Synonym Discovery for Structured Entities on Heterogeneous Graphs

Xiang Ren*
University of Illinois at Urbana-Champaign
Urbana, IL, USA
xren7@illinois.edu

Tao Cheng
Microsoft Research
Redmond, WA, USA
taocheng@microsoft.com

ABSTRACT

With the increasing use of entities in serving people's daily information needs, recognizing synonyms—different ways people refer to the same entity—has become a crucial task for many entity–leveraging applications. Previous works often take a "literal" view of the entity, *i.e.*, its string name. In this work, we propose adopting a "structured" view of each entity by considering not only its string name, but also other important structured attributes. Unlike existing query log-based methods, we delve deeper to explore sub-queries, and exploit tailed synonyms and tailed web pages for harvesting more synonyms. A general, heterogeneous graph-based data model which encodes our problem insights is designed by capturing three key concepts (synonym candidate, web page and keyword) and different types of interactions between them. We cast the synonym discovery problem into a graph-based ranking problem and demonstrate the existence of a closed-form optimal solution for outputting entity synonym scores. Experiments on several real-life domains demonstrate the effectiveness of our proposed method.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—Data mining

Keywords

Synonym Discovery; Structured Entity; Heterogeneous Graph;

1. INTRODUCTION

Entities are becoming increasingly important and prevalent in people's daily information quest. One major hurdle in entity understanding is that content creators and search users often use a variety of alternate names, *i.e.*, entity synonyms, to reference the same entity. For instance, "*Kobe Bryant*" is also referred to as "*black mamba*" or "*lakers24*". Entity synonym discovery is useful for a rich set of applications (e.g., vertical search, web search), as it helps boost recall, improves precision and enhances user experience.

*Work was partially done while author was visiting MSR.

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2745396.

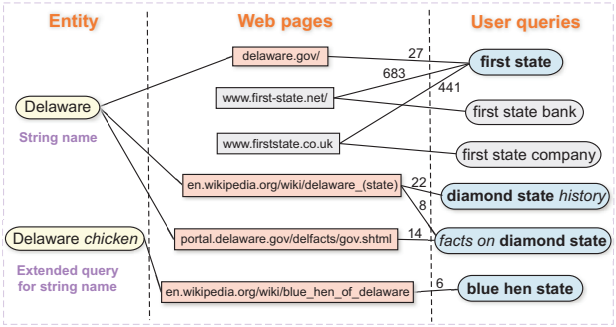


Figure 1: Motivating example.

One way to gather entity synonym information is to leverage existing synonyms in entity knowledge bases. For example, in Freebase, entity synonym lists (called *aliases*) are manually created for some of the entities, and in Wikipedia, entity synonyms can be obtained based on redirect pages and disambiguation pages. However, since most of such information is manually curated, synonyms so obtained typically suffer from limited coverage and diversity.

A recent trend avoids expensive human laboring and increases coverage and diversity of synonyms by automatically discovering entity synonyms from query logs [7, 5, 21], where web queries contain focused and succinct information about entities. The main insight here is that by tapping into information provided by both content creators and search users, one can more effectively discover the synonyms. For example, Fig. 1 highlights the main idea behind these query log-based approaches: from entity name "*Delaware*", get its clicked web pages which lead to queries clicking on these pages, and then perform analysis on these queries to mine the synonyms. One can mine entity synonyms by analyzing different kinds of information in the query log, such as query click-through data [7], query context [5], pseudo-document for web page [5] and their combinations [21]. However, existing query log-based methods encounter two major limitations as follows:

• **Ambiguous Entity Names and Synonyms:** Most existing methods take only the entity's string name as input, and thus cannot handle the *name ambiguity* issue. For example, "*jr smith*" may refer to either the NBA player *JR Smith* or the *JrSmith Manufacturing Company*. Further, many synonyms can also be ambiguous. In Fig. 1, as a synonym of *Delaware*, "*first state*" can also refer to other entities like First State Bank and First State Investments, which are in fact two more popular meanings in terms of user clicks. Both click-based [7] and context-based [5] similarities will fail to discover such synonyms due to weak click statistics and noisy contexts. Dealing with ambiguity is crucial for discovering high quality entity synonyms.

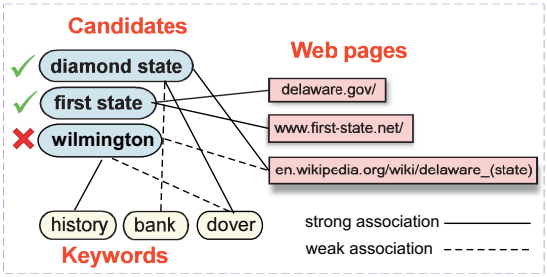


Figure 2: Example candidates, pages, keywords.

• **Sub-query Synonyms:** All existing works use web queries as a whole to generate synonym candidates but ignore cases when target synonyms appear only as *sub-queries* in query log. Such cases are common since some synonyms are tailed synonyms (*e.g.*, "*diamond state*" in Fig. 1), which express tailed meanings of the synonym strings. Such tailed synonyms have difficulty in returning relevant web pages for the entity (*i.e.*, *Delaware*) by themselves. Additional keywords are typically added (*e.g.*, "*history*") for clarifying the search intent. It is challenging to recognize such sub-queries as candidates and identify true synonyms.

To tackle the *ambiguity issue*, we propose a novel problem definition for entity synonym discovery, by taking a structured entity instead of only an entity name as input. This input paradigm allows us to take advantage of structured attributes of an entity in addition to its name. These structured attributes help crisply define an entity and therefore dramatically reduce ambiguity. This in turn also facilitates in finding tailed synonyms, since we understand the input entity better. It leads to the possibility of discovering high quality entity synonyms with good coverage. There exist a variety of attributes for a structured entity in either knowledge bases or ad-hoc entity collections. We focus on two interesting and generally available attributes, *i.e.*, entity source web pages and existing synonyms in this work, and discuss in Sec. 5 how to generalize the proposed framework to incorporate other structured attributes.

