Data Services for E-tailers Leveraging Web Search Engine Assets

Tao Cheng, Kaushik Chakrabarti, Surajit Chaudhuri, Vivek Narasayya, Manoj Syamala
Microsoft Research
Redmond, WA
{taocheng, kaushik, surajitc, viveknar, manojsy}@microsoft.com

***Abstract*—Retail is increasingly moving online. There are only a few big e-tailers but there is a long tail of small-sized e-tailers. The big e-tailers are able to collect significant data on user activities at their websites. They use these assets to derive insights about their products and to provide superior experiences for their users. On the other hand, small e-tailers do not possess such user data and hence cannot match the rich user experiences offered by big e-tailers. Our key insight is that web search engines possess significant data on user behaviors that can be used to help smaller e-tailers mine the same signals that big e-tailers derive from their proprietary user data assets. These signals can be exposed as data services in the cloud; e-tailers can leverage them to enable similar user experiences as the big e-tailers. We present three such data services in the paper: entity synonym data service, query-to-entity data service and entity tagging data service. The entity synonym service is an in-production data service that is currently available while the other two are data services currently in development at Microsoft. Our experiments on product datasets show (i) these data services have high quality and (ii) they have significant impact on user experiences on e-tailer websites. To the best of our knowledge, this is the first paper to explore the potential of using search engine data assets for e-tailers.**

I. Introduction

Retail is increasingly moving online[3]. Revenues of online retailers, also known as e-tailers, follow a power-law distribution: there are a few giant-sized e-tailers like Amazon and Buy.com while there is a long tail of small-sized e-tailers. Big e-tailers like Amazon have many users visiting and shopping at their websites. Their activities on websites yield a "treasure trove" of data, e.g., queries they issue to search for products, search results they click on, products they actually buy. Big e-tailers mine such user data to offer superior experience to their users. We describe two illustrative examples of such experiences (We use the terms product and entity interchangeably in this paper.):

• **Recognizing Synonyms**: Users search for products on e-tailer websites using keyword queries, as they do in web search engines. Consider the product search query "canon 650d" on Amazon.com as shown in Figure 1(a). The product "Canon EOS Rebel T4i 18.0 MP CMOS Digital SLR Camera", which is present in Amazon's catalog, is also known as "canon 650d". However, neither the name nor the description of the product has any mention of "canon 650d". Despite that, Amazon is able to correctly return it as the top result because Amazon understands that "canon 650d" is a *synonym* for this specific camera. This is an example of how big e-tailers leverage data assets for user experience.

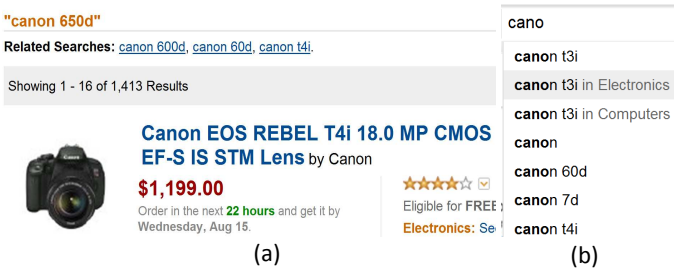

Fig. 1. Features on Amazon: (a) Synonyms (b) Query Auto-completion

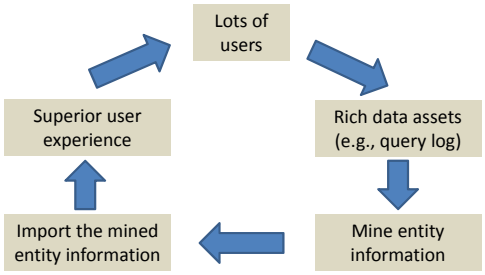

Fig. 2. Virtuous Cycle in Big E-tailers

• **Query Auto-completion**: Like web search engines, Amazon offers query auto-completion experience: it provides a list of query completions with each key stroke of the user (as shown in Figure 1(b)). This significantly reduces the typing effort of its users. Big e-tailers are able to mine the knowledge of most likely auto-completions from its query log.

These superior experiences make the sites more usable and therefore bring in even more users, leading to even richer data assets. This creates a *virtuous cycle* as depicted in Figure 2.

In contrast, small e-tailers cannot create the above virtuous cycle. They have much fewer users, hence they do not have the rich data assets possessed by the big e-tailers. Furthermore, even when such data assets are available, they often lack the mining expertise and the computing infrastructure to leverage them. Hence, they cannot match the rich user experiences offered by big e-tailers. For example, many successful camera e-tailers do not have synonym information: they fail to return "Canon EOS Rebel T4i 18.0 MP CMOS Digital SLR Camera" for the query "canon 650d" in spite of it being present in its catalog (as shown in Figures 3(a) and (b) of a prominent camera e-tailer). Similar comments apply to query auto-completion (as shown in Figure 3(c)).

Main idea and contributions: Over the last decade, search engines have been continuously evolving, attracting more users

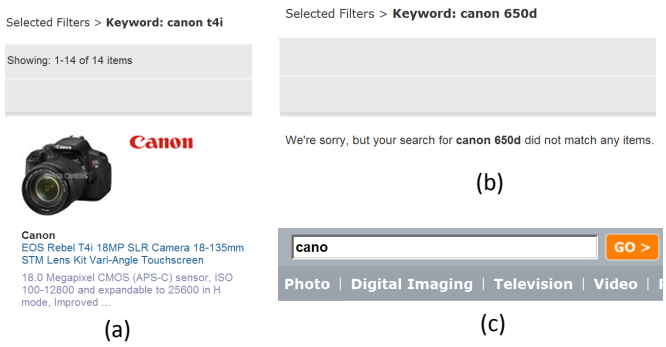

Fig. 3. Lack of Features on a Prominent Camera E-tailer Website: (a) Catalog Hit for Query "canon t4i" (b) No Hits for "canon 650d" Showing Lack of Synonyms (c) No Query Auto-completion

and gathering more data about user activities. So far, the benefit of their rich data assets is reaped only by the search engine themselves. At the same time, most small-sized e-tailers are suffering due to the lack of data assets. Our main idea is to leverage user behavior data assets of web search engines to offer superior user experiences to all e-tailers. Furthermore, we propose that search engines should offer the mined signals as *data services on the cloud*. An e-tailer can subscribe to these data services and create superior experiences for its users. For example, search engine can offer entity synonyms as a data service; an e-tailer can use it to obtain the synonyms for the products in its catalog. E-tailer can then use such information for its product search (e.g., FAST Sharepoint Search [2], Endeca [1]) to improve search quality.

Our contributions can be summarized as follows:

- We present a general API for data services for e-tailers. The API enables e-tailers to subset the signals to the specific products they are interested in (Section II).
- We present three data services for e-tailers: entity synonym data service (Section III), entity tagging data service (Section IV) and query-to-entity data service (Section V). All the data services comply to the general API. For each data service, we (i) define the data service, (ii) describe the algorithms to mine the desired signals for the subset of the products an e-tailer is interested in and (iii) discuss how the e-tailer can consume the output and implementation details. The entity synonym service is an in-production, publicly accessible data service known as the Bing Synonyms API (https://datamarket.azure.com/dataset/bing/synonyms) while the other two are prototype data services accessible only within Microsoft.
- We perform extensive experiments on product datasets. Our experiments show that the mined signals are of high quality for all the three data services (Section VI).

