Entity Synonyms for Structured Web Search

Tao Cheng, Hady W. Lauw, and Stelios Paparizos

Abstract—Nowadays, there are many queries issued to search engines targeting at finding values from structured data (e.g., movie showtime of a specific location). In such scenarios, there is often a mismatch between the values of structured data (how content creators describe entities) and the web queries (how different users try to retrieve them). Therefore, recognizing the alternative ways people use to reference an entity, is crucial for structured web search. In this paper, we study the problem of automatic generation of *entity synonyms* over structured data toward closing the gap between users and structured data. We propose an offline, data-driven approach that mines query logs for instances where content creators and web users apply a variety of strings to refer to the same webpages. This way, given a set of strings that reference entities, we generate an expanded set of equivalent strings (entity synonyms) for each entity. Our framework consists of three modules: candidate generation, candidate selection, and noise cleaning. We further study the cause of the problem through the identification of different entity synonym classes. The proposed method is verified with experiments on real-life data sets showing that we can significantly increase the coverage of structured web queries with good precision.

Index Terms—Entity synonym, fuzzy matching, structured data, web query, query log.

1 Introduction

Web search has evolved over the past few years, from a carefully selected hierarchy of bookmarks, to a huge collection of crawled documents that require sophisticated algorithms for identifying the few most relevant results, to, more recently, an advanced answering mechanism returning relevant media or short snippets of useful structured information.

With the ever-growing exposure of rich data on the web, user queries have also become much more diverse. Consider, for example, a query such as "*indy 4 near san fran*." In this example, the user is looking for movie theaters and show times for "*Indiana Jones and the Kingdom of the Crystal Skull*" nearby the city of "*San Francisco*." Such structured query is not a rare handpicked case; on the contrary, similar trends appear in other real scenarios for a variety of domains like products, recipes, weather, stock quotes, etc. The answers to such queries can often be found in structured data sources, such as product catalogs, movie databases, etc. *Structured web Search*, aiming at answering web queries with values from structured data, has the potential to significantly upgrade users' search experience from seeing webpages to rich and diverse structured data. Many recent works [19], [28], [14] have emerged in this area of structured web search in better understanding and supporting such structured user queries.

Effective structured web search requires a fast and accurate matching between the various query parts and the underlying structured data. In other words, a web search engine has to capture the web queries that target structured data, find the corresponding data source, and analyze the query into appropriate pieces that map to the structured data attributes.

An effective way to attack this problem is via the usage of dictionaries or lookup tables that can be produced from columns of structured data. One can use such lookup tables to simplify annotating query parts and determining which database or attribute to retrieve when returning results. For example, a lookup table for movies and one for cities can be used to handle the query "*indy 4 near san fran*."

However, a direct application of such dictionaries and lookup tables is not always possible when querying structured data. There is often a gap between what end users type and how content creators describe the actual data values of the underlying entities in a structured data source. Content creators tend to use high quality and formal descriptions of entities, whereas end users prefer a short, popular, more informal "synonymous" representation. In the example above, a movie database lists the full title of "*Indiana Jones and the Kingdom of the Crystal Skull*," when web users may type "*indy 4*." This phenomenon exists across virtually all domains. The movie "*Madagascar: Escape 2 Africa*" is more commonly known as "*Madagascar 2*." Apple's "*Mac OS X*" is also known as "*Leopard*." The digital camera "*Canon EOS 350D*" also goes by the names of "*Digital Rebel XT*" and "*EOS Kiss Digital N*." Meanwhile the newer model "*Canon EOS 400D*" also goes by "*Digital Rebel XTi*" and "*EOS Kiss Digital K*."

Besides the simple usage of dictionaries and lookup tables, we have also recently seen more complicated systems (e.g., [20], [27], [26]) to support keyword queries over structured data (e.g., databases). Being able to identify the "synonymous" representations is key to these tasks to allow fuzzy matching between queries and structured data.

However, existing approaches are not always successful in finding automatically such "synonymous" representations. Thesauri or WordNet [25] focus on language-based alterations but is insufficient when looking at the semantic

- *T. Cheng is with Microsoft Research in Redmond, One Microsoft Way, Redmond, WA 98052. E-mail: taocheng@microsoft.com.*
- *H.W. Lauw is with the Institute for Infocomm Research, Singapore. E-mail: hwlauw@i2r.a-star.edu.sg.*
- *S. Paparizos is with Microsoft Research in Silicon Valley, 1065 La Avenida, Mountain View, CA 94043. E-mail: steliosp@microsoft.com.*

Manuscript received 26 Nov. 2009; revised 4 Nov. 2010; accepted 5 June 2011; published online 20 July 2011.
Recommended for acceptance by D. Cook.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2009-11-0804.
Digital Object Identifier no. 10.1109/TKDE.2011.168.

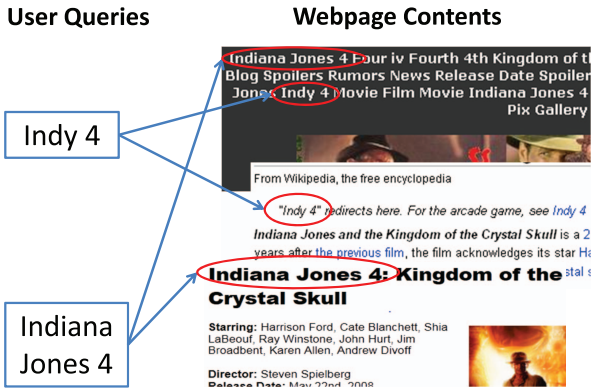

Fig. 1. Wisdom of both end users and content creators.

alterations necessary for vertical domains, such as movie, product, or company names. Substring matching works well for some cases ("*madagascar 2*" from above), fall short in others ("*escape africa*" would also be considered incorrectly for "*Madagascar: Escape 2 Africa*") and is hopeless for the rest ("*Canon EOS 400D*" has no string similarity to "*Digital Rebel XTi*"). Manual authoritative efforts can be of high quality, like Wikipedia's redirection or disambiguation pages. In such cases contributors construct the list of other titles of a given paper in order to capture variations in queries. However, as we will show experimentally in Section 6, Wikipedia's query expansion possibilities are limited to only very popular entities.

The same gap between users and content creators also exist in typical web search. In web search, it is alleviated by the efforts of some content creators, who resort to including known alternative forms within the content of a webpage (e.g., enriching pages about a movie with various alternative names of the movie as shown in Fig. 1) so as to facilitate a textual match by the search engine. End users resort to trying different queries (e.g., trying "Indy 4" or "Indiana Jones 4" to retrieve information about the movie as shown in Fig. 1) until they find some webpage that satisfies them. The web gathers the wisdom of both content creators and end users. Due to the scale of the web, there is enough webpage content produced that when considered in unison can handle most of the different query variations. However, the same cannot always be said for searching over structured data sources. It is not scalable to assume a manual solution where a content creator enters all known alternative forms for each attribute for each entity in a database.

In this paper, we propose a fully automated solution that can enrich structured data with "synonymous" representations (what we call *entity synonyms*). We leverage the collective wisdom generated by webpage content creators and end users toward closing the above-mentioned gap. The enriched structured data can be used as the basis for fast and accurate approximate matching via the usage of lookup tables and existing indexing techniques. Our solution can benefit many structured web search applications, such as the ones described in [20], [28], [26], in covering more structured web queries.

At a high level, we use a multistep data driven approach that relies on query and click logs to capture the wisdom of crowds on the web. We first retrieve relevant webpage urls that would be good surrogates or representatives of the entity. We then identify queries that have accessed at least one of these urls. We qualify which queries are more likely to be true entity synonyms by inspecting click patterns and click volume on a large subset of such urls. Finally, we clean the produced queries taking advantage of noise words that tend to appear frequently over entity synonyms of entities in the same domain.

Note that our approach focuses solely on query and click logs, and it does not require us to do content analysis on webpages. Content analysis is computationally more expensive given the relative complexity of full-length text documents when compared to concise queries. Speed of processing is of the essence due to the potential scale of structured data entries.

While this work focuses on generating entity synonyms for supporting more effective structured web search, this technology can also be applicable to general web search. Our experiment in Section 6.5 will show that we can capture many more general web queries with the addition of entity synonyms. In fact, our technique works well in practice and has been included in production as part of Microsoft's "Bing" search engine.

We summarize the contributions of this work as follows:

- We give formal definition of entity synonym, hypernym, hyponym in the context of a structured data source.
- We propose a fully automatic entity synonym generation framework, consisting of candidate generation, selection, and cleaning, which mines entity synonyms from query log by leveraging the wisdom of crowds.
- We study in depth the different entity synonym classes, and propose methods for detecting them.
- We verify our design using real data to show that our proposed approach can significantly help increase the coverage of web queries.

The rest of the paper is organized as follows: We start in Section 2 with formalizing the notions of entity synonym, hypernym, and hyponym; and the entity synonym finding problem. In Section 3, we describe our bottom-up data-driven solution to generate entity synonyms and in Section 4, we introduce several strategies to remove noise words from the produced entity synonyms further enhancing the results. In Section 5, we group the produced entity synonyms in classes and discuss the application value of each class. In Section 6, we perform a comprehensive experimental study to validate the proposed method on large-scale real-life data sets. We conclude with a review of related work in Section 7, and a summary of results and future work in Section 8.

2 Problem Definition

In this section, we will first give our formal definitions of entity synonym, hypernym, and hyponym, before defining the entity synonym finding problem.