To exploit *sub-query synonyms*, we leverage holistic query log statistics to extract synonym candidates by identifying salient sub-queries. The main insight here is that the words in a legitimate candidate would naturally appear together (rather than by accident) in a good number of queries. More specifically, we identify synonym candidates by examining word collocation for n-grams in queries.

While generating candidates by exploring sub-queries, we identify the keywords that co-occur with the candidates from queries. These extracted keywords, along with the web pages clicked by the candidates, provide highly valuable information for us to identify true synonyms. As illustrated in Fig. 2, it is desirable for descriptive web pages (e.g., delaware.gov) or keywords (e.g., dover) to be strongly associated (indicated by solid link) with true synonyms (e.g., diamond state). Meanwhile, false synonym candidates (e.g., wilmington) would have weak relationships (indicated by dashed link) with such pages or keywords. Sec. 3 discusses in detail the relationships between these objects.

Given these heterogenous objects as well as the relationships between them, it is crucial to capture them in a principled way. Rather than studying these objects and relationships in silo, we believe it is much more effective to model them holistically, so that two objects can influence each other even though they are not directly related. To this end, we propose a heterogeneous graph-based framework, called **StrucSyn**, to discover entity synonyms, by exploring the interplays between these three types of objects, *i.e.*, synonym candidates, keywords and web pages. It is through the subtle interactions via the relationships between these objects and holistic graph prorogation that we are able to glean signals for mining *tailed synonyms* (see "*diamond state*" in Fig. 1). We further extend our synonym candidate generation by expanding the seed query set (*e.g.*, "*delaware chicken*" in Fig. 1) to retrieve *tailed web pages* for enhancing the coverage of discovered synonyms, which we discuss in more details in Section 3.5.

Specifically, a *signed heterogeneous graph* is constructed to represent all the available information in a unified form, which encodes candidate-page, candidate-keyword, keyword-page and candidate-candidate relations. The relation strength connecting two objects indicates how much impact they have on each other when deriving entity synonyms. Synonym discovery is then cast into a heterogeneous graph-based ranking problem, which takes entity name, source web pages and existing synonyms as labels, and tries to rank each type of objects by preserving the graph structure. We propose a convex optimization problem for learning the ranking scores of all types of objects *jointly*. We then derive a *globally optimal*, closed-form solution. An iterative algorithm is further designed to solve the optimization problem efficiently while preserving the global optimum.

Our experiments on five entity sets of different real-life domains demonstrate the power of the proposed method. In comparing the output as a ranked list, the proposed method achieves 53.49% improvement over the best performing compared method in Precision@10. For the study of enriching entity knowledge base where automatic cut-off is applied, our method is able to output on average 10.35 new synonyms with 81.67% Precision for Freebase. Our case study results show that the proposed method can discover many semantic synonyms beside spelling variants which nicely complements the existing synonyms in entity knowledge bases.

The rest of paper is organized as follows. Sec. 2 provides problem definition and details of candidate generation. Sec. 3 introduces our hypotheses and data model. Sec. 4 proposes our synonym discovery method on the constructed heterogeneous graph. We discuss how to generalize our model in Sec. 5. We provide experimental results and analysis in Sec. 6 and discuss the related work in Sec. 7.

2. PROBLEM DEFINITION AND CANDIDATE GENERATION

2.1 Problem Definition

In our problem setting, a *structured entity*, instead of only entity's string name [7, 5, 21], is taken as input, which consists of its string name and other structured attributes of the entity, *e.g.*, entity type, entity description.

Many entity attributes are generally available across different domains. However, in this work, we mainly focus on studying two types of structured attributes, *i.e.*, entity source web pages and existing synonyms. They are generally available in entity knowledge bases and are domain independent. By leveraging source web pages, which are less ambiguous than the entity name, one can generate better quality candidates and avoid semantic drift, compared to using only the entity name. Similarly, multiple existing synonyms can help consolidate contexts for query entity, and boost tailed synonyms and tailed web pages. Generalization of the framework to incorporate other kinds of structured attributes will be discussed in Sec. 5. Specifically, we define structured entity as follows:

DEFINITION 1 (STRUCTURED ENTITY). *A structured entity, e, has a reference name r_e, a set of source web pages*

Table 1: Structured attributes for entity *Delaware*.

Attributes	Values
Reference name	delaware
Source web pages	http://www.delaware.gov http://en.wikipedia.org/wiki/Delaware
Existing synonyms	DE; Del; Dela

$\mathcal{U}_e = \{u_1, ..., u_{|\mathcal{U}_e|}\}$ *which deliver focused information about the entity, and a set of existing synonyms* $\mathcal{C}_e = \{c_1, ..., c_{|\mathcal{C}_e|}\}$.

Table 1 shows these structured attributes and their values for example entity *Delaware*. Let $\mathcal{S}$ be the universal set of strings, where each string is a sequence of one or more words. Given a structured entity e, its synonyms are strings $s \in \mathcal{S}$ that are used to reference e, including semantic alternations, abbreviations, acronyms and spelling variants. For example, synonyms for NBA player *Kobe Bryant* can be "*black mamba*", '*kobe*', "*KB*" and "*coby*". Formal definition of entity synonym is given as follows:

DEFINITION 2 (ENTITY SYNONYM). *A synonym for entity* e, $s \in \mathcal{S}$, *is a sequence of one or more words, which not only is an entity mention, but also can serve as an alternative name to refer to the entity* e.