II. API for Data Services for E-tailers

We have developed cloud-based data services that leverage the search engine's data assets to serve rich information useful for products. Figure 4 shows the general API for a data service for e-tailers. It takes one or more entities as input and returns some information about each of those entities. We assume that each input entity is specified by the canonical string used to refer to the entity (say, the one used in its product catalog). We refer to it as the *entity reference string* or simply *entity name*. For example, the entity reference string for the entity *Canon EOS Rebel T4i Camera* is "Canon EOS Rebel T4i 18.0 MP CMOS Digital SLR Camera". We assume a 1:1 correspondence between the entity and entity reference string; we use them interchangeably in this paper. This assumption does not hold for ambiguous entities. While such entities are very common for other domains (e.g., people names, location names), it is relatively uncommon for product names. We focus on unambiguous entities in this paper.

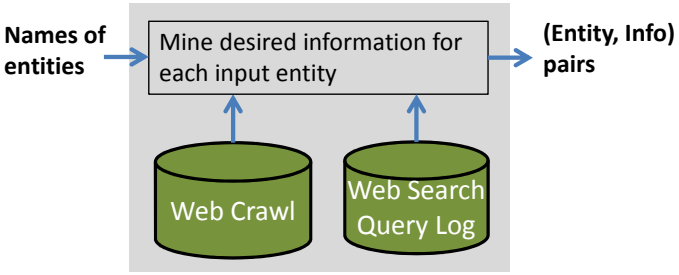

Fig. 4. API for Data Services for E-tailers

Although web search engines have many data assets, the data services described in this paper focus on two data assets: the *query log* which contains the queries issued on the search engine and the links clicked by users for those queries and the *web crawl* which contains a recent snapshot of all documents on the web.

We present three cloud data services, namely entity synonym data service, entity tagging data service and query-to-entity data service, in the next three sections that are consistent with the above API.

III. Entity Synonym Data Service

A. Problem Definition

People often use several alternative strings to refer to the same named entity. For example, the product "Canon EOS T4i Digital SLR Camera" is also referred to as "canon 650d" and "canon kiss x6i". As we saw in Figures 3(a) and (b), product search fails to return the product entity named "Canon EOS T4i Digital SLR Camera" for query "canon 650d" if it does not recognize that they are synonymous. Therefore, the knowledge of synonyms of entities could significantly improve the users' search experience. We develop an entity synonym data service for this purpose. Specifically, the entity synonym data service takes an entity reference string as input and outputs a set of entity synonyms (each a string) for the input entity. Formally, we define the problem as follows:

Definition 3.1: **(Synonym Discovery Problem)** Given entity reference string r_e of entity e, discover the set S_e of synonym strings such that each string $s_e \in S_e$ in the set is a synonym of entity e, i.e., string s_e refers to entity e. □

B. Mining Algorithm

The fact that different people use different ways to search for the same entity is captured by search engines in their query

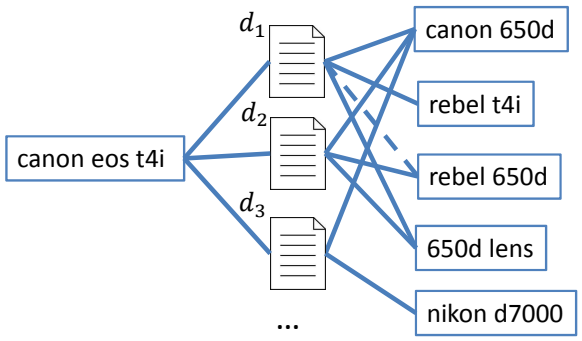

Fig. 5. Synonym Example

click log. Specifically, if two different search queries refer to the same entity, there will likely be significant overlap among the links clicked for the two queries. We thus leverage the query log to identify the various synonyms of a given entity.

We use a two-step data driven approach to mine the synonyms of a given entity.

Step 1: Candidate Generation: Given an entity, we first identify web documents that would be good representative documents of the entity. For example, for a camera, its page on Amazon or dpreview.com is a good representative document. We look for the entity reference string in the query log; we treat the set of web documents clicked when the entity reference string is issued as the query as the *representative documents* for the given query. The queries that have clicked on at least one of these web documents are treated as candidate synonyms. Figure 5 shows the set of candidate synonyms of "canon eos t4i". A solid edge between a query and a document represents a click on the document for that query.

Step 2: Candidate Filtering: Many synonyms are spelling variants of the entity reference string (e.g., misspellings, normalizations); hence, one can use string similarity functions to identify such synonyms among the candidate synonyms. However, a large fraction of synonyms are semantic in nature. For instance, "canon 650d" and "canon eos t4i" are very far away in terms of string distance and therefore cannot be found using string similarity functions, although they represent the same camera. Toward the goal of a general synonym service which finds the broadest class of synonyms with good precision and recall, we need to exploit features leveraging the click information from query click log.

We now use the example in Figure 5 to illustrate how we filter out false synonym candidates, such as "nikon d7000" and "650d lens", and obtain true synonyms such as "canon 650d", "rebel t4i" and "rebel 650d".

We begin by looking at *click similarity*. Specifically, we examine the overlap and the difference between the clicked documents. Intuitively, a true synonym would have a large number of overlapped clicked documents with the corresponding entity reference string. This will help to eliminate candidates such as "nikon d7000" as it will have a small number of overlapped click documents with the entity "canon eos t4i".

Often, many true synonyms of an entity are tail queries, i.e, they are asked by very few users and thus there are few clicked documents. They often would have few or no clicked documents in common with entity. We illustrate this using the "rebel 650d" candidate synonym in Figure 5 as an example. "rebel 650d" is a tail query and has only one clicked document in common (d_2) with the representative documents for the entity. Assuming the threshold is 2, "rebel 650d", in spite of being a true synonym, will not be adjudged a synonym. We refer to it as the query log sparsity problem.

We propose the concept of *pseudo document similarity* to address this sparsity problem. Just like the entity, we represent a synonym candidate by the set of documents it clicked on. In turn, each document is represented by the set of words from all queries for which someone clicked on the document. For instance, the document d_1 in Figure 5 can be represented by words "canon, 650d, rebel, t4i, lens", which is referred to as its corresponding pseudo document. This can be matched with synonym candidates more easily. For instance, candidate "rebel 650d" can be matched with document d_1 via containment checking. As a result, we can establish a link shown by the dotted line in Figure 5 between document d_1 and candidate "rebel 650d". Specifically, the pseudo document similarity from candidate string s_e to entity reference string r_e is the percentage of r_e's pseudo documents which contain string s_e. This leads to much higher recall without sacrificing precision.

One of the key challenges in synonym discovery is to identify and eliminate strongly related yet false synonym candidates such as "650d lens". We notice that such candidates are often of a different entity type with respect to the input entity. In this case, the candidate is of the lens type whereas the input entity is of the camera type. We use *query context similarity* to filter out strongly related candidate synonyms which are of a different entity type of the input entity. For each entity and its candidate, we find longer queries that contain those strings and obtain the tokens that frequently co-occur with them (as prefixes or suffixes); we refer to them as context words. We use the query log to obtain a set of context words for an entity and its candidates by examining tokens which frequently appear with the entity or the candidates in queries. We observe that entities of different types typically have very different context words, whereas entities of the same type have similar context words. For instance, entities of the type camera often come with context words such as "manual", "megapixel", etc., whereas entities of the type lens come with context words such as "filter", "protector", etc. As a result, by checking the similarity between the context words, we can further filter out candidates such as "650d lens" and improve precision.