2.1 Entity Synonym, Hypernym, and Hyponym

Let $\mathcal{E}$ be the set of entities over which the entity synonyms are to be defined. *An entity is an object or an abstraction with a distinct and separate existence from other objects/abstractions of the same type (those having similar attributes).* For example, "Indiana Jones and the Kingdom of the Crystal Skull" is an entity of the type *Movie*, and "Canon EOS 350D" is an entity

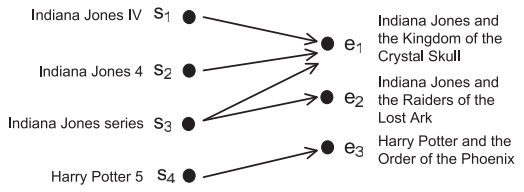

Fig. 2. Example mapping function $\mathcal{F}$.

of the type *Digital Camera*. Entities in a given $\mathcal{E}$ are of the same type. Hence, we may have different $\mathcal{E}$ sets based on the entity type. For example, we may have $\mathcal{E}_M$, which is a set of *Movie* entities; $\mathcal{E}_D$, which is a set of *Digital Camera* entities.

Let $\mathcal{S}$ be the universal set of strings, where each string is a sequence of one or more words. Users may use any number of different strings to refer to an individual entity or a subset of entities.

We assume that there exists an oracle function $\mathcal{F}(s, \mathcal{E}) \rightarrow E$, which is an ideal mapping from any string $s \in \mathcal{S}$ that users may think of (in order to refer to or retrieve entities) to the very subset of entities $E \subseteq \mathcal{E}$.

We are now ready to put forward our definitions of entity synonym, hypernym, and hyponym.

Definition 1 (Entity Synonym). *A string $s_1 \in \mathcal{S}$ is an **entity synonym** of another string $s_2 \in \mathcal{S}$ over the set of entities $\mathcal{E}$ if and only if $\mathcal{F}(s_1, \mathcal{E}) = \mathcal{F}(s_2, \mathcal{E})$.*

Definition 2 (Entity Hypernym). *A string $s_1 \in \mathcal{S}$ is an **entity hypernym** of another string $s_2 \in \mathcal{S}$ over the set of entities $\mathcal{E}$ if and only if $\mathcal{F}(s_1, \mathcal{E}) \supset \mathcal{F}(s_2, \mathcal{E})$.*

Definition 3 (Entity Hyponym). *A string $s_1 \in \mathcal{S}$ is an **entity hyponym** of another string $s_2 \in \mathcal{S}$ over the set of entities $\mathcal{E}$ if and only if $\mathcal{F}(s_1, \mathcal{E}) \subset \mathcal{F}(s_2, \mathcal{E})$.*

To illustrate the above definitions, we present an example of mapping function $\mathcal{F}$ in Fig. 2 defined for $\mathcal{S} = \{s_1, s_2, s_3, s_4\}$ and $\mathcal{E} = \{e_1, e_2, e_3\}$. $\mathcal{F}$ is represented as a bipartite graph, where a link exists from s_i to every $e_j \in \mathcal{F}(s_i, \mathcal{E})$. In this example, s_1 = "Indiana Jones IV" is an entity synonym of s_2 = "Indiana Jones 4" (vice versa) as both reference the same entity e_1, which is the movie titled "Indiana Jones and the Kingdom of the Crystal Skull." Meanwhile, s_3 = "Indiana Jones series" is an entity hypernym of s_1 and s_2, as s_3 references e_1 (which both s_1 and s_2 reference) and also e_2 (which neither s_1 nor s_2 does). Equivalently, s_1 and s_2 are entity hyponyms of s_3. s_4 = "Harry Potter 5" has no entity synonym, hypernym, or hyponym in $\mathcal{S}$.

2.2 Entity Synonym Finding Problem

We focus on the problem of finding entity synonyms. We believe that a similar approach may also be applicable to the related problems of finding entity hypernyms and hyponyms, but we keep these problems as future work.

Entity synonym finding problem. Formally, our formulation of the entity synonym finding problem is as follows: As input, we are given a set of entities $\mathcal{E}$ and a set of strings $U \subseteq \mathcal{S}$. Strings in U are "homogeneous," in the sense of belonging to the same class of attributes. For example, for $\mathcal{E}_M$, U_M could be a set of movie titles; for $\mathcal{E}_D$, U_D could be a set of digital camera names. As output, we would like to produce for each string $u \in U$, its set of entity synonyms $V_u = \{v \in \mathcal{S} | \mathcal{F}(u, \mathcal{E}) = \mathcal{F}(v, \mathcal{E})\}$. Note that the input $\mathcal{E}$ can be a singleton set.

In the problem formulation above, we do not assume that $\mathcal{F}$ is given. This is because, while we assume that such an oracle function exists (if only abstractly), we do not claim it is obtainable in practice. True $\mathcal{F}$ exists only in the collective minds of all users. Hence, the equality $\mathcal{F}(u, \mathcal{E}) = \mathcal{F}(v, \mathcal{E})$ that underlies Definition 1 cannot be determined exactly.

To resolve this, we propose to relax Definition 1, and instead estimate the equality $\mathcal{F}(u, \mathcal{E}) = \mathcal{F}(v, \mathcal{E})$ using real-life data. The gist of our approach is to approximate the synonymy between two strings by observing in real-life data how well these two strings have been used by users to retrieve equivalent subsets of entities.

We identify the following real-life web-based data sets as especially relevant for this approach:

Search data. A consists of a set of tuples, where each tuple $a = \langle q, p, r \rangle$ denotes the relevance score r of a webpage URL p for the search query $q \in \mathcal{S}$. For simplicity, in this paper, we assume r is the relevance rank of p, with rank 1 being the most relevant. A captures the "relevance" relationship between a query string and a webpage as determined by a *search engine*.

Click data. L is a set of tuples, where each tuple $l = \langle q, p, n \rangle$ denotes the number of times $n \in N^+$ that users click on p after issuing query $q \in \mathcal{S}$ on a search engine. L captures the "relevance" relationship between a query string and a webpage as determined by *search engine users*.

How these data sets may be used to find entity synonyms can be summarized as follows: Let $\mathcal{P}$ be the union of all webpages, and $\mathcal{Q}$ be the union of all query strings, in A and L. Since both A and L represent some form of relationship between query strings and webpages, we can learn from A and L, respectively, two functions $\mathcal{G}_A(q, \mathcal{P}) \rightarrow P$ and $\mathcal{G}_L(q, \mathcal{P}) \rightarrow P$, which map a query string $q \in \mathcal{Q}$ to the subset of relevant webpages $P \subseteq \mathcal{P}$. Assuming that for any entity $e \in \mathcal{E}$, there always exist webpages that are appropriate and representative surrogates of e (which is a reasonable assumption given the scope of the web), we consider it probable that two query strings q_1 and q_2 are entity synonyms if $\mathcal{G}_A(q_1, \mathcal{P}) \approx \mathcal{G}_L(q_2, \mathcal{P})$.

Definition 4 (Entity Synonym-Relaxed). *A string $s_1 \in \mathcal{S}$ is an **entity synonym** of another string $s_2 \in \mathcal{S}$ over the set of webpages $\mathcal{P}$ (keeping in mind the actual reference set of entities $\mathcal{E}$) if $\mathcal{G}_A(s_1, \mathcal{P}) \approx \mathcal{G}_L(s_2, \mathcal{P})$.*

Entity synonym finding problem (relaxed). In this paper, our specific problem formulation is as follows:

- Given: A set of "homogeneous" strings U; the data sets A and L; and the reference set of entities $\mathcal{E}$.
- Output: We would like to produce for each string $u \in U$, its set of entity synonyms $W_u = \{w \in \mathcal{S} | \mathcal{G}_A(u, \mathcal{P}) \approx \mathcal{G}_L(w, \mathcal{P})\}$.

Note that in practice, it is not always necessary to list out all the entities in $\mathcal{E}$, as long as it is well understood what type of entities may belong in $\mathcal{E}$. The nature of $\mathcal{E}$ helps define the entity synonyms, as for instance, the string "Jaguar" might have different sets of entity synonyms, depending on whether $\mathcal{E}$ is a set of animals or a set of cars.

3 Candidate Generation and Selection

To solve the entity synonym finding problem, we propose a two-phase solution. In Section 3.1, we describe how given

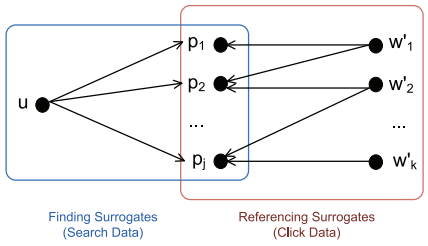

Fig. 3. Candidate generation phase.

the input set U, we will generate the set of entity synonym candidates W'_u. In Section 3.2, we describe how we select the final set of entity synonyms.

3.1 Candidate Generation

Considering every possible string $s \in \mathcal{S}$ as an entity synonym candidate for any input string $u \in U$ is impractical, since only a small subset of $\mathcal{S}$ is likely to be the true entity synonyms of u. We propose to quickly generate entity synonym candidates for a given u in two steps, as shown in Fig. 3. First, we seek the webpages that are good representation for entities referenced by u. We term these webpages *surrogates* of u. Second, we find out how users refer to these surrogates. We now zoom into each of these steps in details.

Finding surrogates. The web has inarguably become the largest open platform for serving various kinds of data. It is almost certain that entities we have in our entity set $\mathcal{E}$ would have some representation on the web. This representation (or surrogates) come mostly in the form of webpages. For a particular digital camera (e.g., Canon EOS 400D), its surrogates may include a page in the manufacturer's site listing its specifications, an eBay page selling it, a Wikipedia page describing it, a page on a review site critiquing it, etc.

Moreover, data appears in various forms on the web. Consider a seller on eBay who explicitly listed some of the alternative ways to access the data to help increase the chances of her item being retrieved, e.g., "Digital REBEL XTi" and "400D." Or consider unstructured web articles, where the various names of one camera model line are described in the text of a webpage, e.g., "Digital IXUS," "IXY Digital," and "PowerShot Digital ELPH." Data, once appearing on the web, gets enriched in various ways, which enables alternative paths for people to access the same information.