We require an entity synonym to be an entity mention so that a string of arbitrary length which expresses equivalent meaning as the entity does will not be considered as synonym (*e.g.*, "*the No.24 player in LA Lakers*" is not a synonym of *Kobe Bryant*). In particular, our definition of entity synonym does not require the entity reference names or synonym strings to be *unambiguous*, which is less restricted than the definitions in previous work [7, 5, 21], and thus can lead to higher coverage of discovered synonyms. We use an entity synonym score function, $f_{\mathcal{C}}(e, s) : \mathcal{E} \times \mathcal{S} \mapsto \mathbb{R}$, to measure how likely a string s is a true synonym for e.

In this work, we aim to harvest entity synonyms from web queries, by analyzing user click-through data. Specifically, user click-through data $\mathcal{L}$ is a set of tuples $l = (q, u, m)$, where q is query, u is web page and m is the number of times users have clicked on web page u when issuing query q. With the definitions of structured entity, entity synonym and query log, we provide a formal definition for **Structured Entity Synonym Discovery** as follows:

DEFINITION 3 (PROBLEM DEFINITION). *Given a structured entity* $e = (r_e, \mathcal{U}_e, \mathcal{C}_e)$ *and click-through data* $\mathcal{L}$, *the problem of synonym discovery for structured entity aims to (1) identify a subset of candidate synonyms* $\mathcal{C} = \{c_1, ..., c_{|\mathcal{C}|}\} \subset \mathcal{S}$; *(2) derive entity synonym scores* $f_{\mathcal{C}}(e, c)$ *for* $c \in \mathcal{C}$; *and (3) select a subset of* K_e *candidates with highest* $f_{\mathcal{C}}$ *from* $\mathcal{C}$, *as the suggested entity synonyms for* e.

2.2 Candidate Generation

We now discuss our candidate generation strategy, which uses web queries in $\mathcal{L}$ to derive a set of candidate synonyms $\mathcal{C} \subset \mathcal{S}$. The key difference from existing work is that we delve deeper to explore sub-queries in candidate generation.

Most existing work use co-click queries, *i.e.*, queries which share clicked web pages with the entity name query, as synonym candidates, based on the assumption that true entity synonyms always appear as web queries themselves. Such process ignores sub-query synonyms (*e.g.*, "*diamond state*" in Fig. 1), and thus hurt coverage and diversity of discovered synonyms. In practice, over 71% of web queries contain entities, and only less than 1% of these entity queries contain two or more entities as shown in [15]. Therefore, we propose to extract the most salient unit, *i.e.*, entity mention, from each co-click query to generate synonym candidate (see Fig. 3), based on the following assumption.

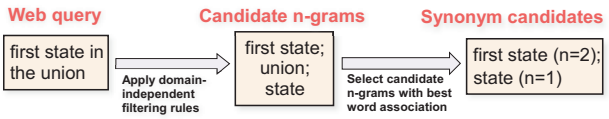


Figure 3: Synonym candidate generation.

ASSUMPTION 1 (SINGLE ENTITY QUERY). *There is only one single entity mention per user query. Each web query is either an entity mention itself, or consists of an entity mention and peripheral context words.*

We start with extracting n-grams from co-click queries for query entity. Given an entity e, let $N_{\mathcal{U}}(e)$ denote the set of web pages which users click when issuing entity name r_e as query. We use $N_{\mathcal{Q}}(u)$ to denote the set of queries which users click on web page u in click-through data. Co-click queries for e, denoted as $\mathcal{Q}$, can then be collected by obtaining $N_{\mathcal{Q}}(u)$ for each $u \in N_{\mathcal{U}}(e)$, *i.e.*, $\mathcal{Q} = \bigcup_{u \in N_{\mathcal{U}}(e)} N_{\mathcal{Q}}(u)$.

From each candidate query $q \in \mathcal{Q}$, we extract n-grams[1] $N_{\mathcal{C}}(q)$, *i.e.*, word sequences of different lengths n, that does not start or end with numbers and stopwords, leading to a set of candidate n-grams $\mathcal{N} = \bigcup_{q \in \mathcal{Q}} N_{\mathcal{C}}(q)$ for e. Such domain-independent filtering rules can help remove many noisy n-grams that are likely not entity mentions, such as "*where is kobe*" and "*kobe bryant from*".

We further apply n-gram testing techniques [30, 10, 11] to select from $\mathcal{N}$ the n-grams that have best word collocation for each $n \geq 2$, in order to obtain more salient n-grams as candidates. Compared to NLP techniques such as chunking and dependency parsing, n-gram testing methods do not rely on any model training and are domain-independent. Specifically, we extend Pointwise Mutual Information (PMI) [30] to measure the collocation for n-grams in $\mathcal{N}$. Let $N_{\mathcal{C}}^{(n)}(q)$ denote length-n candidate n-grams in $N_{\mathcal{C}}(q)$, the collocation score for candidate n-gram $c = [w_1, w_2, ..., w_n] \in N_{\mathcal{C}}^{(n)}(q)$ is defined as $g^{(n)}(c) = \log_2 \frac{p(w_1,...,w_n)}{p(w_1)\cdots p(w_n)}$ where $p(s)$ denotes the probability of seeing string s in $\mathcal{Q}$, *i.e.*, $p(s) = |N_{\mathcal{Q}}(s)|/|\mathcal{Q}|$. We then can select candidate n-gram $c \in N_{\mathcal{C}}^{(n)}(q)$ with highest score $g^{(n)}(c)$ to get a length-n synonym candidate $c^{(n)}$, *i.e.*, $c^{(n)} = \text{argmax}_{c \in N_{\mathcal{C}}^{(n)}(q)} g^{(n)}(c)$ (*e.g.*, length-2 candidate "*first state*" in Fig. 3). For unigrams, we simply set $g^{(1)}(c) = p(c)$ and select the one with highest $p(c)$. After above steps, we can collect a set of synonym candidates, $\mathcal{C} = \bigcup_{q \in \mathcal{Q}} \bigcup_{i=1}^{n} \text{argmax}_{c \in N_{\mathcal{C}}^{(i)}(q)} g^{(n)}(c)$, which are salient in terms of word collocation and thus are more likely entity mentions.

