done for removing noise and edge cases. This distribution mirrors user's intent for the query. The inferred distribution is then used to infer few key taxa which are most representative of user's intent.

Relaxation of original query that is subset queries are chosen with decreasing length. Firstly subsets which are just 1 term smaller from original query are applied with the taxa constraints. We search these queries with inferred taxa on S_{now}. Relaxation of original query may lead to decreased fidelity but constraining items to be found in relevant taxa help boost the fidelity. These taxa serves another purpose where they remove ambiguity from relaxed query. This is illustrated in the example.

Consider the query "*state fair schnibbles pattern*". This is a user typed in null query. The query matches 14 items in $U_{12months}$. The Table 7 shows the histogram and distribution of taxas for the above mentioned query. The most popular taxa in for the query was *Crafts → Sewing & Fabric → Quilting → Quilt Patterns.*

Table 7: Histogram of category (taxa) of items matched from last 12 months ($U_{12months}$) with the null query.

Category	Freq	Prob
Crafts > Sewing & Fabric > Quilting > Quilt Patterns	8	0.58
Crafts > Sewing & Fabric > Quilting > Quilting Books & Instruction	4	0.28
Crafts > Sewing & Fabric > Quilting > Quilting Kits	1	0.07
Crafts > Sewing & Fabric > Quilting > Quilting Tools & Equipment	1	0.07

There are 15 possible relaxation of the given query. All the 3 length relaxation of the query also do not return any item. Table 8 shows the performance of various 2 length relaxation of the query with and without taxonomy constraint. It can be seen that the most ambiguous and misleading relaxations *"state fair"*, *"state pattern"* and *"fair pattern"* lead to 0, 8 and 7 items respectively instead of 3110, 328 and 468 respectively. This happens as other more popular resolution of relaxation are ignored due to taxa constraints. For example the query *"state fair"* is more popular because of the movie by that name rather than the quilt pattern.

Table 8: Performance of various relaxation with and without taxonomical constraint. Queries are performed with and without inferred taxonomy constraint of *Crafts → Sewing & Fabric → Quilting → Quilt Patterns*. The noisy relaxations *"state fair"*, *"state pattern"* and *"fair pattern"* which has most items matching without constraint. Although with the constraint lead to very few items after right taxon constraint is applied.

Query	Nbr of item matching without taxa constraint	Nbr of item matching with taxa constraint
state fair	3110	0
state schnibbles	0	0
state pattern	328	8
fair schnibbles	0	0
fair pattern	468	7
schnibbles pattern	79	68

The algorithm generates 83 items for the above mentioned example query in the category *Crafts → Sewing & Fabric → Quilting → Quilt Patterns.*

6. IMPLEMENTATION AT WEB SCALE

eBay is a high-volume and dynamic marketplace [10, 19]. The marketplace is vibrant with nearly 100 million buyers and sellers, and with over 10 million items listed for sale on a daily basis. Items are listed in explicitly defined hierarchy of categories and there are over 30,000 nodes in the category tree. Hundreds of millions of searches are done on a daily basis. Inventory changes dynamically, hence the same query might retrieve different products at different times. For e.g., if a user searches for "warren buffett" during the June timeframe then she is most likely looking to bid on the charity auction for lunch with Warren Buffett. However at other times of the year, when the lunch auction is not available the intent of the same query might be to find books about Warren Buffett or the Warren Buffett Monopoly Game. Only a fraction of items are cataloged [22]. Majority of items are ad hoc listings not covered by any predefined product or catalog taxonomy as eBay features a very long tail of item types and not uncommon one-of-a-kind-inventory [11].

Today, users on popular commerce sites expect results to be returned in a fraction of a second. Also for dynamic marketplaces the utility of caching is limited and systems that rely on pre-computed statistics of the index like those described in [8] are not helpful. This made our task challenging. The actual system that we built had to meet user expectations in terms of speed which imposed a strict performance requirement on the query rewriting task. Also, we had to make sure that neither do we rely on stale pre-computed information nor do we put undue load on the search engine so as to be able to keep the system deployable to a large user-base.