As shown in Fig. 3, our approach is to use the *Search Data* A to find webpage surrogates for a given u. A is derived by issuing each $u \in U$ as a query to the Bing Search API[1] and keeping the top-k results. Based on A, we can define the mapping function $\mathcal{G}_A(u, \mathcal{P})$ between u to the set of top-k pages, as shown in

$$\mathcal{G}_A(u, \mathcal{P}) = \{a.p | a \in A,\ a.q = u \wedge a.r \leq k\}. \quad (1)$$

The choice of the value of k is empirical. Our current implementation uses $k = 50$, which we found to be sufficient in our experiments. The tradeoff when changes the value of k is quantity versus the quality of synonyms discovered. If a synonym does not hit any page in the top k retrieved documents (e.g., top 50), it is unlikely that it will quality as a high-quality synonym.

1. http://search.bing.com/developer.

Definition 5 (Surrogate). *A webpage $p \in \mathcal{P}$ is a **surrogate** for u if $p \in \mathcal{G}_A(u, \mathcal{P})$.*

Based on Definition 5, we consider a webpage p a surrogate of u, if p is among the top-k most relevant results for query u in the data set A.

It may also be possible to use *Click Data* in place of *Search Data*, whereby a webpage is a surrogate if it has attracted many clicks when the entity's data value is used as a query. In some cases, clicks may be a better indicator of relevance than the search engine's relevance function. However, clicks may not work for all entities, as the entities' data values usually come in the canonical form (e.g., the full title name of a movie), and therefore may not be used as queries by people. Meanwhile, a search engine is usually friendly to any data value as query for finding relevant pages, which is why we rely on a hybrid approach of using *Click Data* for data values that are popular queries, and *Search Data* for unpopular queries.

One potential issue with using search API is that our mining relies on the returned pages of search engines. In particular, if search results are very ambiguous, the effectiveness could be compromised. However, on average from our empirical evaluation, this is not the case for most of entities, especially considering the queries we issue to search API are mostly canonical entity names. It is important to notice that even in cases where search results are ambiguous, as long as there is a core subset of relevant webpages about the entity, we can find something useful. Collectively, this core subset could outweigh the noises from other ambiguous/unrelated webpages.

Referencing surrogates. Having identified u's surrogates, we next ask how users would access those surrogates. Remember that these surrogates are webpages available for access by the general public. Again, search engine is the primary channel people use for accessing information on the web. We can therefore regard the queries issued to get to these surrogate pages as the various ways users use to refer to the entities represented by these pages. Consequently such queries are good entity synonym candidates for u.

As shown in Fig. 3, click data L offer us the mapping from candidates to surrogates. Based on L, we can define the mapping function $\mathcal{G}_L(w', \mathcal{P})$ between a potential entity synonym candidate w' to the set of clicked pages, as shown in

$$\mathcal{G}_L(w', \mathcal{P}) = \{l.p | l \in L, l.q = w' \wedge l.n \geq 1\}. \quad (2)$$

Definition 6 (Entity Synonym Candidate). *A string w' is a **entity synonym candidate** for u if and only if $\mathcal{G}_A(u, \mathcal{P}) \cap \mathcal{G}_L(w', \mathcal{P}) \neq \emptyset$.*

Based on Definition 6, we regard w' as an entity synonym candidate for u if there is at least one surrogate of u that has been clicked when w' is issued as a query. Therefore, the candidate set for u is $W'_u = \{w' | \mathcal{G}_A(u, \mathcal{P}) \cap \mathcal{G}_L(w', \mathcal{P}) \neq \emptyset\}$.

3.2 Candidate Selection

Not all candidates generated in the previous phase are equally good. Some are more likely to be actual entity synonyms than others. To estimate the likelihood that a candidate w' is an entity synonym of the input value u, we identify two important measures that can be captured from search data A and click data L. The two measures,

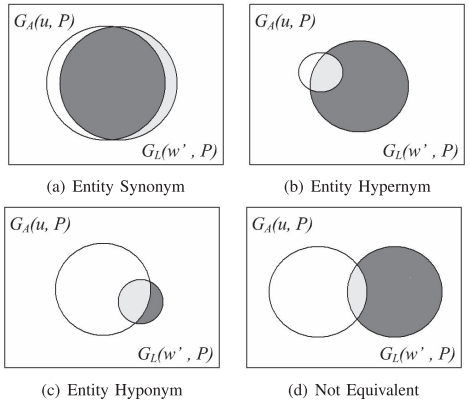

Fig. 4. Venn diagram illustration.

respectively, capture the *strength* and *exclusiveness* of the relationship between a candidate w' and the input value u.

Intersecting page count (IPC). Here, we seek to measure the strength of relatedness between an input value u and a candidate w'. In the candidate generation phase, we look for u's candidates by looking at queries (w') for which the following holds: $\mathcal{G}_A(u, \mathcal{P}) \cap \mathcal{G}_L(w', \mathcal{P}) \neq \emptyset$. In (3), we derive the $IPC(w', u)$ as the size of this intersection. Intuitively the higher the value of IPC is, the larger the size of the intersection is, the more common pages have been referred to using u and w', and the more likely u and w' would be related to one another

$$IPC(w', u) = |\mathcal{G}_L(w', \mathcal{P}) \cap \mathcal{G}_A(u, \mathcal{P})|. \quad (3)$$

Intersecting click ratio (ICR). Another indicator for the strong relationship between w' and u is if a majority of the clicks resulting from w' as a query land on u's surrogate pages more often than on nonsurrogate pages. The click ratio measure $ICR(w', u)$ is determined as shown in (4). The higher the value of $ICR(w', u)$ is, the more exclusive the relationship between w' and u is, and the more likely w' would be an entity synonym of u

$$ICR(w', u) = \frac{\sum_{l \in L,\ l.p \in \mathcal{G}_L(w', \mathcal{P}) \cap \mathcal{G}_A(u, \mathcal{P})} l.n}{\sum_{l \in L,\ l.p \in \mathcal{G}_L(w', \mathcal{P})} l.n}. \quad (4)$$

We use a Venn diagram illustration in Fig. 4 to describe how the above two measures work in selecting the best entity synonyms. Consider the example where the input value u is the movie title "Indiana Jones and Kingdom of the Crystal Skull." Fig. 4a illustrates the case where a candidate w' (e.g., "indiana jones 4") is a likely entity synonym of u. The sets denote the webpages that are retrieved by u ($\mathcal{G}_A(u, \mathcal{P})$) and are clicked on for query w' ($\mathcal{G}_L(w', \mathcal{P})$), respectively. In this case, the size of the intersection of the two sets is large, indicating a high IPC value. For the set $\mathcal{G}_L(w', \mathcal{P})$, the darkly shaded (resp. lightly shaded) area indicates the subset of pages getting the most clicks (resp. fewer clicks). In this case, most of the clicks fall within the intersection, as opposed to outside of the intersection, indicating a high ICR value. Thus, w' is likely an entity synonym of u.

Both IPC and ICR also help to weed out candidates that are related, but not entity synonyms. Fig. 4b illustrates the case of an entity hypernym (e.g., "indiana jones"). Since an entity hypernym considers a broader concept, it may be used to refer to many more pages (e.g., concerning other Indiana Jones movies), and consequently most of the clicks may fall outside of the intersection (low ICR). An entity hyponym concerns a narrower concept, for which there might be more specific pages about the concept outside of the intersection that receive the most clicks (see Fig. 4c). Finally, a candidate such as "harrison ford" is related since it concerns the main actor of the movie. However, it is not an equivalent concept to the movie, and it would have low IPC and ICR since most clicks would fall on pages about the actor, rather than on those about the movie (see Fig. 4d).

We produce the final entity synonym by applying threshold values β and γ on IPC and ICR, respectively, i.e., $W_u = \{w' | w' \in W', IPC(w', u) \geq \beta \wedge ICR(w', u) \geq \gamma\}$. To help determine the right settings for β and γ, we will experiment with varying β and γ in Section 6.

TABLE 1
Examples of Candidates

Indiana Jones and the Kingdom of the Crystal Skull	The Lord of the Rings: The Return of the King
kingdom of crystal skull	lord of rings 3
indiana jones 4	return of the king
indy iv trailer	lord of rings 3 review
indiana jones 4 review	www.lotr.com

4 Cleaning

Naturally, the query strings users may use within the data sets may not be exactly the same as those that users might have used to refer to entities in $\mathcal{E}$. Hence, the entity synonym candidates or entity synonyms output from the selection phase may have some noise in them as shown in examples in Table 1. While these noise words (e.g., "trailer," "review") are informative by themselves, possibly covering facets about entity, they should not be included as part of the entity synonym to ensure the high quality of synonyms generated. We therefore propose a cleaning module to deal with such noises.

Our approach is to construct a white list of noise phrases, and to use regular expression rules to remove these noise phrases from any entity synonym candidate that contains them. We identify three major types of noise phrase that may be present within the entity synonym candidates: 1) *common noise*, 2) *context-specific noise*, and 3) those entity synonym candidates that are more likely to be *entity hypernyms*.

The benefit of cleaning is fourfold. First, it helps remove many false candidates. Second, as a result of removing noise, new candidates can be generated (e.g., we can get "indy iv" after removing noise word "trailer" from "indy iv trailer"). Third, it can help us in the identification of the final entity synonyms, as we can get better aggregated statistics about such candidates after noise removal. Finally, cleaning entity hypernym candidates helps us to focus on entity synonyms.