Finally, it is important to perform two way checking of all similarity functions. This is due to the symmetric nature of synonym relationship, which states that if a is a synonym of b, then b is also a synonym of a. Specifically, we need to check the similarity from entity reference string r_e to synonym candidate s_e, and vice versa. More in depth discussion of this aspect may be found in [8].

C. Consumption by E-tailers

There are two ways entity synonyms can be incorporated into product search for e-tailers. We now discuss these two ways of consumption.

Catalog Enrichment With Synonyms: E-tailers can use the discovered synonyms to augment and enrich its product catalog. For instance, an e-tailer can introduce an additional attribute where the string "canon 650d" can be put to enrich the information for entity "Canon EOS T4i Digital SLR Camera". The enriched catalog can be subsequently indexed and used for capturing more queries. In this example, a user submitting the query "canon 650d" can retrieve entity "Canon EOS T4i Digital SLR Camera" in the search result.

Query Alteration Using Synonyms: Another way of leveraging the generated synonyms is in query alteration. An e-tailer will still use the synonym data service to obtain the synonyms for its entities. Once these entity and synonym pairs are stored, they can be used for query alteration. For instance, given the stored synonym pairs, the product search engine can choose to alter the query "canon 650d" to query "Canon EOS T4i Digital SLR Camera". The altered query can then retrieve the right product. In fact, existing product search solutions like Microsoft FAST Sharepoint Search have API for e-tailers to specify their own synonym pairs offline, which is then used for query alteration in online search.

D. Implementation

We now discuss implementation options for the entity synonym data service. Since the mining algorithm deals with large scale query log and involves sophisticated computation, we leverage the MapReduce framework (specifically COSMOS, a MapReduce framework based on Dryad [13]). Our implementation assumes that a snapshot of the query log already exists in the MapReduce framework. It is straightforward to implement both the candidate generation step and the candidate filtering step of the mining algorithm using MapReduce.

One option is to submit a MapReduce job to generate the set of entity synonyms when an input set of entities is given by performing the mining algorithm described above for each entity in the input set. This option, although an intuitive one, involves running a batch MapReduce job, which is typically time consuming. This means the e-tailer has to wait (typically in the order of hours) for the output. Can we overcome this inconvenience and offer e-tailers much faster turn-around time?

The second option is motivated by the insight that the entity reference strings of many entities are themselves web search queries. If we simply treat each web search query as an entity reference string, we can *offline* generate synonyms for all queries. Although this process is extremely expensive as it involves synonym generation for all queries, in practice it can be done in a scalable MapReduce framework where the computation can be highly distributed. Further, this job only needs to be run periodically (say once every week). Computing synonyms for all queries typically takes one day to finish over two years of query click log. The computation output is a synonym pairs file, or a synonym thesaurus, upon which very efficient lookup can be performed. With this option, we can generate the set of entity synonyms for a given set of entities very efficiently, since it only involves dictionary lookup for each of the input entity. This way, e-tailer can get the output almost instantly. Our publicly available entity synonym data service (https://datamarket.azure.com/dataset/bing/synonyms) adopts this option which eliminates the need to run batch MapReduce job given input.

While a large number of entity reference strings can be directly found in web search queries, there exists many entity references strings which cannot be directly matched. Such entity reference strings are typically long entity reference strings with extraneous tokens. To deal with such input entity reference strings, approximate string matching techniques (e.g., [9], [4]) can be leveraged to match such input strings to queries in the query log.

IV. Entity Tagging Data Service

A. Problem Definition

In Section III, we considered search for products using the name of the product. While search-by-name is very popular, users often search for products based on desired features. Examples in the camera domain are "underwater disposable camera", "rugged camera" and "point and shoot camera". We refer to such queries as *"search-entity-by-feature" (SEF)* queries. A recent study reports that about 42% of all product queries are *SEF* queries [14].[1] [2]

For *SEF* queries, searches over the product catalog often miss relevant results as the feature-describing keywords are often not present in the product information in the catalog [5]. Consider a product catalog containing the name, technical specifications and possibly a short description of each product. For query "rugged camera", *Ricoh G600 Digital Camera* is a relevant product but its name, technical specifications or short description in the catalog might not have the mention of "rugged". Hence, search over the above catalog would fail to return this relevant product. If we can automatically assign descriptive phrases to entities (e.g., assign "rugged", "outdoor" and "water resistant" to *Ricoh G600 Digital Camera*) and augment the product catalog with them, search over the catalog will be able to answer such *SEF* queries more effectively.

Tagging is very popular in web 2.0 sites (e.g., Flickr) where users *manually* annotate items like images, videos and internet bookmarks with phrases to enable effective browsing and search. Those tags are free-form and have no fixed semantics. Since our goal is to identify the tags *automatically*, we focus on a restricted class of tags for entities: *phrases that describe features of the entity* (e.g., "rugged", "outdoor" and "water resistant" for the entity *Ricoh G600 Digital Camera*). We focus on this restricted class for several reasons. First, we

[1] [14] reports that 19.9% of all web search queries are product queries out of which 8.56% are *SEF* queries (referred to as general product queries) while 11.35% are specific product name queries.

[2]The first three paragraphs of this subsection have been adapted from our previous paper [7].

believe such tags would be most helpful in answering *SEF* queries. Second, we show it is feasible to identify such tags automatically with high quality across a wide variety of entity domains. Third, we can systematically evaluate the quality of our entity tagging system for the above class of tags. We refer to such tags as *entity tags* or *etags* in short.

One option is to follow the synonym mining approach and leverage the query click log. For example, one can assign the etag "rugged" to Ricoh G600 Digital Camera if there is significant overlap among the links clicked for the queries "rugged camera" and "ricoh g600 digital camera" (or a shorter synonym of the camera name, say "ricoh g600"). This approach suffers from poor recall. Search engines may not return specific pages about Ricoh G600 Digital Camera in the top few results for the query "rugged camera"; they might return general articles about rugged cameras (e.g., a cnet article titled "Why rugged cameras are not as rugged as you'd think") or pages listing many rugged cameras. These pages are unlikely to be returned as top results for the query "ricoh g600 digital camera"; hence, the two queries are unlikely to have significant overlap of clicks.

Our approach is based on the following insight: *if t is an etag truly associated with the entity e, t will occur in close textual proximity of the name of entity e in a large number of web documents*. For example, the word "rugged" will appear in close proximity of the string "Ricoh G600 Digital Camera" in many web documents like reviews, blogs and expert advice articles. There are several technical challenges. First, how do we obtain the etags for a wide variety of domains? We observe that the etags vary from one domain to another; etags for cameras are quite different from etags for laptops which in turn are very different from etags for shirts. How do we discover etags of various entity domains with little or no manual effort and with high precision and high recall? Second, how do we associate the etags of a domain with entities of that domain with high precision and high recall?

We formally define the problem. Let $\mathcal{D}$ denote a domain of entities. An example domain is that of cameras. We assume that, in text documents, each entity domain is referred to by *one or more alternative strings*. Let $\mathcal{N}_\mathcal{D}$ denote the set of strings used to refer to $\mathcal{D}$; we refer to them as *domain name strings*. For the camera domain, $\mathcal{N}_{camera}$ is simply {"camera"}. For the laptop domain, $\mathcal{N}_{laptop}$ ={"laptop", "notebook", "laptop computer", "notebook computer"}. We require this input from the domain expert. We typically expand $N_\mathcal{D}$ with both singular and plural forms of those alternative names.