An exponential number ($O(2^n)$) of sub-queries can be obtained from a query of length n [16]. If all sub-queries need to be evaluated for goodness, the combinations would be expensive to execute. Median number of terms in null queries in our corpus is 3 [21]. If we were to run all sub-queries against the search index to find the best rewritten relaxed query we would have to run on an average 8 queries for every null query. More number of queries not only mean more load on the search servers, but also increased bandwidth consumption. But by using the algorithm described in section 4, we are able to do reduce to run an average of only 2.8 queries (the exact numbers are dependent on the length of the original query)for every null query, a reduction by 65%.

Details of our system are shown in the Figure 5. The Query Rewrite Engine is deployed as a web service running over an application server. The historical database is an in-memory inverted index of more than 1 Billion products seen on eBay in history. A big portion of time is taken by network latencies while calling search engine and expired database services. The Query Rewrite Engine component is light-weight and a single instance of the server can easily support on the order of 150+ queries per second.

The average time taken to process a request is 63 milliseconds with median of 54 milli-seconds. Most time is spent making network calls to historical item service and the search engine. Retrieved products based on our rewrite engine are available via an XML API at labs.ebay.com. Specifically, for the query "vintage coffee grinder w wheel", the response from our system can be retrieved through the call to the

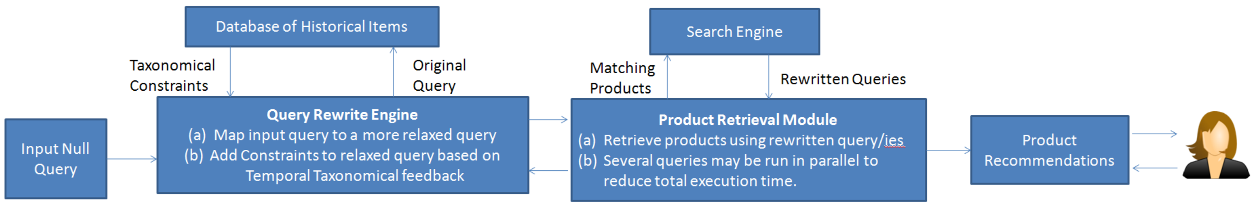

Database of Historical Items
Taxonomical Constraints
Original Query
Search Engine
Matching Products
Rewritten Queries
Input Null Query
Query Rewrite Engine
(a) Map input query to a more relaxed query
(b) Add Constraints to relaxed query based on Temporal Taxonomical feedback
Product Retrieval Module
(a) Retrieve products using rewritten query/ies
(b) Several queries may be run in parallel to reduce total execution time.
Product Recommendations

Figure 5: If a user input query leads to null results through normal processing flow, the query is sent to our Query Rewrite Engine. Relaxations are computed as discussed in section 4. Further, constraints are added to the relaxed rewrites based on historical database. The final rewritten queries with rewrites then are run in parallel against the eBay search engine. If inventory can be retrieved with the rewrites then it is shown to the end user.

algorithm-API[2]. The XML response is similar to the eBay product calls described at eBay developer-API[3].

7. ANALYSIS

In this section we present the performance analysis of our algorithm. We present three different experiments. In section 7.1 we present the result of simulating real user traffic. This section delves into coverage and runtime metrics of the algorithm. In Section 7.2 we present human judgment of the algorithm. And in section 7.3 we compare performance of the algorithm with power users. The Section 7.2, 7.3 delve into precision of the algorithm. This is approximated by studying the relevance of the items surfaced to original query and user intent.

7.1 Simuation of real user traffic

Using the search logs of eBay website we simulated live traffic on the algorithm. We extracted zero recall searches done by a random sample of users and timestamp at which they were performed on eBay.com. We issued these searches to the algorithm to study the performance and load handling of the algorithm. 120,000 searches were issued in the simulation. In this section we present the result of the simulation.

First key performance metric is *coverage* of the algorithm. We define coverage as percentage of null queries for which the algorithm extracts atleast 1 item. We found the *coverage* to be 39% (See Table 9). Section 7.2, 7.3 discuss the quality of the algorithm for these covered searches.