4.1 Common Noise

Common noise refers to noise phrases that may appear in entity synonym candidates regardless of the specific domain. They are often noninformative and generally should not be part of any entity synonym. Examples of such noise are "www.", ".com," ".net," etc. The list of common noise can be constructed based on mining across entity synonyms produced from many different domains

and determining frequent words that appear throughout all entity synonyms. Note that stop words are not considered common noise and their removal can alter the value of the produced entity synonyms.

4.2 Context-Sensitive Noise

The second type of noise is sensitive to the specific context of the domain of interest, and therefore is called context-sensitive noise. Since the produced entity synonyms were the result of a url-query graph, it is very common to have patterns of additional common words. For instance, entity synonyms for movie titles contain additional keywords like "trailer," "review," "sound track." As another example, it is also common for "best deal," "spec," "price" to appear together with camera names. Note that the context-sensitive noise for the domain of movie titles is very different from those for the domain of camera names.

Toward removing context-sensitive noise, we draw the following insights. First, such noise phrases appear in the entity synonym candidates of many input values. In Table 1, we see the noise "review" occurring within the candidates ("indiana jones 4 review" and "lord of rings 3 review") for both movies. Second, such noise should not appear within the original input value. In this case, "review" does not occur within both movie titles.

We regard a string s as a set of words, and denote a phrase x occurring within a string s with $x \subset s$, and x *not* occurring within s with $x \not\subseteq s$. We consider only phrases that occur within a candidate to be possible noise.

Definition 7 (Context-Sensitive Noise). *A phrase x is a **context-sensitive noise phrase** in the context of U if and only if x is frequent among candidates whose corresponding input values do not contain x, i.e.,*

$$\frac{|\{u|u \in U, x \not\subseteq u \wedge x \subset w' \wedge w' \in W'_u\}|}{|U|} \geq \alpha.$$

Specifically, the fraction of input values (not containing the phrase) with at least one candidate containing the phrase has to be above a given threshold $\alpha \in [0, 1]$. We will experiment with varying α in Section 6.

An example of a noise phrase that appears in Table 1 is "review." The phrase "review" never occurs within either movie title, but occurs within the candidates of two out of two movie titles (frequency is 2). Thus, "review" is a context-sensitive noise phrase and will be removed from the candidates. However, in some cases, what appears to be a noise phrase may occur within an input value, and we take care not to remove it from such an input value. For instance, "review" is a not considered noise for the movie title "Newsnight Review."

Frequency of a noise phrase is defined over the set of input values, and not over the set of all candidates. This is to avoid penalizing those input values with many candidates. It is possible that an input value may have many candidates containing a valid phrase that does not occur in the input value. For example, the misspelling "Cannon" may be a valid phrase within many candidates for the camera name "Canon EOS 350D." The same phrase may not appear within the candidates of different brands (e.g., Sony). Counting frequency based on the number of candidates may *wrongly* classify "Cannon" as frequent and thus a noise phrase.

TABLE 2
Examples of Cleaned Candidates

Indiana Jones and Kingdom of the Crystal Skull	The Lord of the Rings: The Return of the King
kingdom of crystal skull	lord of rings 3
indiana jones 4	return of the king
indy iv	

Table 2 shows the result after cleaning the candidates in Table 1. Note that we have removed such common noise phrase as "www.", ".com," as well as context-sensitive noise phrases such as "trailer," "review."

4.3 Entity Hypernyms

In addition to removing noise phrases from within entity synonym candidates, we also remove from consideration those entity synonym candidates that appear as candidates for more than one input value. Such candidates are more likely to be entity hypernym candidates (see Definition 2). For example, "Indiana Jones" may appear as a candidate for all four movies in the Indiana Jones series, and thus is not considered an entity synonym for any of them. Note, however, that this approach only removes those entity hypernym candidates with at least two entity hyponyms within the input set of entities.

5 Classes of Entity Synonyms

After the generation and cleaning phases, we categorize the discovered entity synonyms into several distinct classes, which we will enumerate shortly. This classification is useful as some applications may be interested in specific classes of entity synonyms. For instance, a search application may have a built-in spellchecking ability, but is not able to handle other kinds of entity synonyms well. Furthermore, this exercise helps to measure the value of the entity synonyms produced, as some classes are inherently more challenging, and thus more valuable, to produce than others. In the following, we will first describe the various classes, and then show the significance in which our approach could discover the different classes with lower complexity and higher precision than comparative approaches.

5.1 Classes of Entity Synonyms

We observe six distinct classes that our entity synonyms fall into with examples shown in Table 3. In Section 6, we will further discuss what fraction of entity synonyms fall into each class for various domains (e.g., movie titles, camera models).

Normalization. This class covers variances from the original input value that can be resolved by stemming and removal of punctuation or other special characters, e.g., [-, .:();]. For the movie title "the Dark Knight," examples include "the dark knights" and "the dark-knight." For the camera model, "Canon EOS 350D," its normalization entity synonyms includes "canon eos-350d." We identify entity synonyms of this class using a stemmer and a list of stop words as shown in Algorithm 1. Performing normalization is important because it also helps to identify the other classes.

TABLE 3
Entity Synonym Class Examples for "the Dark Knight"

Entity Synonym Classes	Example
Normalization	the dark knights, the dark-knight
Spelling	the dark knigth, the dark night
Subset	dark knight
Superset	batman the dark knight
Acronym	TDK
Atypical	batman 2

Algorithm 1. Identifying Normalization Entity Synonym

- Input: input value u, entity synonym $w \in W_u$
- Output: T/F whether w is a normalization entity synonym of u

1: $w_1 \leftarrow removeSpecialCharacters(w)$
2: $w_2 \leftarrow stem(w_1)$
3: $u_1 \leftarrow removeSpecialCharacters(u)$
4: $u_2 \leftarrow stem(u_1)$
5: **if** $w_2 = u_2$ **then**
6: return TRUE
7: **else**
8: return FALSE
9: **end if**

Spelling. This class covers various spelling errors of the input value. Generally, there are two kinds of errors, namely *mistyping* which comes from accidentally missing or pressing wrong keys, e.g., "the dark knight," and *misspelling* which are common misconceptions of how a word is spelled, e.g., "the dark night." Identifying this class can be accomplished with a high quality dictionary or spell-checker. In our case, we make use of "Bing" Search's query alteration capability.

Subset. Suppose that $wordset(s)$ respresents the set of words in a string s after stop word removal, we define subset entity synonyms as follows:

Definition 8 (Subset Entity Synonym). *An entity synonym $w \in W_u$ is a **subset entity synonym** of u if and only if $wordset(w) \subset wordset(u)$.*

This is an important class given that the original input values tend to be long and formal, e.g., movie title "Madagascar Escape 2 Africa," while most users would simply type "Madagascar 2" to refer to the same movie. Meanwhile, it is crucial to note that not all proper subsets are proper entity synonyms, e.g., "Africa" or "Escape."

Superset. Entity synonyms of this class are proper supersets of the input value.

Definition 9 (Superset Entity Synonym). *A entity synonym $w \in W_u$ is a **superset entity synonym** of u if and only if $wordset(u) \subset wordset(w)$.*

Examples include "batman the dark knight" whereby a user expresses additional information about the movie such as a character or actor. While categorizing such entity synonyms are relatively simple, it is much more complex to discover such entity synonyms in the first place, mainly because there could be so many proper supersets that are plainly not entity synonyms, e.g., "the dark knight review."

Acronym. Given that s_i denotes the ith nonempty space character in a string s, and that $|s|$ denotes the total number of nonempty space characters in s, we define acronyms as follows:

Definition 10 (Acronym Entity Synonym). *A entity synonym $w \in W_u$ is an **acronym entity synonym** of u if and only if w meets all of the following conditions:*

- $|w| < |u|$
- $\forall w_i,\ \exists u_m,\ w_i = u_m$
- *For $j > i$, **if** $(w_i = u_m) \wedge (w_j = u_n)$, **then** $n > m$*
- $wordset(w) \not\subseteq wordset(u)$

Acronyms are shorter than the input value, and each character in the acronym is derived from a character in the input value, while preserving the original sequence. The last condition ensures that acronyms are not simply subset entity synonyms of the input value. This definitions covers various acronyms such as "HSM3" for "High School Musical 3" and "XML" for "Extensible Markup Language." In addition to the input value u, we also accommodate cases where a word within u is substituted with an equivalent word. For instance, for movie title "Young and Restless," its acronym is "Y&R," which is detectable given that "&" is recognized as a proper substitute of "and." Similar substitutions may appear for numbers, e.g., "FF7" for "Final Fantasy VII."

Atypical. The remainder falls into this class. These are mainly entity synonyms that do not share any text similarity with the input value, or those with a few overlapping words but neither fully contains or are fully contained by the input value. An example is "batman 2" for the movie "the Dark Knight." Consequently, this is the most challenging class of entity synonyms to produce using other approaches, and where we believe we have made the most valuable contribution.

In addition to the above classes, there are also composite classes, such as $Spelling + Normalization$ (e.g., "the dark-knight"), $Normalization + Superset$ (e.g., "batman dark-knight"), etc.

5.2 Complexity Analysis

Here, we argue that our approach of first generating and then categorizing the entity synonyms offline is superior to the alternative approaches of either attempting to handle the entity synonyms online or constructing the various classes of entity synonyms from scratch offline. In the following discussion, we will focus on the three classes: *Subset*, *Superset*, and *Atypical* where we believe we make the most valuable contribution. The two classes of *Normalization* and *Spelling* are well-studied problems, and they can be accomplished online. While it is not efficient to match *Acronym* entity synonyms online, relatively few entity synonyms fall into this class as will be shown in Section 6.