Definition 4.1: **(Entity Tagging Problem)** Given the set $\mathcal{N}_\mathcal{D}$ of strings used to denote an entity domain $\mathcal{D}$, a set $\mathcal{W}$ of web documents[3] and one or more entities $\mathcal{E}$ belonging to domain $\mathcal{D}$, find etags associated with each entity $e \in \mathcal{E}$. □

B. Mining Algorithm

We adopt a two-step architecture. In the first step, we discover etags for any entity domain. In the second step, we associate the etags of a domain with entities of that domain. The architecture is shown in Figure 6. We describe the two steps in further detail.

[3]The set of documents can be either from a focused crawl of a specific domain or a general crawl of the web.

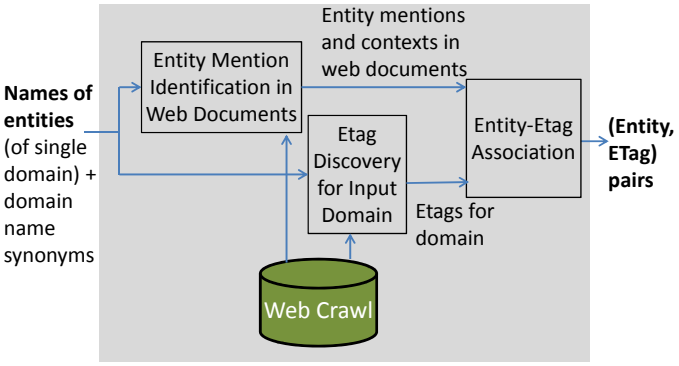

Fig. 6. Core Components of Entity Tagging Data Service

Step 1: Etag Discovery: Given the set $\mathcal{N}_\mathcal{D}$ of alternative strings used to refer to the entity domain $\mathcal{D}$ and the set $\mathcal{W}$ of web documents, the goal is to discover the etags $\mathcal{T}_\mathcal{D}$ for that domain. The main challenge is to achieve this in a domain independent manner while ensuring high precision and high recall.

Step 2: Entity-Etag Association: Given the set $\mathcal{T}_\mathcal{D}$ of etags for a domain (discovered in Step 1) and a set $\mathcal{E}$ of entities belonging to the same domain, the goal is to associate an etag $t \in \mathcal{T}_\mathcal{D}$ to an entity $e \in \mathcal{E}$ if t is truly associated with e, i.e., t is a descriptive phrase for e. Since our association is based on textual proximity between entities and etags, we first need to identify where the entities in $\mathcal{E}$ are mentioned in web documents. We refer to them as *entity mentions*. Subsequently, we can look into the contexts of the entity mentions to find out which entity-tag pairs appear in close proximity of each other and aggregate evidence across all web documents. Hence, this step has two software components:

(a) *Entity Mention Identification in Web Documents*: Given the set $\mathcal{E}$ of entities belonging to domain $\mathcal{D}$ and the set $\mathcal{W}$ of web documents, the goal is to identify the mentions of those entities in those web documents. Document authors often miss (unimportant) tokens, have additional tokens or use a different ordering of tokens compared with the entity reference string in $\mathcal{E}$. For example, one might refer to Ricoh G600 Digital Camera as "Ricoh G600" or "G600 10mp camera from Ricoh". It is critical to identify such approximate mentions of entities. The main challenges are (i) to identify such approximate mentions with high precision and recall and (ii) to scale to billions of web documents.

(b) *Entity-Etag Association Using Contexts*: Given the set $\mathcal{T}_\mathcal{D}$ of etags for a domain and the mentions of entities in $\mathcal{E}$ and their contexts in web documents, the goal of this component is to associate an etag $t \in \mathcal{T}_\mathcal{D}$ to an entity $e \in \mathcal{E}$ if t is a valid etag for e. The challenge is to develop robust techniques that performs this association with high precision and high recall.

Note that another alternative is to first identify occurrences of the etags and then look for entity mentions in the context of those occurrences. For many domains, the etags can be commonly occurring terms (e.g., "cheap", "light", "small",

"large"). In such cases, occurrence of entities is much more selective than occurrence of etags. It is more efficient to evaluate the more selective condition first. Hence we adopt the former alternative.

We discuss the three components of etag discovery, entity mention identification and entity-etag association in further detail.

Etag Discovery

We use precise lexical patterns that identify etags for any entity domain. Consider the domain of cameras. Suppose $\mathcal{N}_{camera}$ is {"camera"}. We look for patterns like "is a t camera", "t cameras such as" and "and other t cameras" in web documents. These patterns are inspired by Hearst patterns but differ from traditional Hearst patterns which identify class-instance relationships. If there are sufficient number of web documents containing the pattern, we identify t as an etag of the domain $\mathcal{D}$. This will discover tags like "rugged", "ultracompact" and "prosumer" for the camera domain. We formally define an etag for a domain $\mathcal{D}$.

Definition 4.2: **(Etag for a Domain)** A phrase t is an etag for the domain $\mathcal{D}$ iff one of the following patterns occur δ or more times in $\mathcal{W}$:

(a) "is a t $n_{\mathcal{D}}$" (b) "is an t $n_{\mathcal{D}}$" (c) "t $n_{\mathcal{D}}$ such as" (d) "and other t $n_{\mathcal{D}}$" (e) "or other t $n_{\mathcal{D}}$" (f) "t $n_{\mathcal{D}}$ including" (g) "t $n_{\mathcal{D}}$ especially"

where $n_{\mathcal{D}} \in N_{\mathcal{D}}$, "t $n_{\mathcal{D}}$" is a noun phrase and δ is an application specified threshold ($\delta \geq 1$). □

The parameter δ is used to eliminate noise. The entity tagging service currently uses $\delta = 5$. Note that the only manual effort for each domain is to provide the set of alternative strings used to refer to the domain.

Entity Mention Identification in Web Documents

To detect approximate mentions of entities, we adopt the definition of mentions proposed in [9]: a subsequence of tokens s in a document is a mention of an entity $e \in \mathcal{E}$ iff the similarity $sim(s, e)$ exceeds a certain threshold θ. Both s and e can be viewed as a set of tokens; then, the similarity can be defined using Jaccard similarity. We only consider subsequences up to length α.

We follow a generate-verify scheme. For each document, we first generate candidates, i.e., identify subsequences of documents which can potentially be a mention. To generate candidates, we build an index on the entities as proposed in [4]; we construct prefix filters on the entities and build an inverted index on those prefix filters. We can then identify candidate subsequences and the corresponding matching entities using this index; we refer to this index as the "approximate match index". Subsequently, we retrieve the corresponding matching entities from the entity set $\mathcal{E}$ and verify whether they are indeed mentions by checking the similarity condition $sim(s, e) > \theta$.

An alternative is to use the search engine API instead of identifying entity mentions in web documents. We can issue the entity reference string as the query and look for etags in the snippets, titles and URLs returned by search engine. This alternative has several limitations. First, this does not scale: for a large set of entities, this approach could take a very long time. Retrieving the body of the documents is an order of magnitude slower and is hence totally infeasible. Second, the amount of context information is small: snippets are small and most APIs have a bound on the number of hits returned. This adversely affects the quality of entity-tag association. Third, the semantics of entity mentions in this case are not as rigorous as the semantics defined in this paper. This might lead to poor quality of association.