3. DATA MODEL

3.1 Framework Overview

Candidate generation in Sec. 2 leads us to two types of information sources that are closely related with the extracted candidates, namely the web pages which the candidates' support queries click on, and the keywords appearing along side with the candidates in the support queries. The clicked web pages offer descriptive information about entity and the keywords act as useful contextual information about entity. However, existing methods judge the importance of a web page for entity merely by click counts from the entity names, and treat all keywords which come along with entity name equally important for the entity. In practice, important keywords and representative web pages help identify sub-query synonyms, boost tailed synonyms, and add robustness in handling noisy click-through data. Therefore, we propose to identify and quantify web pages from click-through data and

[1]In this work we focus on unigram, bigram and trigram.

Table 2: Top-5 entity contexts and entity pages of two example entities derived by StrucSyn.

Entity	Context	Web page
JR Smith	tattoos	en.wikipedia.org/wiki/J._R._Smith
	dunk	twitter.com/jr_swish
	young	www.nba.com/playerfile/jr_smith
	nba	www.imdb.com/name/nm0808646/bio
	basketball	sports.yahoo.com/nba/players/3835
CBS	owners	en.wikipedia.org/wiki/CBS
	chairman	cbs.com
	sports	cbscorporation.com
	news	forbes.com/companies/cbs
	founder	youtube.com/user/CBS

context words from web queries, which can provide focused entity information, in a unified framework.

Intuitively, a candidate is more likely an entity synonym, if it is closely related to web pages which deliver focused information about the entity.

Definition 4 (Entity Page). *Entity page, u, of an entity e, is a surrogate web page for e, which is not only relevant to e but also representative for e.*

Wiki page of *Delaware*, for example, is a high quality entity page. In contrast, although relevant to entity, a news page or lottery page is not entity page since it is either related to multiple entities or merely focuses on a certain aspect of the entity. To model these properties, we use an entity page score function, $f_{\mathcal{U}}(e,u):\mathcal{E}\times\mathcal{U}\mapsto\mathbb{R}$, to measure how likely a web page u is a true entity page for entity e.

Similarly, if a candidate frequently co-occurs with the context keywords which are important for the entity, it is more likely an entity synonym. We define the concept of entity context for keywords in queries as follow.

Definition 5 (Entity Context). *Entity context, w, is a keyword in the context of an entity e in queries, which is not only relevant to e but also exclusive to e.*

Example entity contexts for *Kobe Bryant* include "*lakers*" and "*24*". Relevant keywords such as "*stats*" and "*team*" are widely related to many entities in sports domain, and thus are not exclusively relevant to *Kobe Bryant*. To model entity context in queries, we use an entity context score function, $f_{\mathcal{W}}(e,w):\mathcal{E}\times\mathcal{W}\mapsto\mathbb{R}$, to measure how likely a keyword w is a true entity context for an entity e.

In order to capture the characteristics of synonym candidate, web page and keyword, as well as the scores associated with them, we propose to use heterogeneous graph model as the underlying data model. A heterogeneous graph model captures not only different types of objects but also different types of interactions between these objects. The basic idea to construct the graph is that two objects are linked with large weight if they play important roles in deciding scores for each other. A graph-based ranking problem can be formulated on the constructed graph such that information of the entity is used as positive labels and ranking scores are propagated based on the graph structure (Sec. 4). By solving the ranking problem, we can jointly derive entity synonym scores, entity context scores and entity pages scores for corresponding objects in the graph.

Table 2 shows top-5 keywords and web pages ranked in terms of entity context score and entity page score by our method. For entity contexts, top ranked keywords are not only relevant to entities but also exclusive, *e.g.*, "*tattoos*" vs. "*basketball*" for *JR smith*. Similarly, top ranked web pages for CBS are both relevant and representative for the entity, instead of focusing on a certain aspect like CBS news.

Specifically, given a structured entity e and click-through data $\mathcal{L}$, we can construct a heterogeneous graph G which

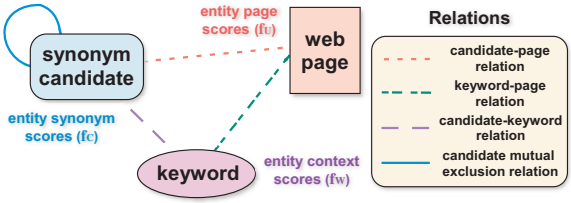


Figure 4: Heterogeneous graph illustration.

consists of three types of objects: synonym candidates $\mathcal{C}=\{c_1,...,c_{|\mathcal{C}|}\}$, keywords[2] $\mathcal{W}=\{w_1,...,w_{|\mathcal{W}|}\}$ and web pages $\mathcal{U}=\{u_1,...,u_{|\mathcal{U}|}\}$ which are clicked by users when issuing queries $\mathcal{Q}$. Naturally, there are three types of relations between these three objects including candidate-page relation, keyword-page relation and candidate-keyword relation. In addition, given there may be multiple candidates generated from one query, we further explore the mutual exclusion relationship between these candidates to help infer correct entity boundary. Together, these four types of relationship lead to four subgraphs $G^{(\mathcal{CU})}$, $G^{(\mathcal{WU})}$, $G^{(\mathcal{CW})}$ and $G^{(\mathcal{C})}$, respectively. We provide an illustration of the heterogeneous graph in Fig. 4 and now introduce its subgraph details.