Subset. A reasonable online approach to handle this class is to construct an inverted index based on keywords found in the input values. However, matching a subset of an input value to retrieve relevant entities may result in a low precision. For instance, the keyword "dark" may retrieve the movie "Dark Knight," in addition to many other unrelated movies.

Meanwhile, attempting to construct such entity synonyms from scratch is a grossly inefficient approach. For a

given input value of length n words, there could be as many as $\sum_{r=1}^{n-1} P_n^r$ permutations that are proper subsets. For the above movie title "Indiana Jones and the Kingdom of the Crystal Skull," after removing stop words, there are 325 distinct permutations with the remaining five words {"Indiana," "Jones," "Kingdom," "Crystal," "Skull"}. Of these, only 38 have ever been asked as a query with a total of 530K impressions. Only 18 are proper entity synonyms with a total of 1,855 impressions. Weighted by the impressions, this results in a very poor precision of 0.3 percent. In comparison, as Section 6 will show, we achieve 87 percent weighted precision for the movie domain, indicating that we produce not just the high quality but also the most popular entity synonyms.

Superset. Similarly to the previous class, an inverted index approach will introduce many false positives. Furthermore, this class is even more complex to produce from scratch because the number of possible additional words to the input value is theoretically infinite, while only a very small number will be good entity synonyms. Our click log lists close to 20K queries starting with "dark knight." Of the top 30 such queries in terms of impressions, only four are proper entity synonyms with a weighted precision of only 11 percent. Many candidates are simply wrong such as "dark knight review," "dark knight spoof," or "dark knight poster." This is where our cleaning phase in Section 4 plays a very important role in weeding out the incorrect ones.

Atypical. Given that entity synonyms of this class share little text similarity with the input values, an online inverted index approach will fail. For instance, such an index will not retrieve the movie "Dark Knight" given "batman 2" as a query. On the other hand, it is impossible, even for a manual expert, to list all such entity synonyms. A brute force approach would produce an infinite number of candidates. This is where bringing in the wisdom of the crowds, in the form of user queries and clicks, really pays off in terms of producing good candidates and in scoring them as discussed in Section 3.

In summary, our approach performs better because we generate fewer and better candidates to begin with, owing to our generation phase that is driven by click logs. In addition, by relying on a list of high-quality entity synonyms generated offline, we can do the fuzzy matching against the database of entities at a much higher precision than an online inverted index approach.

6 Experiments

We implemented our entity synonyms solution as a stand-alone tool, which is now a component in the Helix [27] project. We use SQL server as back end store for our query and clicks logs having a clustered index on queries and urls, respectively. We used query and click logs from "Bing" Search that were available to us for the months of July to November 2008. Note that due to proprietary and privacy concerns we cannot share all the details of the click and query logs.

The main target application of our solution is enabling fuzzy matching over (semi)structured data represented as lookup tables or dictionaries. To this extend we report results on a variety of available data sets used to produce entity synonyms. Our two main data sets are: D1) the titles of the top 100 movies of 2008 as reported by the Box office [8], and D2) a collection of 882 canonical camera product titles crawled from MSN Shopping [24]. These data sets have different characteristics that affect the entity synonym process. The movies data have very popular entries, e.g., {Shrek the Third}, the cameras have a strict canonical form using a combination of brand, product family and model, e.g., {Canon EOS 400D}. In addition, we also use other data sets to test broadness, D3) a collection of 2,117 camcorder product titles, D4) 937 recent car models, D5) 448 nba athlete names.

All experiments were done on a single windows 2003 server workstation with 8 GB of RAM and 2 TB of hard disk space. Data sets went through all phases of our entity synonym generation process and took on average under 1 second to produce results for each input entry. As this is an offline process, we feel the speed by which entity synonyms are generated is adequate. To measure the speed of a potential online process we loaded the expanded dictionaries with our entity synonyms in a prefix-tree/trie implementation and run over queries from the log—we found that on average matching happened in 3.2 milliseconds.

To quantify the effectiveness of our approach we performed a detailed study on the selectivity of our parameters, evaluated the effect of query alterations and finally, compared against other approaches that could potentially be used for the same purpose (discussed in Section 6.4).

Note that during the evaluation process we required some manual labeling for testing. For this purpose we used the popular Mechanical Turk platform. We used five judges to produce scores. For each entry in our data sets we created a list of pairs between the entry and all the produced entity synonym candidates. Essentially, the judges were given lists of string pairs and a choice box with three entries 1) entity synonym ($\cong$), 2) entity hypernym ($\supset$) or entity hyponym ($\subset$), and 3) not equivalent (NE) ($\neq$). For example, "bike ($\neq$), ($\cong$), ($\supset$-$\subset$) bicycle" would be used to score the relationship between bike and bicycle. After our judges scored all string pairs, we aggregated the results and only kept as "truth" the pairs where at least three judges agreed.

6.1 Parameter Sensitivity

In this section, we evaluate in detail the behavior of **IPC**, **ICR**, and **Cleanup** thresholds. As our primary target class of applications is *fuzzy matching* or *expansion* of a dictionary of related entities. To capture the effect of these parameters on entity synonym production we use the notions of precision, weighted precision, hit ratio, and coverage increase.

Precision is described with $p = \frac{TP}{TP+FP}$, where TP = True Positives, FP = False Positives. *Weighted Precision* takes into account the number of times each entity synonym was asked as a stand-alone query using exact match on our query logs (freq(q)), hence

$$wp = \frac{\sum freq(TP)}{\sum freq(TP) + \sum freq(FP)}.$$

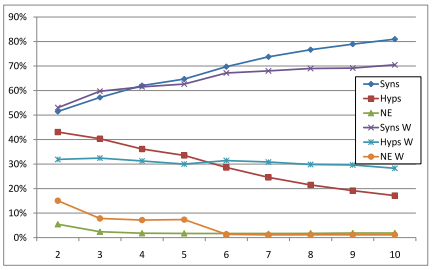

Fig. 5. IPC precision analysis.

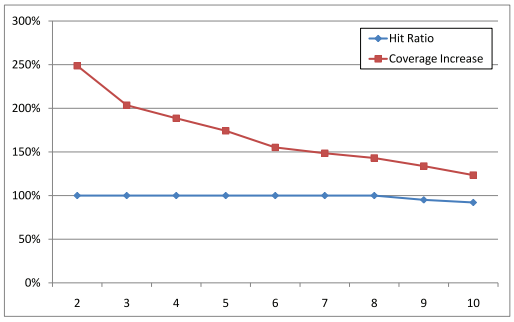

Fig. 6. IPC coverage increase and hit ratio.

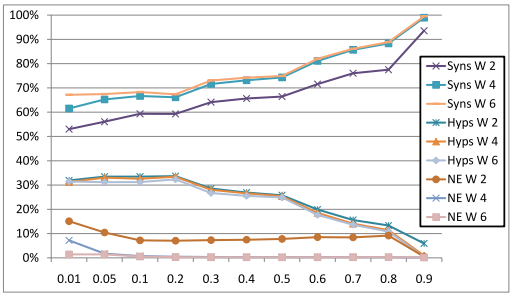

Fig. 7. ICR precision analysis, for IPC 2, 4, 6.

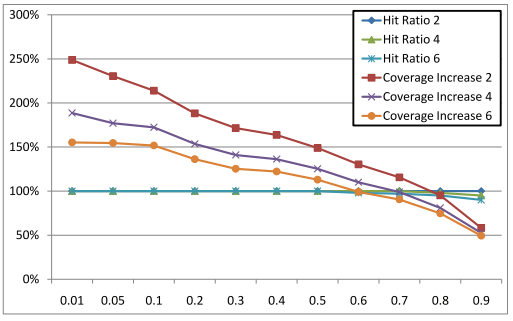

Fig. 8. ICR coverage and hit ratio for IPC 2, 4, 6.

We calculate precision and weighted precision for our three buckets of labels, entity synonyms (Syns), entity hypernymsor hyponyms (Hyps), and not equivalent.

To balance the precision measurement, we look at Hit Ratio and Coverage Increase. *Hit Ratio* is the percentage of entries that produce at least one entity synonym over all the entries in a dictionary, $HR = \frac{nonzeroentries}{allentries}$. As for coverage, given a set of queries Q, we call coverage C the sum of the number of times $freq(q_i)$ that each query q_i in Q was asked. In other words, $C_Q = \sum_{q_i} freq(q_i)$. Given a dictionary of entities E with coverage C_E and a set of produced entity synonyms for those entities S with coverage C_S, we call *Coverage Increase* the ratio $CI = \frac{C_S - C_E}{C_E}$.

6.1.1 Intersecting Page Count

First we focus on the **IPC**. Results for our (D1) movies data set and measurements of Precision, Weighted Precision, Hit Ratio, and Coverage Increase for various IPC thresholds $\beta \in [2, 10]$ are shown in Figs. 5 and 6. As discussed in Section 3, IPC is a strong indicator that results are related with each other. The higher the IPC, the stronger is the relation. As such, we notice that the Not Equivalent numbers are very low for both weighted (NE W) and unweighted (NE) scores. As we increase the threshold, entity synonym (Syns) precision improves and entity hypernym/hyponym (Hyns) reduces. When we look at the more interesting weighted results (Syns W, Hyns W), we see that effect being reduced. The reason is that although IPC is crucial to show strong relation, it is difficult to distinguish between popular entity hypernyms/hyponyms and entity synonyms that carry the most weight. As a result he Hyns W curve does not go down significantly when IPC value is increased as shown in Fig. 5. This is exactly we need the ICR measure to further help. Hit Ratio remains steady at 100 percent, meaning that we produce results across the board for the popular movies data. Coverage increase reduces as we increase IPC due to the removal of many entries. Yet, even at value $\text{IPC} \geq 10$ Coverage Increase is at 120 percent, meaning that our entity synonyms are very useful as they more than double $(2.2\times)$ the original coverage.