Algorithm was also able to approximate user intent over taxonomy for 63% of searches. For these searches the algorithm found relevant taxa in the taxonomy hierarchy. Although the taxa were found by the algorithm but no items were found which are live at that particular instant. This is also to be expected due to volatility of search space.

Another important metric relevant to the algorithm is the run-time metrics. We studied average and median response time for the algorithm. Table 10 describes these metrics. The average and median response time was found to be 63 and 54 milliseconds respectively. The algorithm rewrites original query in many different ways. For each way it issues a search to the search engine. If this number gets high then we have artificially increased the load on the search engine. For our algorithm we found average number of searches taken by the algorithm per query to be 2.8.

[2] http://labs.ebay.com/erlSRPMerch/LowNullServlet?query=vintage+coffee+grinder+w+wheel

[3] http://developer.ebay.com/DevZone/shopping/docs/CallRef/GetMultipleItems.html

Table 9: Key performance metrics from the simulation. The coverage of the algorithm is defined as percentage of queries for which algorithm fetches items. Coverage is observed to be 39%. User intent is approximated to taxa using historical data. Algorithm returned atleast 1 taxon for 63% of queries.

Performance Metric	Value
Number of searches issues	120,000
Coverage of algorithm: Percentage of searches for which algorithms extract items	39%
Percentage of queries for which atleast 1 taxon is inferred	63%

Table 10: Key runtime metrics from the simulation. Average response time was 63 milliseconds and median response time was 54 milliseconds. Number of different queries to be searched with for a given query was 2.8. And average load on the service during simulation was 12 searches/second.

Runtime Metric	Value
Avg. time to run algorithm per query	63 milliseconds
Median time to run algorithm per query	54 milliseconds
Avg. searches to run algorithm per query	2.8 searches
Avg. query per second	12 searches

As we have observed in section 3 that zero recall queries are verbose. Due to this fact these queries are less likely to be ambiguous. The number of taxa over which the matching items should span is also expected to be low. As expected for 37% of queries the algorithm was not able to infer any taxon from the taxonomy class. For 85% of queries 5 or less taxa were inferred by the algorithm. This to be expected due to verbosity of the null queries.

7.2 Human Judgment

In this section we present the results of human judgments on the algorithm. For this purpose we created a human evaluation tool. Figure 7 shows a snapshot of the evaluation tool. In this tool judges can type in query of their choice or

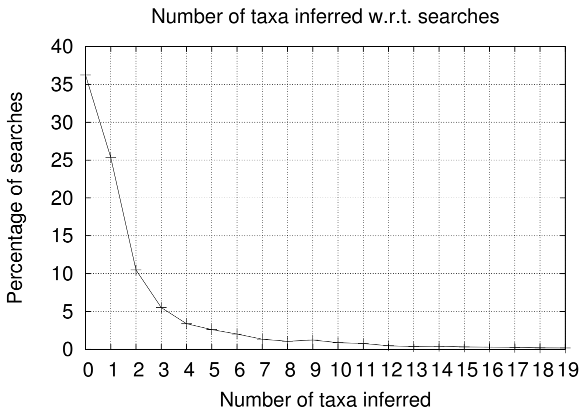
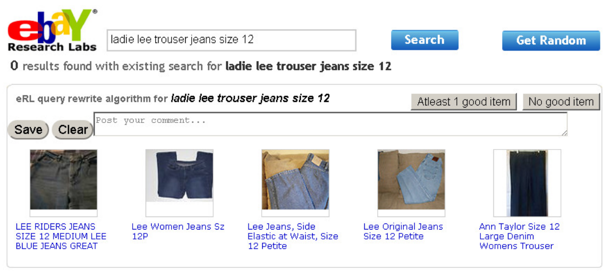


Figure 6: Number of taxa inferred by algorithm using historical data. Here taxa represents user intent as approximated by the algorithm. It can be seen that for 37% of queries algorithm was not able to infer any taxon. For 85% of queries number of taxa inferred is less than or equal to 5.