Entity-Etag Association

Our goal is to associate an etag to an entity iff (i) they occur in close textual proximity of each other (e.g., within a window) and (ii) they co-occur more than expected. We first describe our co-occurrence analysis technique. Then we strengthen it by leveraging proximity information and documents that are exclusively about an entity.

Preliminaries: Co-occurrence Frequency Analysis: Recall that $\mathcal{W}$ represent the collection of web documents. Let $N_{\mathcal{W}}$ denote the number of documents in the collection $\mathcal{W}$. We denote the frequency of an entity e and an etag t in the web document collection $\mathcal{W}$ as follows:

- $Freq(e)$: the number of documents in $\mathcal{W}$ where mention of entity e is identified.
- $Freq(t)$: the number of documents in $\mathcal{W}$ where etag t is identified.

The *observed co-occurrence frequency* is the actual number of documents in collection $\mathcal{W}$ which contain both etag t and mention of entity e, defined as: (s refers to a mention of entity e in document d): $O(e,t) = \sum_{d \in \mathcal{W}} 1$, subject to $t \subset d, s \subset d$.

The *expected co-occurrence frequency* is the expected number of documents in collection $\mathcal{W}$ which contain both etag t and mention s of entity e, under the assumption that etag t appears independently of entity e. It is defined as: $E(e,t) = \frac{Freq(e)}{N_{\mathcal{W}}} \frac{Freq(t)}{N_{\mathcal{W}}} N_{\mathcal{W}} = \frac{Freq(e)Freq(t)}{N_{\mathcal{W}}}$.

To compare the observed co-occurrence frequency with the expected co-occurrence frequency, we perform standard statistical testing using G-test of goodness-of-fit. The outcome of the test identifies the significance of difference in helping decide where etag t and entity e are positively correlated. Specifically, $G-test = 2Oln(\frac{O}{E}) + 2(N_{\mathcal{W}} - O)ln(\frac{N_{\mathcal{W}}-O}{N_{\mathcal{W}}-E}) \approx 2Oln(\frac{O}{E})$. This describes G-test, where we use O and E as shorthand for $O(e,t)$ and $E(e,t)$. Since both $O(e,t)$ and $E(e,t)$ is insignificant when compared to $N_{\mathcal{W}}$ under the assumption that an entity and an etag will typically co-occur in a small faction of documents of the entire corpus $\mathcal{W}$, the second component in the G-test can be neglected as an approximation. Different significance levels can be used as thresholds to the result of G-test for selecting the associated set of etags for an entity.

Leveraging Entity-Etag Proximity Information: We can improve the co-occurrence analysis by incorporating the proximity information between etag and mentions of entity. The intuition here is that the closer entity e appears in proximity to etag t in a document, it is more likely that they are associated. We measure the proximity between an etag and a mention of entity in a document as the number of tokens between them.

Therefore, rather than counting each entity-etag co-

occurrence in a document as 1, the co-occurrence is weighted by the proximity between the entity mention and the etag in the document. As a consequence, co-occurrences that appear in closer proximity gain more weight. This idea has been shown to be effective in supporting keyword searching of entities [10], [11], [5]. In our implementation, we adopt the proximity weighting scheme used in [11].

Leveraging Documents Exclusively About an Entity: A common problem in performing proximity based co-occurrence analysis is the noise introduced by other entities occurring near the target entity. Etags of other entities can sometimes get incorrectly associated with the target entity. This is especially true for listing pages where many entities in the same domain are mentioned in a list, e.g., an Ebay listing page containing a list of cameras, each with a short description.

One possible solution is to ignore the co-occurrence of an entity mention and an etag if there are other entities of the same domain mentioned between them. This requires a comprehensive input list of entities for a domain. While it may be feasible for small scale, closed domains, identifying such a comprehensive list of entities for general domains is very challenging.

We take an alternate approach. In order to identify etags for entities robustly, we identify web documents that are exclusively about an entity. This could be a review page of a specific product, or a specification page of a product. The insight here is that etags mentioned in such documents are more likely to be valid etags for the specific entity. We identify such documents by performing entity mention identification over web documents's url and title, leveraging the entity mention identification technique described above. We peform proper tokenization over urls to facilitate entity mention identification over them. Subsequently, we give more weights to co-occurrences identified on such documents. Note that many entities do not have such exclusive pages. Therefore, we adopt this exclusive document identification technique as a boost to co-occurrence analysis of all matched co-occurrences, rather than a replacement.

Other refinements we apply in enhancing association accuracy include: eliminating (i) the effect of mirroring documents by counting duplicated documents only once in aggregation; (ii) the effect of one specific tag or entity appearing multiple times in a document, by only allowing the same (entity, etag) pair output at most once by one document; (iii) the effect of copy/pasting of segments of text, by building context signature of etags (for instance, using the five left and right words as signature). Etags with exact same contexts are likely caused due to copy/pasting, and should only be counted once.

C. Consumption by E-tailers

The etags can be used to augment the product catalog. For instance, an e-tailer can introduce an additional "tags" attribute where the etags "rugged", "outdoor" and "water resistant" can be put to enrich the information for entity "Ricoh G600 Digital Camera". The enriched catalog can be subsequently indexed and used for capturing more queries. In this example, a user submitting the query "rugged camera" can retrieve entity "Ricoh G600 Digital Camera" in the search result.

D. Implementation

Both steps require processing of billions of web documents; we leverage the MapReduce framework for scalable processing of web documents. We describe the implementation for one domain of entities; it is straightforward to extend it to multiple domains. Our implementation assumes that a snapshot of the web already exists in the MapReduce platform. Search engine companies like Google and Microsoft have such snapshots; such snapshots are also available publicly (e.g., the ClueWeb09 dataset).

We first consider the etag discovery. The lexical patterns can be detected and the etags can be identified from each document independently; hence, this task can be executed in parallel using MapReduce. We need a single MapReduce job for this task. The map step looks for the lexical patterns in each document and, for each pattern found, outputs the key-value pair $\langle t, 1\rangle$ where t denotes the etag in the pattern (1 is the occurrence count). We look for lexical patterns by first detecting exact occurrences of strings in $N_{\mathcal{D}}$ in the web document (using an exact multi-string matching algorithm like Aho-Corasick algorithm); we then look for the remaining portion of the lexical pattern in the left/right of those occurrences. The reduce step simply sums up all the input values to obtain the number of occurrences of each etag. We filter out etags that occur less than δ times in web documents.

We now consider entity mention identification. We first build the approximate match index on the set $\mathcal{E}$ of entities using a MapReduce job. The index as well as the set $\mathcal{E}$ of entities is then broadcast to all the nodes. Since the mention identification can now be done independently for each document, we perform it in parallel using another MapReduce job. The Map step runs a sliding window over each document (of size α) and identifies approximate mentions using the generate-verify scheme described above. The reduce step simply concatenates all the intermediate outputs.

We perform entity-etag association as part of the entity mention identification job. After we identify an entity mention, we can check the identified entity mention's surrounding context to look for etags. We aggregate the (entity, etag) pairs across different documents using a reducer for the co-occurrence frequency analysis.

V. Query-to-Entity Data Service

A. Problem Definition

E-tailing begins with a catalog of product entities. Initially, the query log is either missing, or is very limited. As a result, many functionalities relying on query log cannot be enabled. One such prominent functionality is the ability to support query auto-completion. Query auto-completion interactively shows the legitimate and popular query completions at every key stroke of the user. This is a very popular feature adopted universally by all search applications and product search portals of e-tailers are no exception. To enable this functionality,

the e-tailer needs query log containing queries related to its products and their popularity (typically captured by the query frequency, i.e., how often users issue the query). Small e-tailers often do not have a query log or have a very limited query log. Hence, they cannot enable query auto-completion.