6.1.2 Intersecting Click Ratio

We then consider the **ICR**. Results for the movies (D1) data set are shown in Figs. 7 and 8. To test the combination of IPC and ICR, we used the threshold values of $\beta \in \{2, 4, 6\}$ for IPC and we did a range of threshold values for ICR $\gamma \in [0.01, 0.9]$. We believe the weighted precision measurements are much more interesting than absolute precision when it comes to web queries, so to simplify the figures and discussion we focus on it. The ICR parameter adds weight to IPC, essentially giving a strong indication whether we are looking at an entity synonym or an entity hypernym/hyponym. As a result, when we increase the ICR parameter we see the entity synonym precision going up (Syns W 2,4,6) and the entity hypernym/hyponym (Hyns W 2,4,6) reducing. On the other hand, this also affects negatively the coverage increase, reducing that number. Hit ratio again remains close to 100 percent meaning there is at least one entity synonym for every original movie entry. It is worth noting that for IPC 4 and 6 the curves are very close to each other on the precision measurement but there is a noticeable difference on coverage. Anecdotally, we have noticed the same for IPC higher than 6. Thus, we believe that IPC 4 is a good default value when it comes to various ICR settings. Some interesting ICR values act as local maxima, 0.1, 0.5, and 0.7 show good balance between precision and coverage increase compared to their immediate neighbors and are suggested depending on the application needs.

6.1.3 Cleaning

Cleaning is a process that takes away noise words and entity hypernyms that appear across dictionaries. The previous experiments were done without any cleanup at all. To measure its effect we fixed the IPC parameter to 4, and tried

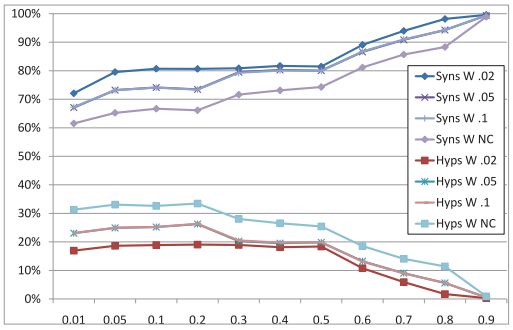

Fig. 9. Cleanup precision analysis, for IPC 4, all ICR values and cleanup thresholds of 0.02, 0.05, 0.1, and NC.

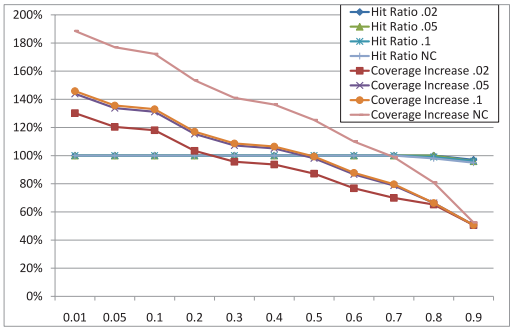

Fig. 10. Cleanup coverage and hit ratio, for IPC 4, all ICR values and cleanup thresholds of 0.02, 0.05, 0.1, and NC.

three thresholds for cleanup $\alpha \in \{0.02, 0.05, 0.1\}$ in addition to no cleanup (NC). The plots in the results show the various ICR parameters, summarized in Figs. 9 and 10. An immediate observation is that cleanup has a dramatic effect on both weighed precision and coverage increase. Entity synonyms precision increases across the board and coverage increase gets reduced. Intuitively, this is justified because the number of true entity synonyms remain stable, whereas cleanup further removes multiple entity hypernyms/hyponyms and the frequency weight they carry.

An interesting observation is that threshold 0.05 and 0.1 produce almost identical results, this is also true for higher values than 0.1. This is attributed to the fact that mining for common noise words fails once the threshold parameter starts becoming high. Again interesting local maxima appear for ICR 0.1 and 0.5. When considering the coverage increase, we have an indication that parameters IPC 4, ICR 0.1, and threshold 0.05 seem like an interesting combination producing good results of 75 percent entity synonym weighted precision and 130 percent coverage increase. If looking for higher quality, with ICR 0.7 or 0.8 we achieve precision at 92-95 percent with coverage increase between 75-80 percent. That is a significant effect when considering dictionary expansion or querying over (semi)-structured data as it is close to doubling the original coverage while maintaining a very high precision threshold.

6.2 Broad Applicability

We tested our approach on multiple data sets and results are summarized in Table 4. For space considerations we present only a few columns. The movie data set discussed in detail above is also included for comparison. Syns, Hyps, Hit Ratio, and Coverage Increase were defined above. Syns per entry shows the average number of entity synonyms we generated per dictionary entry. As seen on the various data sets the effectiveness of our approach varies. On cameras and camcorders the canonical form of the products is too restrictive and the original coverage is small, hence entity synonyms have a big effect on the coverage increase—even though hit ratio is relatively small on camcorders data set. For cars we see a large increase but also a large hit ratio as cars are popular entries asked by users using a variety of strings and nicknames. Finally, nba athletes show a small coverage increase, as there are not many entity synonyms one can use to ask for an athlete.

TABLE 4
Multiple Data, IPC 4, ICR 0.1, Cleanup 0.05

	movie	camera	camcord	cars	nba
Syns	87.18%	86.27%	87.24%	90.12%	85.30%
Hyps	10.99%	12.95%	11.19%	8.35%	14.23%
Hit Rat	100%	86.96%	30.40%	92.61%	43.97%
Cov Incr	133.71%	1684.6%	4785.7%	353.7%	28.26%
syns per entry	47.42	5.83	2.9	15.78	3.77
Alts%	38.70%	12.26%	3.29%	29.59%	34.23%
Alts%W	7.90%	1.41%	0.38%	4.63%	21.03%

6.3 Explicit Search Engine Knowledge

The last two rows of Table 4 present the effect of query alterations in our entity synonym discovery. Alterations are popular alternative format of a query (quite often misspellings) that a search engine triggers upon using various mechanisms—usually seen as *results were included for entry*. Alterations can be considered as explicit knowledge a search engine has to influence our entity synonyms creation process. To measure the correctness of such assumption we mined the logs for all such alterations that apply to our entity synonyms. We discovered that a percentage of our entity synonyms can in fact be attributed to alterations (Alts percent in Table 4). For movies, cars and nba athletes such number is high due to popularity of regular words and misspellings. Yet, as we established above, weighted measurements are more important than unweighted when looking at web queries. Hence, we use the weighted alterations ratio (Alts percent W) to take into consideration the sum of the frequency of the alterations over that of the produced entity synonyms. In that case, only a small fraction of the entity synonyms are due to alterations. The low-weighted alteration ratio on multiple data sets illustrates our effectiveness is not due to explicit knowledge the search engine already posses, rather implicit knowledge the users enforce with their clicking behavior.

6.4 Other Approaches to Entity Synonyms

We demonstrated that our approach works in producing entity synonyms and has positive effects. But it is essential to determine how we measure up against other such approaches. For this purpose we looked at using Wikipedia information to produce entity synonyms as well as performing a random walk on the click graph.

For both of these methods we determined that precision related metrics are not good measures of effectiveness—such numbers do not make sense in Wikipedia and were very low for the random walk. So instead we calculated the hit ratio and expansion ratio. By expansion ratio we mean the sum of produced entity synonyms and all original data entries over all original data entries. The results for both Wikipedia and random walk as well as our own approach (Us) with thresholds IPC 4, ICR 0.1 and cleanup 0.05 are summarized in Tables 4 and 5. Detailed discussion follows:

TABLE 5
Expansion

	Original	Hits	Ratio
Movies Us	100	99	99%
Movies Redirection	100	58	58%
Movies Disambiguation	100	38	38%
Movies Walk(0.8)	100	100	100%
Cameras Us	882	767	87%
Cameras Redirection	882	101	11.5%
Cameras Disambiguation	882	0	0%
Cameras Walk(0.8)	882	479	54%

	Original	Entity synonyms	Expansion
Movies Us	100	437	537%
Movies Wikipedia	100	270	370%
Movies Walk(0.8)	100	229	329%
Cameras Us	882	4286	586%
Cameras Wikipedia	882	576	165%
Cameras Walk(0.8)	882	697	179%

6.4.1 Wikipedia

Wikipedia uses redirection and disambiguation pages to capture alternative forms of accessing information. For example, the Wikipedia entry for "LOTR" will send someone to "Lord of The Rings." Such pages are a potential source of entity synonyms. We managed to get 1,091,910 redirection and 36,895 disambiguation entries. We intersected the data sets of cameras and movies with redirection and disambiguation entries and produced entity synonyms.

Wikipedia does well for very popular entries such as some movies, but does poorly for less popular entries (e.g., cameras) that generally do not attract enough user interest to create redirect or disambiguation pages. In any case, our approach still creates more entity synonyms (expansion) and for more entries (hit) for both movies and camera names. We attribute this to the value of query logs that creates more possible entry combinations to consider. Web users are orders of magnitude more than Wikipedia authors and are very creative in forming queries that represent alternative ways to access information. Harvesting that knowledge by our framework produces better results.

6.4.2 Random Walk on a Click Graph

We used the random walk solution described in [15] to evaluate the potential of generating entity synonyms on both movies and cameras data sets. This method takes two parameters: *alpha* (set to 0.01) measures the jump to empty state and *gamma* (set to 0.001) the pruning threshold for the probabilities. For every input data entry we got a list of potential entity synonyms sorted by probability. We kept the ones above 0.8 or the top 50 (whichever was lower) and performed an entity synonyms labeling task using our judges.