3.2 Candidate-page and Keyword-page Subgraphs

Intuitively, if users mainly click on web page u after issuing query q, then u is likely relevant to information needs behind q. Such idea can be extended to measure semantic relatedness between synonym candidates (keywords) and web pages, by aggregating click-through information with respect to support queries of candidates (keywords). Ideally, if most support queries of a candidate (keyword) are related to web page u, then the candidate (keyword) should be related to web page u as well. We use the above observation on interplay between candidate (keyword) and web page to model their respective desired properties.

Hypothesis 1 (Click-Based Relevance). *(1) A candidate (keyword) is more likely an entity synonym (context) if its primarily related web pages are entity pages; (2) A web page is more likely an entity page if its primarily related candidates and keywords are entity synonyms and contexts.*

Specifically, relation strength between a synonym candidate c and a web page u can be approximately represented by the average query-page relation strength over support queries of c, *i.e.*, $N_{\mathcal{Q}}(c)$. We use $\mathbf{W}^{(\mathcal{CU})}\in\mathbb{R}^{|\mathcal{C}|\times|\mathcal{U}|}$ to denote the bi-adjacency matrix for $G^{(\mathcal{CU})}$, where $W_{ij}^{(\mathcal{CU})}$ is the edge weight between c_i and u_j which reflects the relation strength, and is defined as follows:

$$W_{ij}^{(\mathcal{CU})}=\frac{1}{|N_{\mathcal{Q}}(c_i)|}\cdot\sum_{q\in N_{\mathcal{Q}}(c_i)}\frac{m(q,u_j)}{\sum_{u'\in\mathcal{U}}m(q,u')}.$$

Here, $m(q,u)$ is the click count between q and u.

For candidate-keyword relationship, we can similarly aggregate the query-page relationships of keyword's support queries, and define edge weight for $G^{(\mathcal{WU})}$ as follows.

$$W_{ij}^{(\mathcal{WU})}=\frac{1}{|N_{\mathcal{Q}}(w_i)|}\cdot\sum_{q\in N_{\mathcal{Q}}(w_i)}\frac{m(q,u_j)}{\sum_{u'\in\mathcal{U}}m(q,u')},$$

where $N_{\mathcal{Q}}(w)$ denote the support queries of keyword w and $\mathbf{W}^{(\mathcal{WU})}\in\mathbb{R}^{|\mathcal{W}|\times|\mathcal{U}|}$ is the bi-adjacency matrix for $G^{(\mathcal{WU})}$.

In practice, it is desirable to discount the edge weights between two objects by their node degrees, in order to help reduce the impact of popularity of objects. For example,

[2]We extract all non-stopword unigrams from queries $\mathcal{Q}$.

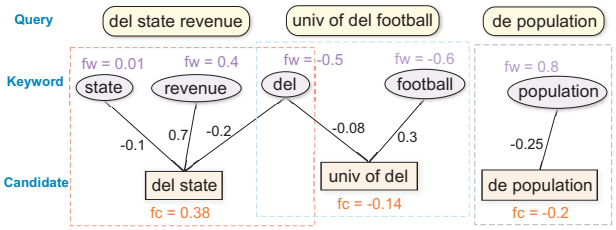


Figure 5: Candidate-keyword relation example.

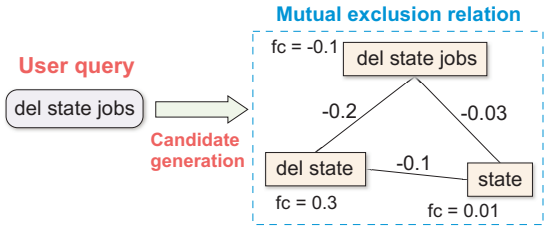


Figure 6: Candidate mutual exclusion example.

web pages that have high exposure, *i.e.*, clicked by many queries, will have high degrees and thus moderate normalized edge weights in turn. Thus, we define the normalized adjacency matrices $\mathbf{S}^{(\mathcal{CU})} = \mathbf{D}^{(\mathcal{CU})-1/2}\mathbf{W}^{(\mathcal{CU})}\mathbf{D}^{(\mathcal{UC})-1/2}$ and $\mathbf{S}^{(\mathcal{WU})} = \mathbf{D}^{(\mathcal{WU})-1/2}\mathbf{W}^{(\mathcal{WU})}\mathbf{D}^{(\mathcal{UW})-1/2}$. Here, $\mathbf{D}^{(\mathcal{CU})} \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$ and $\mathbf{D}^{(\mathcal{UC})} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{U}|}$ denote the diagonal degree matrices for candidates and web pages in $G^{(\mathcal{CU})}$, respectively, where $D^{(\mathcal{CU})}_{ii} = \sum_{j=1}^{|\mathcal{U}|} W^{(\mathcal{CU})}_{ij}$ and $D^{(\mathcal{UC})}_{jj} = \sum_{i=1}^{|\mathcal{C}|} W^{(\mathcal{CU})}_{ij}$. Similarly, we define $\mathbf{D}^{(\mathcal{WU})} \in \mathbb{R}^{|\mathcal{W}|\times|\mathcal{W}|}$ and $\mathbf{D}^{(\mathcal{UW})} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{U}|}$ for $G^{(\mathcal{WU})}$. One can check that $\mathbf{D}^{(\mathcal{CU})} = \mathbf{I}_{|\mathcal{C}|}$ and $\mathbf{D}^{(\mathcal{WU})} = \mathbf{I}_{|\mathcal{W}|}$.