As we have discussed, most e-tailers do not natively have rich query logs. In this section, we discuss how despite this shortcoming, we can provide e-tailers with a set of queries "related" to the products in its catalog, with a frequency of each query reflecting its popularity in web search engines. For the purpose of this section, we say that a query is related to a product if the product search engine returns the product in its top results for that query. Such a "related query" set can help e-tailers enable query auto-completion. Thus, the query-to-entity data service takes the entity reference string of an entity as input and outputs a set of related queries (each a string) along with frequency for the input entity. Formally, we define the data service as follows:

Definition 5.1: **(Query-to-Entity Discovery Problem)** Given entity reference string r_e of entity e, output a set of related queries and their respective query frequency for the entity. Each related query q_e in the set is either a subset synonym or hypernym of the entity reference string r_e. □

Here we restrict the related queries to subset synonym or hypernym only. This means any output query q_e has to be a subset of the reference string r_e and is either a synonym or a hypernym of the entity e. This guarantees that any query generated will result in a hit on the entity. This is crucial for enabling query auto-completion since any auto-completed query must have at least one entity hit in its result.

B. Mining Algorithm

We employ a two-step approach on the query click log to find such queries.

Step 1: Candidate Generation: Given the reference string of an entity, e.g., "Canon EOS T4i Digital SLR Camera", we first subset to the set of queries whose tokens are subsumed by the tokens in the input entity reference string. The subsumption condition guarantees that each query will retrieve at least one result entity, since the reference string (name) of an entity is always indexed and is therefore searchable. Some example queries subsumed by the reference string for the above entity are: q_1:"canon camera", q_2:"eos t4i camera", q_3:"digital", etc. While q_1 and q_2 are highly related with the entity, q_2 is very vague by itself and is not very selective in identifying the entity. To reach such set of candidate queries, we leverage search engine APIs to first retrieve a set of web documents relevant with the entity by issuing the entity reference string as query. Then we treat all the other queries which clicked on these web documents and are subset of the entity reference string as candidates.

Step 2: Candidate Filtering: The next step is to ensure that we output only highly related queries that are among the candidate queries. Our mining algorithm focuses on two types of such highly related queries: synonyms or hypernym of the entity. Both types are popular in search queries, as synonym refers to the name of an entity whereas hypernym refers to the category of an entity.

In this step, we leverage the query click log. Specifically, we can check the intersection between the web documents clicked in response to a search query "Canon EOS T4i Digital SLR Camera" and candidate queries (e.g., q_1, q_2, q_3), and eliminate candidates (e.g., q_3) that have very small intersection of web documents. This small intersection indicates that the candidate query is not highly related with the entity, since most of its clicks land on other web documents not about this entity. On the other hand, when there is large intersection with these web documents, it often indicates the query is a subset synonym of the entity (e.g., "canon eos t4i" for "Canon EOS T4i Digital SLR Camera"), and when there is moderate intersection with these web documents, it often indicates the query is a hypernym of the entity (e.g., "canon digital camera" for "Canon EOS T4i Digital SLR Camera"). A threshold is set on the extent of this intersection in order to filter out non-related queries, and return related queries which are either subset synonyms or hypernyms. The query frequency from the search query log is used as a good proxy for the popularity of the query. This frequency is typically used to rank the set of auto-completed queries so that popular queries appear as top completions.

It is worth noticing that the related query generation algorithm is different from that of synonym discovery discussed in Section III. In fact, the subset synonym mining algorithm presented here will not ensure that the two way checking of pseudo document similarity presented in Section III will succeed. The focus for query-to-entity mapping is on entities with very long reference strings. Such reference strings typically cannot be matched to any query in the log and therefore do not have their respective pseudo documents. The insight here is that the constraint on subsets makes mining much easier. As a result, simple intersection based measures already yield highly effective results.

C. Consumption by E-tailers

There are two ways the query-to-entity data service can be consumed by an e-tailer. We now discuss these two options.

Offline Query Log Feed: The query-to-entity data service can be used to obtain a query log given a product catalog as input. Such a query log can be ingested into a product search engine that supports auto-completion. Many existing product search solutions support this functionality. For instance, Microsoft FAST Sharepoint search has an API which takes queries and their frequencies as input. Once ingested, it can provide query auto-completion based on the generated query log.

Online Query Auto-completion: Another option is to directly offer an online query auto-completion service. This differs from providing an offline query log feed to e-tailers. This alleviates e-tailers from hosting their own query auto-completion. On the other hand, this requires the online query auto-completion service to be of very low response time to offer completions (under 100ms - see below for additional discussions on implementation). Once the online query auto-completion service is ready, the e-tailer who subscribed to the

Domain	Entity	Associated Synonyms
Camera	Canon 600D	canon t3i, cannon 600d, canon eos t3i, ...
Camcorder	Sony Dcr Sr87	sony handycam dcr sr87, dcrsr87, ...
Fantasy Movie	Harry Potter and the Half Blood Prince	harry potter 6, half blood prince, hp6, ...
Software	Windows 8	win8, microsoft windows 8, windows 7 successor, ...

TABLE I
EXAMPLE SYNONYMS ASSOCIATED TO ENTITIES

service will only need to make minimal changes to their search box script, so that the search box talks to the service at every key stroke for getting query completions.

D. Implementation

The related query information for products of a given e-tailer can be mined offline (as explained above) and can be directly returned to the e-tailer, who can leverage the log to power its product search auto-completion. However, for the rest of the implementation section we focus on the online query auto-completion scenario discussed above. With this online service option, e-tailers only need to slightly modify their search box script to directly talk to the data service for getting query completions. While this option alleviates the e-tailers to power their own auto-completion, this mode demands our query-to-entity data service is able to ensure extremely fast response to satisfy the need of online query auto-completion. Toward this goal, we implemented a memory based Trie leveraging the Windows Azure infrastructure. First, the query log for a given e-tailer has to be computed offline based on the input set of entity reference strings given by the e-tailer. Next, for a given e-tailer, a Windows Azure instance loads the custom generated query log (from the first step) into the Trie, and then it is able to serve online the top k completions for incoming queries ranked by their frequency in web search query log given a prefix input. A separate Windows Azure instance is needed for each output query log, since the log is specific to an e-tailer. Our results show that by building an efficient Trie over 100 million queries, we can output query completions under 100ms for 99.9% of the test cases.

VI. EXPERIMENTAL EVALUATION

We present an experimental evaluation of the mining algorithms for the three data services. The main goal of the study is to evaluate the quality of the mined signals in terms of precision and recall. Since the focus is on e-tailers, all the entities considered are products from real-life product catalogs. The exact set of domains of products considered is different for the different data services (there are common domains as well); we describe them in the individual subsections.

To evaluate the quality of a service for a particular entity domain, we provide a set of entities of that domain as input to the service, obtain the output and manually judge the output pairs. We report two measures, *precision* and *recall*. Precision is the fraction of output pairs that are judged correct. Recall is difficult to compute since we do not know the set of *all* valid output pairs for an entity (i.e., set of all synonyms/tags/related queries of the entity). Hence, we use the average number of outputs pairs per entity (e.g., average number of synonyms/tags/related queries) as recall.