As Table 5 shows, expansion ratio is low because, as the number of generated entity synonyms is relatively low. We see in Table 4 that the random walk has a very high hit ratio on movies but has a low hit ratio on cameras. This is because the random walk operates completely on the click graph. So if a query has not been asked then it has no starting point and cannot produce any entity synonyms. For the movies this was not a problem as they were popular enough. But for the cameras this is a problem due to the purely canonical form of camera entries that few people query upon as is—for example, "Sony Cyber-shot DSC-W120/L."

Furthermore, random walk is known to be good at generating related queries, but not entity synonyms as the output from random walk often contains a lot of noise. For example, consider the movie entry "Shrek the Third." Top entries from random walk are "shrek monologues," "shrek 3 the movie in you tube," "apple shreck 3 trailer," "help with the ps2 game shrek the 3rd game" and so forth. Although these are highly related queries, they cannot be considered entity synonyms.

In contrast to the random walk, our approach produces entity synonyms of high precision due to the combination of cleaning process and our ICP/ICR measures which are specifically designed for mining synonyms. In addition, our approach produces entity synonyms far more unique entries (hit ratio) because we take advantage of the search API as appropriate, so even a query has never been asked before, we still get a good number of urls to work with.

6.5 Scalability and Generic Usage

An important test of our solution was to determine how it behaves on large data sets and generic web queries. To this extent we took 2 million popular web queries from the head of our query logs having greater than 15 characters in length. We performed a run using parameters 4, 6, 8, 10 and 0.7, 0.8, 0.85, and 0.9, respectively, without the cleanup phase. Fig. 11 summarizes an analysis between the precision and number of produced entity synonyms. In lower parameter combos (4-0.7) we get a large number of entity synonyms. When looking at more restrictive parameter combos (like 6-0.8) we get precision that is approximately 95 percent and still expand the dictionary by 56 percent. This significant increase in coverage while still maintaining high precision can benefit many structured query understanding works (e.g., [20], [28], [26]).

6.6 Classification of Entity Synonyms

The objective of this set of experiments is to see what fraction of entity synonyms fall into each class, and to show the value of our approach in discovering the most challenging classes of entity synonyms.

Fig. 12 shows a pie chart showing the percentage of entity synonyms in the movies domain (D1) falling into each class listed in Section 5. We combine the Normalization and Spelling classes into a single $S+N$ class. The other classes are Acronym Acr, Containment Con, Superset Sup, Atypical Aty, as well as the composite classes $S+N+Con$ and $S+N+Sup$. Fig. 13 shows the corresponding pie chart for Cameras. From these figures, we make the following observations.

First, a large fraction of entity synonyms, 34 percent for Movies and 43 percent for Cameras, fall into the Aty class. These are the entity synonyms that are not discoverable by approaches relying on string similarity or containment. For the movie "Dark Knight," we discover such entity synonyms as "2008 batman movie," "batman 2," etc. This result shows that our approach of mining click logs results in a significant additional contribution over what can be gained by string similarity alone.

Containment-related classes Con and $S+N+Con$ make up 10 percent of entity synonyms for movies (D1) and 31 percent for cameras (D2). The number is relatively lower for Movies because many movie titles contain relatively common words, and easily become ambiguous if a few words are removed, e.g., "dark" is not a good entity synonym for "Dark Knight." On the other hand, for Cameras, the model number such as "EOS 350D" in

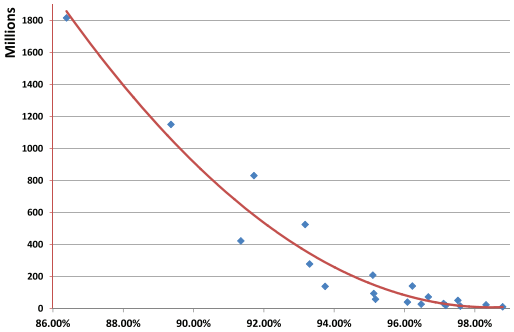

Fig. 11. Precision and number of entity synonyms analysis for 2 m web queries.

"Canon Digital EOS 350D" is a very important identifier, and thus most Con entity synonyms having the model number will be correct.

Superset-related classes Sup and $S+N+Sup$ make up 42 percent for Movies and 19 percent for Cameras. The number is quite high for Movies as there are many entity synonyms that are extensions of movie titles, which include additional words such as the main characters, actors or actresses, or movie year, e.g., "dark knight 2008."

Notably, the $S+N$ class that online approaches can handle are responsible for only 14 percent of entity synonyms for Movies and 7 percent for Cameras. Therefore, relying on online approaches alone would severely limit the coverage of user queries that can be handled by the system. Meanwhile, although Acr is an interesting class that are not easily handled online, it is also extremely rare, responsible for less than 1 percent of entity synonyms.

In summary, our approach not only discovers many more entity synonyms than online approaches can handle, it also generates the various entity synonyms in a unified manner at once, without having to run different methods for different classes of entity synonyms.

7 Related Work

WordNet's [25] *synset* (synonym set) feature provides synonyms for a given word or phrase. WordNet synonyms are fundamentally different from the kind of entity synonyms that we are interested in. WordNet synonyms are for common English words, such as those that would be found in dictionaries and thesauri. Thus, WordNet will not be able to provide synonyms for movie titles, digital camera names, etc. However, WordNet synonyms can still be useful for enhancing recall in information retrieval [21], [32] when

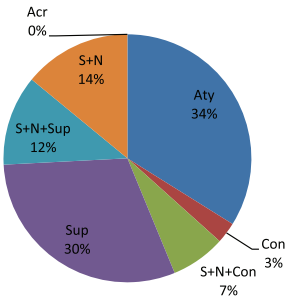

Fig. 12. Classes of entity synonyms for movies.

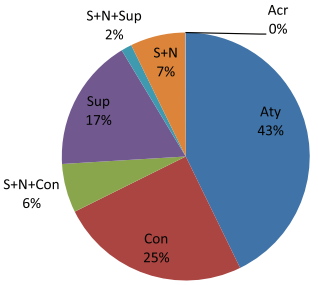

Fig. 13. Classes of entity synonyms for cameras.

properly applied on the query words that are common English words.

Wikipedia is another online resource that has been proposed to help measure the semantic similarity between a pair of words or phrases. One way to do that is by looking at whether the pair of queries tend to retrieve pages that fall under the same Wikipedia categories [29]. Another way is to exploit the redirection relationship between article titles [17]. Unlike WordNet, Wikipedia does cover some entities (movies, musicians, etc.). However, Wikipedia is much smaller than the web, and therefore the former's coverage of entities are severely limited to only a very small number of the most popular ones, as has been shown in Section 6. This limitation motivates us to leverage on the massive data available from the web and query logs.

Chaudhuri et al. [9] is a recent work on finding synonyms by leveraging web search. Their work focuses on a specific kind of synonym, the ones that are substrings of given entity names. Our focus is more generic, as we have showed using the many different classes of entity synonyms in Section 5. Our work relies more on the web user aspect, in exploiting the alternative ways a user may want to refer to an entity. Another work [22] addresses a related problem, which is to turn a query (e.g., "lord rings") into a displayable form (e.g., "Lord of the Rings") by word reordering, modifier addition, and capitalization. In a way, we can see it as a complementary problem, going in the reverse direction from our work (canonical forms into synonyms), but limited only to those synonyms with significant text similarity to canonical form.

Our work is related to reference reconciliation or record matching techniques [11], [13], [7], [5], which try to resolve different references or records in a data set to the same real world entity. They are closely related with our work, since we are trying to find the different references in the form of entity synonyms for a set of given entities. However, there are a few differences from our work. First, they normally assume the "references" are given, while we have to generate the candidate "references" ourselves. Second, such approaches usually rely on multiple attributes to be present to produce high-quality results (e.g., name, age, gender for person record). Yet, web queries normally lack multiattribute semantic context. Third, our framework takes as input a set of homogeneous entity string values, rather than one single entity value. This unique view enables us to perform mining (e.g., noise discovery) across difference input entries for coming up with more meaningful results.

Entity recognition or entity extraction often refers to the problem of identifying mentions of specific entity types (e.g., person, location, etc.) from unstructured data. Doan

et al. [12] and Uren et al. [31] provide a comprehensive overview covering the various information extraction techniques and systems, respectively. However, they are different from the problem we are solving in terms of both input and output. Our input is a list of specific entity values. Our output are the alternative names (what we call entity synonyms) for each input value. Entity recognition normally takes input a description of entity type (e.g., a dictionary, patterns, or a language model) and aiming at identifying the mentions of the entity type from text.

There has been previous work to measure similarity between queries by leveraging web data, for various purposes such as document ranking [10], semantic relation discovery [4], and keyword generation for advertisement [15]. The random walk approach [10], [15] conduct random walk over the query click graph (which is a bipartite graph representation of the Click Data described in Section 2). The similarity between two queries is the probability that a random walk starting at one query reaches the other. A large-scale query-click bipartite graph is studied in [4], to reveal interesting characteristics of the graph and enable analysis over the graph. One of the analysis considers queries resulting in the same set of URLs to be semantically related. The co-occurrence approach [30] considers two queries similar if they co-occur in documents with much greater probability than random chance allows. The co-occurrence counts are derived by issuing queries to a search engine. There are several other measures such as the number of overlapping words and the number of common documents clicked between the pair of queries [33]. These similarity-based approaches do not work well for our problem for several reasons. First, they may discover many pairs of similar or related queries that are not synonyms (e.g., "Windows Vista" and "PC"). Second, the input for which we seek to derive synonyms are generally well-formed strings as full movie titles or digital camera names, which real users seldom use and may not appear frequently as queries. Third, we take a very different perspective of the input, a list of values from a structured data set. This perspective allows us to apply novel techniques such as vertical cleanup to remove noises.