browse a randomly chosen query which have been done a real human in recent past on the website. Given the query the tool runs the algorithm described in Section 5 and shows the matched items. On these items judges can tag the algorithm to be performing good (*Atleast 1 good item*) or bad (*no good item*). Moreover, for a query if judge choses so she can provide with more verbose textual comments.



ladie lee trouser jeans size 12

0 results found with existing search for **ladie lee trouser jeans size 12**

eRL query rewrite algorithm for ***ladie lee trouser jeans size 12***

LEE RIDERS JEANS SIZE 12 MEDIUM LEE BLUE JEANS GREAT

Lee Women Jeans Sz 12P

Lee Jeans, Side Elastic at Waist, Size 12 Petite

Lee Original Jeans Size 12 Petite

Ann Taylor Size 12 Large Denim Womens Trouser

Figure 7: Human evaluation tool. Judges can see performance on their own query or browse a random query which was performed by some user of eBay. As response to algorithm judges can give feedback of (a) found atleast 1 good item or (b) no good item. On top of that they can also provide verbose text comment.

Seven individuals took part in our human judgment. They together evaluated 200 unique randomly chosen queries. A single query was evaluated by atleast 2 judges. Table 11 describes the result of the evaluation. It can be seen that for 77.2% of queries algorithm extracted atleast one good matching item. For the remaining 22.8% queries the algorithm did not return results that the judges accepted.

7.3 Algorithm compared to power users.

Users who spend more than a threshold amount of money on the site are identified as power users [21].

$$Aggregate\ money\ for\ all\ purchases\ \geq \delta$$

Where δ is a large constant. These users comprise of a very small segment of total user population. Power user tend to spend more time on the site ([21]). For power user purchase rate remained high even if they encountered null searches. We sampled 100,000 sessions of power user from over a month. In these sessions power user encountered a null search and was still able to discover and purchase an item. She did so by changing the queries themselves. As power users are expert in searching and forming queries, we take the purchased item as a proxy to intent of the null search. We use the taxon associated with the purchased item as one of the taxon signifying user's intent. We run our algorithm on these 100,000 null search and tag the algorithm to work if the taxa inferred includes the purchased item taxon. Table 12 describe the result. For 55% of searches our algorithm was able to find the right taxon.

Table 11: Performance of human evaluation of the algorithm. 200 queries were evaluated by seven human judges. Each query was evaluated by atleast 2 human judge. For 77.2% queries algorithm extracted atleast 1 good item whereas for remaining 22.8% queries no good items were extracted.

Algorithm performance	Percentage of queries
Atleast 1 good item is picked up by algorithm	77.2%
No good item is picked up	22.8%

Table 12: For 55% for queries our algorithm matches the exact same taxon for most granular cases. For the mid-level taxon the precision was found to be 74%.

Algorithm performance	Percentage of queries
Taxon at most granular level	55%
Taxon at mid-level	74%

In one of the example session a power buyer searched with the null query *"state faur schnibbles patter"*. Then she ended up buying the item titled *"schnibbles george quilt pattern"* in the category *Crafts* → *Sewing & Fabric* → *Quilting* → *Quilt Patterns*. As described in Table 7 the inferred category is same.

8. CONCLUSION AND FUTURE WORK

E-commerce sites are prone to null queries resulting in customer frustration. In this paper, we described a system for rewriting null e-commerce queries to match relevant products and present them to users. Our contributions are the following:

- An efficient algorithm to rewrite null queries. Our algorithm preserves fidelity of rewritten query with the original query and is efficient to execute at industry scale.
- A novel way to use time-based relevance feedback to improve the fidelity of rewrites. We search the original query against a database of expired items, use

the meta-data of matched expired items, to constraint the rewritten query and get more precise results while matching against a database of active items.