A. Quality of Entity Synonyms

Setup: To generate entity synonyms, we leverage query click log from Bing from 2010 to 2012[4]. We conduct our evaluation over 5 different product domains from the Bing shopping catalog. These domains cover popular categories such as cameras, as well as tail categories such as car speakers. To manually judge an output pair (e, s), we ask the judge the following question: is s a valid synonym of the entity e?

Results for Quality of Synonyms: To provide a feeling for the output, Table I shows a few example entities and the synonyms discovered for them by our service.

Table II reports the average number of synonyms discovered per entity for these 5 domains, as well as the precision for these domains. Our data service discovers 2-4 synonyms per entity with a precision of around 94%. Notice here we optimize for very high precision, which is well above 90%. We can discover significantly more synonyms per entity if we sightly lower the precision to around 85%-90%.

Product Category	#synonyms/entity	Precision
Camera	3.4	94%
Camcorder	3.3	96%
Car Speaker	2.1	92%
CD Player	2.0	99%
Speaker	2.2	90%

TABLE II
QUALITY OF ENTITY SYNONYMS FOR 5 PRODUCT CATEGORIES.

B. Quality of Entity Tagging

Setup: Our experimental study uses a snapshot of the web corpus of high static rank documents, consisting of roughly 1.4 billion documents with total corpus size at 35.2T. We conduct empirical evaluation on two real product domains, camera domain (with 3,557 cameras) and shirt domain (with 22,182 shirts). We obtained both sets of products from Bing shopping.

We chose these two domains because they are important ones (e.g., frequently queried) and have widely different characteristics. First, unlike the camera domain where entities typically have a unique model name, such as *G600*, entities in the shirt domain are more difficult to distinguish from one another. As a result, approximate entity mention identification could potentially lead to more errors for the shirt domain. Second, entities in the camera domain are relatively well represented on the web, as they are often available on their specification pages

[4]Note that due to proprietary and privacy concerns we cannot share all the details of the query click log.

Domain	Entity	Associated Etags
Camera	Ricoh G600 Digital Camera	waterproof, outdoor, rugged, 10mp, ...
Camera	Go Photo Easy Pix 30 Digital Camera	pink, blue, ultra compact, mini, ...
Camera	Sony Cyber-shot DSC-W220 Point and Shoot Camera	image stabilized, digital zoom, compact, ...
Shirt	Ralph Lauren Childrenswear Striped Oxford Shirt	polo, soft, ...
Shirt	Dogwood Boys Striped Short Sleeve Polo	light blue, little boys, cotton, ...
Shirt	Under Armour Heatgear Short Sleeve Tee Girls	base layer, moisture wicking, casual, lightweight, ...

TABLE III
EXAMPLE ETAGS ASSOCIATED TO ENTITIES

and review pages. Entities in the shirt domain, on the other hand, typically are not as well represented on the web, with most of their appearances in listing pages (e.g., a listing page of all Ralph Lauren shirts). These different characteristics of the two domains give us the opportunity to study the sensitivity of our techniques to different characteristics.

Domain	Total Number of Etags Discovered
Camera	1,166
Shirt	935

TABLE IV
STATISTICS OF ETAGS DISCOVERED

Results for Quality of E-tags The domain name strings used for the camera domain are: "camera", "digital camera", and for the shirt domain are: "shirt", "t shirt". Table IV shows the number of etags discovered for the two domains. On manual evaluation, we found the set of discovered etags to be highly accurate, at around 99% precision.

Results for Quality of Entity Mention Identification: We compare the approximate entity mention identification based on Jaccard containment against exact entity mention identification in terms of precision-recall tradeoff, i.e., the amount we lose in entity mention identification accuracy versus the amount we gain in the number of entity mentions identified. We use the Jaccard containment threshold θ at 0.9 and sliding window size of 30.

Camera	Estimated Precision	Median # of Mentions
Exact Match	100%	36
Approx Match	98%	221
Shirt	**Estimated Precision**	**Median # of Mentions**
Exact Match	100%	7
Approx Match	96%	16

TABLE VI
ENTITY MENTION IDENTIFICATION RESULTS

Table VI reports the comparison result for the two domains. We use the median number of entity mentioned identified, to eliminate the effect of very popular entities which have an extremely high number of mentions on the web. We estimate the precision by manually judging a random sample of entity mentions of 50 camera entities and 50 shirt entities. Compared to exact matching, approximate matching identifies significantly more entity mentions at the cost of a small loss in precision. The higher number of identified entity mentions is critical for the subsequent association analysis, which relies on large scale aggregation. This underscores the importance of approximate matching based on Jaccard containment.

Results for Quality of Final Output: We present our evaluation of the *final output* of our system (i.e., etags associated with the entities) for the two domains. For each domain, we randomly select 50 entities among the entities that have more than 20 mentions identified on our web corpus. The threshold 20 is empirically chosen as the basic limit to perform robust statistical analysis over the entities. This gives us 2,811 qualifying camera entities and 10,103 qualifying shirt entities respectively. Table III shows a few example entities and the etags associated to them by the service.

To manually judge an etag t assigned to an entity e under domain $\mathcal{D}$ with a domain name string $n_{\mathcal{D}} \in N_{\mathcal{D}}$, we ask the judge the following question: "is entity e a t $n_{\mathcal{D}}$?". For instance, for entity *Ricoh G600 Digital Camera* and etag "rugged" in the camera domain, we ask the following question: "is *Ricoh G600 Digital Camera* a *rugged* camera?" Note that this evaluation stands for the end-to-end accuracy of the overall system.

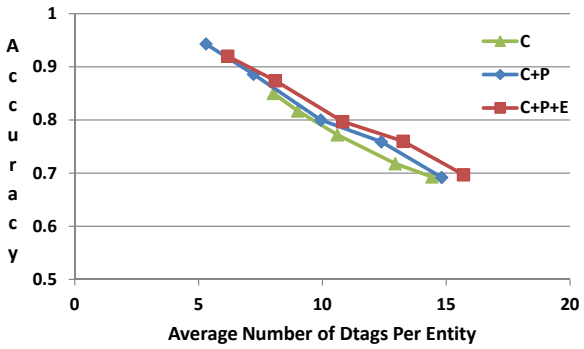

Fig. 7. Entity-Etag Association Results (Camera)

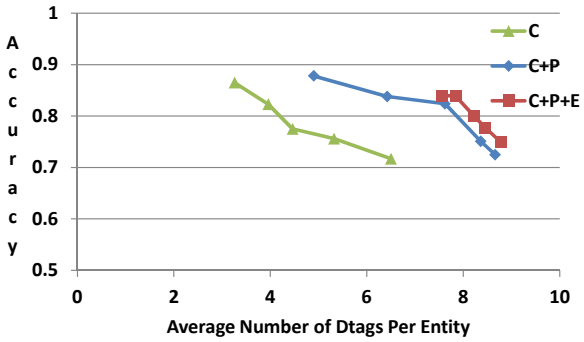

Fig. 8. Entity-Etag Association Results (Shirt)

To evaluate various association techniques described in Section IV, we compare the following methods in terms of precision and recall (the average number of tags associated per entity):

Domain	Entity	Associated Queries
Camera	Canon EOS Rebel T2i 18.0 Megapixel Digital Slr Camera	canon camera, canon eos rebel, canon t2i, ...
Camera	Fujifilm Finepix T350 14.0 Megapixel Digital Camera	fujifilm digital camera, fujifilm camera, ...
Camcorder	Cobra Digital Dvc3100 Digital Camcorder	cobra camcorder, cobra dvc3100, ...
Camcorder	Bushnell Trophy Cam Bundle HD Flash Memory Camcorder	bushnell cam, trophy cam bushnell, ...