3.3 Candidate-keyword Subgraph

In general, co-occurrence of keyword and candidate across support queries can provide useful information on judging whether the candidate is entity synonym, and deciding whether the keyword is entity context. In Fig. 5, "*del state*" is more likely a synonym of *Delaware*, compared to "*univ of del*", since it is surrounded by a keyword with higher entity context score "*revenue*" (vs. "*football*") for *Delaware*. On the other hand, "de population" is probably not a synonym for *Delaware* since it contains a keyword "*population*" which is likely an entity context. Intuitively, we can capture relations between candidates and keywords, and model properties for entity synonyms and contexts, by following the query-based co-occurrence hypothesis.

HYPOTHESIS 2 (QUERY-BASED CO-OCCURRENCE). *(1) If a candidate (keyword) often appears around a entity context (synonym) in support queries, then it is more likely an entity synonym (context); (2) If a candidate (keyword) often contains (appears in) an entity context (synonym), it is probably not an entity synonym (context).*

We propose to construct candidate-keyword subgraph $G^{(\mathcal{CW})}$, which is a signed bipartite graph, to capture two different types of co-occurrence relations: (1) "contains" (negative) relation and (2) "appears around" (positive) relation. A "contains" relationship between c and w indicates that c contains w whereas an "appears around" relationship between them indicates that w appears around c in at least one query.

Formally, we use $\mathbf{W}^{(\mathcal{CW})} \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{W}|}$ to denote the signed bi-adjacency matrix for $G^{(\mathcal{CW})}$. We define the indicator function $I_C(c,w)$ which has value 1 if c contains w and 0 otherwise, and $I_A(c,w;q)$ which returns 1 if w appears around c in q and 0 otherwise. The edge weight between c_i and w_j can be defined as follows:

$$W^{(\mathcal{CW})}_{ij} = \sum_{q\in\{N_{\mathcal{Q}}(c_i)\cap N_{\mathcal{Q}}(w_j)\}} \Big(I_A(c_i,w_j;q) - I_C(c_i,w_j)\Big).$$

To discount popular candidates and keywords, we normalize $\mathbf{W}^{(\mathcal{CW})}$ into $\mathbf{S}^{(\mathcal{CW})} = \mathbf{D}^{(\mathcal{CW})-1/2}\mathbf{W}^{(\mathcal{CW})}\mathbf{D}^{(\mathcal{WC})-1/2}$ by a similar idea as the PMI score, where $\mathbf{D}^{(\mathcal{CW})} \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$ and $\mathbf{D}^{(\mathcal{WC})} \in \mathbb{R}^{|\mathcal{W}|\times|\mathcal{W}|}$ are degree matrices of the signed graph with $D^{(\mathcal{CW})}_{ii} = \sum_{j=1}^{|\mathcal{W}|} |W^{(\mathcal{CW})}_{ij}|$ and $D^{(\mathcal{WC})}_{jj} = \sum_{i=1}^{|\mathcal{C}|} |W^{(\mathcal{CW})}_{ij}|$.

With this subgraph, if strong "contains" (negative) relationship exists between an entity synonym and a keyword, the keyword is less likely an entity context. On the other hand, a keyword is more likely an entity context if one observe strong "appear around" (positive) relationship between an entity synonym and the keyword. Similar interactions exist between an entity context and a candidate as well.

3.4 Candidate Mutual Exclusion Subgraph

In the candidate generation step, multiple candidates (of different lengths) may be generated from one query (see Fig. 3), which results in conflict with the single entity assumption in Sec. 2.2. Fig. 6 shows that "*state*", "*del state*", and "*del state jobs*" have strong mutual exclusion relationships with each other, since they are frequently extracted from the same queries as synonym candidates. To assist modeling of entity mention property, we propose to exploit the mutual exclusion relationship between conflicting candidates in our data model, based on the following hypothesis.

HYPOTHESIS 3 (CANDIDATE MUTUAL EXCLUSION). *If two candidates tend to co-occur across many different queries, and if one of them is an entity synonym, then the other is less likely an entity synonym.*

Specifically, we construct a candidate mutual exclusion subgraph, $G^{(\mathcal{C})}$, to capture conflict co-occurrences in queries between candidates. With the proposed hypothesis, if there is a strong mutual exclusion relationship between two candidates, and one of them is likely an entity synonym, then the other one tends to have low entity synonym score. An affinity matrix $\mathbf{W}^{(\mathcal{C})} \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$ is used to represent the candidate mutual exclusion subgraph, where edge weight is defined by:

$$W^{(\mathcal{C})}_{ij} = -\Big|N_{\mathcal{Q}}(c_i) \cap N_{\mathcal{Q}}(c_j)\Big|.$$

To avoid bias to popular candidates, we further normalize $\mathbf{W}^{(\mathcal{C})}$ as $\mathbf{S}^{(\mathcal{C})} = \mathbf{D}^{(\mathcal{C})-1/2}\mathbf{W}^{(\mathcal{C})}\mathbf{D}^{(\mathcal{C})-1/2}$, which discounts the co-occurrence frequency with popularity (frequency) of each candidate. Here, $\mathbf{D}^{(\mathcal{C})} \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$ is the degree matrix with $D^{(\mathcal{C})}_{ii} = \sum_{j=1}^{|\mathcal{C}|} |W^{(\mathcal{C})}_{ij}|$. One can see that the normalized edge weight $S^{(\mathcal{C})}_{ij}$ has similar statistical property as the PMI score, in terms of measuring association between c_i and c_j.

3.5 Graph Extension for Tail Web Pages

When collecting candidate queries, existing works only consider web pages that are clicked by entity name. Tailed web pages will be ignored in such case, and thus we will miss many true synonyms (*e.g.*, "*blue hen state*" in Fig. 1). To encounter this issue, one needs to collect appropriate seed queries which help reach tailed web pages in query log, and construct a more comprehensive data model based on a richer set of web pages and queries.