The approach that we described to handle null e-commerce queries can be used for low recall cases as well. As the items retrieved from our algorithm are shown to users as recommendations, the retrieval can be more forgiving and focus on increased recall. Also, temporal feedback which works well for short eBay documents could work well for searching other time-sensitive corpora like micro-blogs. We plan to study this in future with Twitter as an example. We used rewrite based on term reduction and taxonomy addition. As part of future work, we would also like to see if term replacement based on algorithms that can find semantically similar terms can lead to better coverage or accuracy for our recommendation system.

9. REFERENCES

[1] I. Antonellis, H. Garcia-Molina, and C.-C. Chang. Simrank++: query rewriting through link analysis of the clickgraph (poster). In *WWW*, pages 408–421, 2008.

[2] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 491–498, 2008.

[3] M. Bendersky and W. B. Croft. Analysis of long queries in a large scale search log. In *WSCD*, pages 8–14, 2009.

[4] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of SIGIR*, pages 231–238, 2007.

[5] S. Datta and V. Varma. Tossing coins to trim long queries. In *SIGIR*, pages 1255–1256, 2011.

[6] M. Efron and G. Golovchinsky. Estimation methods for ranking recent information. In *Proceedings of SIGIR*, pages 495–504, 2011.

[7] M. Fontoura, V. Josifovski, R. Kumar, C. Olston, A. Tomkins, and S. Vassilvitskii. Relaxation in text search using taxonomies. In *Proc. VLDB Endow.*, volume 1, pages 672–683, 2008.

[8] S. Gollapudi, S. Ieong, A. Ntoulas, and S. Paparizos. Efficient query rewrite for structured web queries. In *Proceedings of CIKM*, pages 2417–2420, 2011.

[9] D. Harman. Relevance feedback revisited. In *Proceedings of SIGIR*, pages 1–10, 1992.

[10] M. A. Hasan, N. Parikh, G. Singh, and N. Sundaresan. Query suggestion for e-commerce sites. In *Proceedings of the WSDM*, 2011.

[11] Y. hen and J. F. Canny. Recommending ephemeral items at web scale. In *Proceedings of SIGIR*, pages 1013–1022, 2011.

[12] R. JJ. Relevance feedback in information retrieval. In *Salton G. (Ed.), The SMART Retrieval System*, pages 313–323, 1971.

[13] R. Jones and D. C. Fain. Query word deletion prediction. In *SIGIR*, pages 435–436, 2003.

[14] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of WWW*, pages 387–396, 2006.

[15] M. Keikha, S. Gerani, and F. Crestani. Temper: A temporal relevance feedback method. In *ECIR*, pages 436–447, 2011.

[16] G. Kumaran and V. R. Carvalho. Reducing long queries using query quality predictors. In *SIGIR*, pages 564–571, 2009.

[17] A. Malekian, C.-C. Chang, R. Kumar, and G. Wang. Optimizing query rewrites for keyword-based advertising. In *Proceedings of the 9th ACM conference on Electronic commerce*, pages 10–19, 2008.

[18] N. Parikh and N. Sundaresan. Inferring semantic query relations from collective user behavior. In *Proceedings of CIKM*, pages 349–358, 2008.

[19] N. Parikh and N. Sundaresan. Buzz-based recommender system. In *Proceedings of WWW*, 2009.

[20] M. S. Pera. Improving library searches using word-correlation factors and folksonomies. In *Master of Science Thesis, Department of Computer Science, Brigham Young University*, 2009.

[21] G. Singh, N. Parikh, and N. Sundaresn. User reaction to zero-recall ecommerce queries. In *SIGIR*, pages 75–84, 2011.

[22] N. Sundaresan. Recommender systems at the long tail. In *RecSys*, pages 1–6, 2011.

[23] K. Toutanova. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP/VLC*, 2000.

[24] X. Xue and W. B. Croft. Modeling subset distributions for verbose queries. In *SIGIR*, pages 1133–1134, 2011.

[25] X. Xue, S. Huston, and W. B. Croft. Improving verbose queries using subset distribution. In *CIKM*, pages 1059–1068, 2010.

[26] L. Zhao and J. Callan. Term necessity prediction. In *CIKM*, pages 259–268, 2010.