TABLE V
EXAMPLE QUERIES ASSOCIATED TO ENTITIES

- **C**: Association by co-occurrence frequency analysis only;
- **C+P**: Association by co-occurrence frequency analysis with proximity information;
- **C+P+E**: Association by co-occurrence frequency analysis with proximity information and entity-exclusive document optimization.

Figure 7 and 8 show the precision and recall for the camera and shirt domains respectively. Co-occurrence frequency analysis alone gives good results; leveraging proximity and leveraging entity-exclusive documents further improve the quality. Overall, the combination of co-occurrence frequency analysis with proximity information and entity-exclusive documents (**C+P+E**) shows the best overall performance. This leads to the assignment of on average ~10 etags per entity with ~85% precision for the camera domain and ~8 etags per entity with ~80% precision for the shirt domain. The differences in precision and recall between the two domains are due to the different characteristics discussed in the beginning of this subsection. The shirt domain presents more challenges in entity mention identification, as well as its lack of web information for robust aggregation.

On analyzing the errors, we found three main sources of errors: (i) etags of some other entities that appear in close proximity to the target entity get incorrectly assigned to the target entity. This is especially true for listing pages where a list of entities are mentioned; (ii) incorrect entity mention identification leads to wrong associations, since the etags associated could be actually about a different entity; (iii) ambiguous tags: tags that are legitimate by themselves, but have different meanings under different contexts. For example in camera domain, "manual", "tracking" are all legitimate etags, since they describe subclasses of camera: "manual camera" and "tracking camera" respectively. However, their appearances on the web often refer to other meanings. For instance, "manual" is often mentioned under the meaning of handbook.

C. Quality and Performance of Query-To-Entity Service

Setup: To generate queries for entities, we leverage query click log from Bing from 2010 to 2012 (similar to the synonym service). We conduct our evaluation over the camera domain and the camcorder domain from the Bing shopping catalog. To manually judge an output pair (e, q), we ask the judge the following question: is q a subset synonym or hypernym of the entity e?

Results for Quality of Query-to-Entity Mapping: Table V shows a few example entities and the queries associated with them by our service.

Table VII reports the number of queries discovered per entity for these two domains, as well as the precision for these domains. Our data service discovers a large number of related queries per entity with very high precision.

Product Category	#queries/entity	Precision
Camera	4.7	99%
Camcorder	5.5	100%

TABLE VII
QUALITY OF QUERY-TO-ENTITY MAPPING FOR TWO PRODUCT CATEGORIES

Auto-completion Performance: We evaluate the latency of online autocompletion based on the queries discovered by the entities. To support online autocompletion, we implemented an in-memory Trie with Windows Azure as the backend infrastructure. Our results show that we can autocomplete 99.9% of the queries under 100ms for a query log consisting of over 100 million distinct queries. This is a "worst case scenario" since most e-tailers' query logs have far less queries.

VII. RELATED WORK

The main idea of the paper is to make the connection between search engine data assets and mining information for e-tailer's products. Given online retail is a $200 billion business [3], this has a tremendous business opportunity. To the best of our knowledge, we are not aware of any other work that has demonstrated the use of web search engine data assets to directly benefit product search functionality for e-tailers. While we believe that the proposed functionality is novel, past works on data mining used for product search as well as for web search are relevant. Naturally, such past work can be divided into two categories: (i) technologies developed by e-tailers to mine their own data assets to obtain information about their own products and (ii) technologies developed by search engines to mine their own data assets to improve their relevance and create new functionalities on their own websites. We now discuss these two lines of related works in more details.

Technologies developed by e-tailers: As discussed in Section I, e-tailers, especially big e-tailers like Amazon, mine their data assets to extract signals that can be leveraged for their portal and thus for customer experience. For example, [15] describes Amazon's product recommendation algorithm. Note that many algorithms, like the product recommendation algorithm in [15], rely on data assets for which there is no "parallel" search engine data asset (e.g., user purchase history data used in product recommendation). Of course, other services such as query auto-completion are more similar to functionalities offered by web search engines.

Technologies developed by search engines: Web search engines

mine their data assets to improve their own relevance ranking and enable new search engine functionalities (e.g., related searches and query auto-completion). For example, past works discuss how to mine synonyms of entities using query logs [8], associate descriptive phrases or tags to entities using web crawl [7], find related queries [12] and perform query autocompletion [6]. Our innovation here is to exploit those web search engine data assets to mine signals *for the product search functionalities for e-tailers*.

VIII. Conclusions

In this paper, we presented the idea of using search engine data assets to enhance the user experience for product search for e-tailers. We have created three cloud data services (entity synonym data service, entity tagging data service and query-to-entity data service) to enable e-tailers to enhance user experience for product search on their sites. These data services provide all but the largest e-tailers with much needed user data that they lack.

Our current work continues to explore opportunities for other cloud data services useful for e-tailers derived from web search data assets that may add to the three we have proposed here. In addition, we also need to identify efficient techniques to incrementally refresh the three data services discussed here in order to respond to incremental additions to search query logs as well as updates to product catalogs of e-tailers (the algorithms proposed in this paper are geared towards bulk processing).

References

[1] Enceda Search. http://www.oracle.com/us/products/applications/commerce/endeca/overview/index.html.

[2] Fast Sharepoint Search. http://sharepoint.microsoft.com/en-us/product/capabilities/search/Pages/Fast-Search.aspx.

[3] U.S. Online Retail Sales to Reach $327 Billion by 2016. http://mashable.com/2012/02/27/ecommerce-327-billion-2016-study/, 2012.

[4] Parag Agrawal, Arvind Arasu, and Raghav Kaushik. On indexing error-tolerant set containment. In *SIGMOD*, 2010.

[5] Sanjay Agrawal et al. Exploiting web search engines to search structured databases. In *WWW*, 2009.

[6] Ziv Bar-Yossef and Naama Kraus. Context-sensitive query auto-completion. In *Proceedings of WWW*, 2011.

[7] Kaushik Chakrabarti, Surajit Chaudhuri, Tao Cheng, and Dong Xin. Entitytagger : automatically tagging entities with descriptive phrases. In *WWW Conference*, 2011.

[8] Kaushik Chakrabarti, Surajit Chaudhuri, Tao Cheng, and Dong Xin. A framework for robust discovery of entity synonyms. In *SIGKDD*, 2012.

[9] Kaushik Chakrabarti, Surajit Chaudhuri, Venkatesh Ganti, and Dong Xin. An efficient filter for approximate membership checking. In *SIGMOD*, 2008.

[10] Soumen Chakrabarti, Kriti Puniyani, and Sujatha Das. Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In *WWW*, 2006.

[11] Tao Cheng, Xifeng Yan, and Kevin Chen-Chuan Chang. Entityrank: searching entities directly and holistically. In *VLDB*, 2007.

[12] Nick Craswell and Martin Szummer. Random walks on the click graph. In *SIGIR Conference*, 2007.

[13] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. *ACM SIGOPS Operating Systems Review*, 41(3), 2007.

[14] Ravi Kumar and Andrew Tomkins. A characterization of online search behavior. *Data Engineering Bulletin*, 2009.

[15] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 2003.