In practice, we observed that users tend to reformulate queries to reach these tailed web pages, by *augmenting entity name with entity contexts* (see "*Delaware chicken*" in Fig. 1). It is thus natural to extend our seed query, *i.e.*, entity name, into a set of seed queries for candidate generation, so that more tailed web pages can be included. Specifically, given a structured entity $e = (r_e, \mathcal{U}_e, C_e)$, we first extract entity contexts for e, by exploiting queries which click on source

web pages $\mathcal{U}_e$, *i.e.*, $\mathcal{Q}' = \bigcup_{u \in \mathcal{U}_e} N_{\mathcal{Q}}(u)$. For queries $q \in \mathcal{Q}'$ containing entity name r_e as substring, we extract unigrams from q, excluding r_e, stopwords, and numbers. By doing so, a set of high quality entity contexts, $\mathcal{W}_e$, can be collected, and will be further used to search seed queries in query log $\mathcal{L}$. In our implementation, we search over $\mathcal{L}$ to find queries which contain r_e, and at least one entity context from $\mathcal{W}_e$, leading to a set of seed queries, $\mathcal{Q}_e$. By replacing r_e with $\mathcal{Q}_e$ in candidate generation step (Sec. 2.2), we can obtain a more comprehensive set of web pages, *i.e.*, $\bigcup_{q \in \mathcal{Q}_e} N_{\mathcal{U}}(q)$, compared to $N_{\mathcal{U}}(e)$. Following the same steps in the rest of candidate generation, an extended graph can be constructed to include more tailed web pages.

4. SYNONYM DISCOVERY ON GRAPHS

4.1 Data Model Instantiation

We start by introducing how to instantiate the heterogeneous graph G for a structured entity by using its structured attributes. Entity name r_e and existing synonyms C_e satisfy the proposed properties for entity synonym, and source web pages $\mathcal{U}_e$ also are high quality instances for entity page. Therefore, we use these objects as positive labels to perform the heterogeneous graph-based ranking, by initializing them with the highest entity synonym scores and entity page scores, respectively.

Specifically, a label vector is defined for each type of object, where the element values represent our prior knowledge on the objects. Specifically, we define label vectors $\mathbf{y}_{\mathcal{C}} \in \mathbb{R}^{|\mathcal{C}|}$ for candidates $\mathcal{C}$ and $\mathbf{y}_{\mathcal{U}} \in \mathbb{R}^{|\mathcal{U}|}$ for web pages $\mathcal{U}$, accordingly. If a candidate $c_i \in \mathcal{C}_e$, then $y_{\mathcal{C}i} = 1$. Otherwise, $y_{\mathcal{C}i} = 0$. Similarly, for web pages $\mathcal{U}$, we set $y_{\mathcal{U}i} = 1$ if $u_i \in \mathcal{U}_e$, and 0 otherwise. As for keywords $\mathcal{W}$, its label vector $\mathbf{y}_{\mathcal{W}} \in \mathbb{R}^{|\mathcal{W}|}$ is set as $y_{\mathcal{W}i} = 0$ in this study, but one can collect entity contexts as positive labels based on the procedure in Sec. 3.5.

4.2 Problem Formulation

Mathematically, we formulate a joint optimization problem to unify different types of relations in the constructed heterogeneous graph G. Each type of relation is encoded into one graph regularization term to enforce the local and global consistency principle [34], *i.e.*, two linked objects tend to have similar or different corresponding scores according to the sign and strength of relationship between them. Different terms are then combined with corresponding weights and further unified with label supervision terms to form the objective function.

More precisely, let vector $\mathbf{f}_{\mathcal{C}} \in \mathbb{R}^{|\mathcal{C}|}$ denote the entity candidate scores for $\mathcal{C}$ given target entity e, *i.e.*, $f_{\mathcal{C}i} = f_{\mathcal{C}}(e, c_i)$. Similarly, we can define the entity context score vector $\mathbf{f}_{\mathcal{W}} \in \mathbb{R}^{|\mathcal{W}|}$ for $\mathcal{W}$ and entity page score vector $\mathbf{f}_{\mathcal{U}} \in \mathbb{R}^{|\mathcal{U}|}$ for $\mathcal{U}$. We thus have the objective function $\mathcal{O}$ as follows:

$$
\begin{aligned}
&\min_{\mathbf{f}_{\mathcal{C}}, \mathbf{f}_{\mathcal{W}}, \mathbf{f}_{\mathcal{U}}} \mathcal{O}(\mathbf{f}_{\mathcal{C}}, \mathbf{f}_{\mathcal{W}}, \mathbf{f}_{\mathcal{U}}) \qquad (1)\\
&= \lambda_{\mathcal{CU}} \sum_{i=1}^{|\mathcal{C}|} \sum_{j=1}^{|\mathcal{U}|} W_{ij}^{(\mathcal{CU})} \Big(\frac{f_{\mathcal{C}i}}{\sqrt{D_{ii}^{(\mathcal{CU})}}} - \frac{f_{\mathcal{U}j}}{\sqrt{D_{jj}^{(\mathcal{UC})}}} \Big)^2 \\
&+ \lambda_{\mathcal{WU}} \sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{|\mathcal{U}|} W_{ij}^{(\mathcal{WU})} \Big(\frac{f_{\mathcal{W}i}}{\sqrt{D_{ii}^{(\mathcal{WU})}}} - \frac{f_{\mathcal{U}j}}{\sqrt{D_{jj}^{(\mathcal{UW})}}} \Big)^2 \\
&+ \lambda_{\mathcal{CW}} \sum_{i=1}^{|\mathcal{C}|} \sum_{j=1}^{|\mathcal{U}|} |W_{ij}^{(\mathcal{CW})}| \Big(\frac{f_{\mathcal{C}i}}{\sqrt{D_{ii}^{(\mathcal{CW})}}} - \frac{\mathrm{sgn}(W_{ij}^{(\mathcal{CW})}) f_{\mathcal{W}j}}{\sqrt{D_{jj}^{(\mathcal{WC})}}} \Big)^2 \\
&+ \frac{\lambda_{ME}}{2} \sum_{i,j=1}^{|\mathcal{C}|} |W_{ij}^{(\mathcal{C})}| \Big(\frac{f_{\mathcal{C}i}}{\sqrt{D_{ii}^{(\mathcal{C})}}} + \frac{f_{\mathcal{C}j}}{\sqrt{D_{jj}^{(\mathcal{C})}}} \Big)^2 + \lambda_{\mathcal{C}} \sum_{i=1}^{|\mathcal{C}|} (f_{\mathcal{C}i} - y_{\mathcal{C}i})^2 \\
&+ \lambda_{\mathcal{W}} \sum_{i=1}^{|\mathcal{W}|} (f_{\mathcal{W}i} - y_{\mathcal{W}i})^2 + \lambda_{\mathcal{U}} \sum_{i=1}^{|\mathcal{U}|} (f_{\mathcal{U}i} - y_{\mathcal{U}i})^2,
\end{aligned}
$$