Another application for query similarity is query suggestion, which attempts to provide a search engine user with an alternative to the user's current query. For example, for the ambiguous query "jaguar," the search engine may make query suggestions such as "Jaguar Animal," "Jaguar Cars," "Jaguar Cat," etc. There are several ways to derive these suggestions. The work in [18] is based on typical reformulations or substitutions that users make to their queries. Antonellis et al. [2] further consider whether the query substitutions lead to user clicks on the same ads, while [23] considers whether they lead to clicks on the same webpages. Baeza-Yates et al. [3] also look at the similarity of the clicked pages' text content. As previously explained, the notion of similarity used by query suggestion covers a much broader set of relations (not just synonyms), and thus using these techniques to generate synonyms will lead to many false positives.

Although in this paper, most of our discussion center around the usefulness of entity synonyms for web search, our work will also be of interest to keyword search over structured and semistructured data. The approaches in [1], [6], [16] present systems that allow users to issue keyword queries to relational database management systems (RDBMSs). Since our technique only relies on general click data and search API, we can derive the entity synonyms of RDBMS's entities from such general web data sets, and store them in the RDBMS to improve keyword search over such RDBMS.

8 Conclusion

In this paper, we address the problem of discovering entity synonyms over a structured data set, to bridge the gap between end-user and structured data, and therefore improve structured web search. Our proposed solution mines query logs both to generate entity synonym candidates, as well as to measure the likelihood of a candidate to be actual entity synonym. We also identify several clean up strategies to remove common and context-sensitive noise words, as well as entity hypernym candidates. We further investigate the underlying cause of the formation of entity synonyms, by studying different entity synonym classes. Experiments on large real-life data sets show that our approach yields high precision in terms of entity synonyms generated, increases significantly the coverage of queries that may hit on the relevant entities, and generates more and better entity synonyms than comparable approaches.

As future study, it will be interesting to discover entity hypernyms, hyponyms in addition to entity synonyms. Discovering entity synonyms from other data sets (e.g., tags, social networks) different from query log, as well as from document content using sophisticated natural language processing techniques should also shed light on this problem. It is also intriguing to further study this problem where the input entities are ambiguous, with multiple meanings. Our current technique does not distinguish such entities from unambiguous entities. The multiple meanings naturally lead to multiple entity synonym candidates, and some true candidates may not get enough support to be output as entity synonyms. One interesting idea is to leverage entity disambiguation techniques while finding surrogate webpapges. If the surrogate webpages can be disambiguated into different classes, then we can use our technique to discover entity synonym for each unambiguous class separately.

References

[1] S. Agrawal, S. Chaudhuri, and G. Das, "Dbxplorer: A System for Keyword-Based Search over Relational Databases," *Proc. 18th Int'l Conf. Data Eng. (ICDE)*, 2002.

[2] I. Antonellis, H. Garcia-Molina, and C. Chang, "Simrank++: Query Rewriting through Link Analysis of the Click Graph," *Proc. Int'l Conf. Very Large Databases (VLDB)*, 2008.

[3] R. Baeza-Yates, C. Hurtado, and M. Mendoza, "Query Recommendation Using Query Logs in Search Engines," *EDBT '04: Proc. Int'l Conf. Current Trends in Database Technology*, 2004.

[4] R. Baeza-Yates and A. Tiberi, "Extracting Semantic Relations from Query Logs," *Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, 2007.

[5] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S.E. Whang, and J. Widom, "Swoosh: A Generic Approach to Entity Resolution," *The VLDB J.*, vol. 18, pp. 255-276, 2009.

[6] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword Searching and Browsing in Databases Using Banks," *Proc. 18th Int'l Conf. Data Eng. (ICDE),* 2002.
[7] I. Bhattacharya and L. Getoor, "Collective Entity Resolution in Relational Data," *ACM Trans. Knowledge Discovery from Data,* vol. 1, pp. 1-36, 2007.
[8] Box Office, 2007 Yearly Box Office Results, http://www.boxofficemojo.com/yearly/chart/?yr=2007, 2007.
[9] S. Chaudhuri, V. Ganti, and D. Xin, "Exploiting Web Search to Generate Synonyms for Entities," *Proc. 18th Int'l Conf. World Wide Web (WWW),* 2009.
[10] N. Craswell and M. Szummer, "Random Walks on the Click Graph," *Proc. 30th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 239-246, 2007.
[11] D. Dey, S. Sarkar, and P. De, "A Distance-Based Approach to Entity Reconciliation in Heterogeneous Databases," *IEEE Trans. Knowledge and Data Eng.,* vol. 14, no. 3, pp. 567-582, May/June 2002.
[12] A. Doan, R. Ramakrishnan, and S. Vaithyanathan, "Managing Information Extraction: State of the Art and Research Directions," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* 2006.
[13] X. Dong, A. Halevy, and J. Madhavan, "Reference Reconciliation in Complex Information Spaces," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* 2005.
[14] R. Fagin, B. Kimelfeld, Y. Li, S. Raghavan, and S. Vaithyanathan, "Understanding Queries in a Search Database System," *Proc. 29th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS),* 2010.
[15] A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal, "Using the Wisdom of the Crowds for Keyword Generation," *Proc. Int'l Conf. World Wide Web (WWW),* pp. 61-70, 2008.
[16] V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient Ir-Style Keyword Search over Relational Databases," *Proc. 29th Int'l Conf. Very Large Data Bases (VLDB),* 2003.
[17] J. Hu, L. Fang, Y. Cao, H.-J. Zeng, H. Li, Q. Yang, and Z. Chen, "Enhancing Text Clustering by Leveraging Wikipedia Semantics," *Proc. 31st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* 2008.
[18] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating Query Substitutions," *Proc. 15th Int'l Conf. World Wide Web (WWW),* pp. 387-396, 2006.
[19] X. Li, Y.-Y. Wang, and A. Acero, "Extracting Structured Information from User Queries with Semi-Supervised Conditional Random Fields," *Proc. 32nd Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* 2009.
[20] F. Liu, C.T. Yu, W. Meng, and A. Chowdhury, "Effective Keyword Search in Relational Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* 2006.
[21] S. Liu, F. Liu, C. Yu, and W. Meng, "An Effective Approach to Document Retrieval via Utilizing Wordnet and Recognizing Phrases," *Proc. 27th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* 2004.
[22] A. Malekian, C.-C. Chang, R. Kumar, and G. Wang, "Optimizing Query Rewrites for Keyword-Based Advertising," *Proc. Ninth ACM Conf. Electronic Commerce (EC),* 2008.
[23] Q. Mei, D. Zhou, and K. Church, "Query Suggestion using Hitting Time," *Proc. 17th ACM Conf. Information and Knowledge Management (CIKM),* 2008.
[24] Microsoft, MSN Shopping XML Data Access API, http://shopping.msn.com/xml/v1/getresults.aspx?text=digital+camera, 2012.
[25] G.A. Miller, "Wordnet: A Lexical Database for English," *Comm. ACM,* vol 38, no. 11, pp. 39-41, 1995.
[26] A. Nandi and H.V. Jagadish, "Qunits: Queried Units in Database Search," *Proc. Conf. Innovative Data Systems Research (CIDR),* 2009.
[27] S. Paparizos, A. Ntoulas, J.C. Shafer, and R. Agrawal, "Answering Web Queries Using Structured Data Sources," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* 2009.
[28] N. Sarkas, S. Paparizos, and P. Tsaparas, "Structured Annotations of Web Queries," *Proc. 35th ACM SIGMOD Int'l Conf. Management of Data,* 2010.
[29] M. Strube and S.P. Ponzetto, "Wikirelate! Computing Semantic Relatedness Using Wikipedia," *Proc. 21st Nat'l Conf. Artificial Intelligence,* 2006.
[30] P.D. Turney, "Mining the Web for Synonyms: PMI-IR Versus LSA on TOEFL," *Proc. 12th European Conf. Machine Learning (EMCL),* 2001.
[31] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna, "Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art," *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 4, pp. 14-28, 2006.
[32] G. Varelas, E. Voutsakis, P. Raftopoulou, E.G. Petrakis, and E.E. Milios, "Semantic Similarity Methods in Wordnet and Their Application to Information Retrieval on the Web," *Proc. Seventh Ann. ACM Int'l Workshop Web Information and Data Management (WIDM),* 2005.
[33] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "Clustering User Queries of a Search Engine," *Proc. 10th Int'l Conf. World Wide Web (WWW),* pp. 162-168, 2001.

Tao Cheng received the PhD degree from the University of Illinois at Urbana-Champaign. He has been a researcher at Microsoft Research in Redmond, Washington, since 2010. His doctoral thesis focuses on proposing and developing a novel search paradigm—Entity Search. His work along this line of research is well recognized, with publications in top database and web search conferences, including SIGMOD, VLDB, ICDE, WSDM, EDBT, and CIDR.

Hady W. Lauw received the PhD degree from Nanyang Technological University, in the area of social network mining from user-generated content. He is a researcher at A*STAR's Institute for Infocomm Research in Singapore. He is also an adjunct assistant professor of information systems at Singapore Management University. Previously, he was a postdoctoral researcher at Search Labs, Microsoft Research. He was also a recipient of several fellowship awards, including A*STAR Graduate Scholarship for his PhD studies. His research interests are in the areas of social network mining and web search/mining.

Stelios Paparizos received the PhD degree from the University of Michigan on XML query processing and optimization where he was a key member of designing and building the Timber native XML database. He has been a researcher at Microsoft Research in Silicon Valley, California, since 2006. His current work revolves around web data management and web mining and their effect on improving web search. His work has found its way in the Bing search engine and significant components have been recognized with three Microsoft Tech Transfer awards. For his graduate studies, he received numerous scholarships including the Fulbright, Bodossakis, and Gerondelis Foundation awards.