where $0 < \{\lambda_{\mathcal{WU}}, \lambda_{\mathcal{CU}}, \lambda_{\mathcal{CW}}, \lambda_{ME}, \lambda_{\mathcal{C}}, \lambda_{\mathcal{W}}, \lambda_{\mathcal{U}}\} < 1$ are tuning parameters which control the trade-off between different types of relationships and prior knowledge.

The candidate-page relation is modeled by the first term in Eq. (1). Minimizing the term forces the linked candidate and web page in $G^{(\mathcal{CU})}$ to have similar entity synonym score and entity page score, which follows Hypothesis 1. We can rewrite the term into $(\mathbf{f}_{\mathcal{C}}^T \mathbf{f}_{\mathcal{C}} - 2 \cdot \mathbf{f}_{\mathcal{C}}^T \mathbf{S}^{(\mathcal{CU})} \mathbf{f}_{\mathcal{U}} + \mathbf{f}_{\mathcal{U}}^T \mathbf{f}_{\mathcal{U}})$. The second term of Eq. (1) can be rewritten as $(\mathbf{f}_{\mathcal{W}}^T \mathbf{f}_{\mathcal{W}} - 2 \cdot \mathbf{f}_{\mathcal{W}}^T \mathbf{S}^{(\mathcal{WU})} \mathbf{f}_{\mathcal{U}} + \mathbf{f}_{\mathcal{U}}^T \mathbf{f}_{\mathcal{U}})$ in a similar way, which models the keyword-page relation based on Hypothesis 1. We omit the details here.

The third term in Eq. (1) attempts to model the signed relation between candidates and keywords in $G^{(\mathcal{CW})}$, following Hypothesis 2. We define the sign function $\mathrm{sgn}(a) = a$ if $a \geq 0$ and $\mathrm{sgn}(a) = -a$ if $a < 0$. When there is positive relation between a candidate and a keyword, the term forces them to have similar entity synonym score and context score; otherwise, the term forces the scores of two linked objects to be different. This term can be further rewritten as $\mathbf{f}_{\mathcal{C}}^T \mathbf{f}_{\mathcal{C}} - 2 \cdot \mathbf{f}_{\mathcal{C}}^T \mathbf{S}^{(\mathcal{CW})} \mathbf{f}_{\mathcal{W}} + \mathbf{f}_{\mathcal{W}}^T \mathbf{f}_{\mathcal{W}}$.

The mutual exclusion relation between candidates is modeled by the fourth term in Eq. (1), which follows the Hypothesis 3. Ideally, if two candidates have strong mutual exclusion relationship and one of them have high entity synonym score, then the other one is forced to have low entity synonym score. We can rewrite the term into $\mathbf{f}_{\mathcal{C}}^T \mathbf{f}_{\mathcal{C}} - \mathbf{f}_{\mathcal{C}}^T \mathbf{S}^{(\mathcal{C})} \mathbf{f}_{\mathcal{C}}$.

The last three terms in Eq. (1) encodes our prior knowledge on entity synonyms and entity pages, by enforcing the consistency between the estimated scores and initial labels.

4.3 The Closed Form Solution

With above equations, the definitions of augmented score vector $\mathbf{f} = [\mathbf{f}_{\mathcal{C}}^T, \mathbf{f}_{\mathcal{W}}^T, \mathbf{f}_{\mathcal{U}}^T]^T$ and label vector $\mathbf{y} = [\mathbf{y}_{\mathcal{C}}^T, \mathbf{y}_{\mathcal{W}}^T, \mathbf{y}_{\mathcal{U}}^T]^T$, Eq. (1) can be further rewritten into a more concise form:

$$
\min_{\mathbf{f}} \mathcal{F}(\mathbf{f}) = \mathbf{f}^T \mathbf{M} \mathbf{f} + \mathbf{f}^T \mathbf{\Omega} \mathbf{f} - 2 \cdot \mathbf{f}^T \mathbf{\Omega} \mathbf{y}. \qquad (2)
$$

Here, we define graph Laplacian matrix $\mathbf{L}^{(\mathcal{C})} = \mathbf{I}_{\mathcal{C}} - \mathbf{f}_{\mathcal{C}}^T \mathbf{S}^{(\mathcal{C})} \mathbf{f}_{\mathcal{C}}$, a symmetric matrix $\mathbf{M}$ as follows:

$$
\begin{bmatrix}
(\lambda_{\mathcal{CU}} + \lambda_{\mathcal{CW}}) \mathbf{I}_{\mathcal{C}} + \lambda_{ME} \mathbf{L}^{(\mathcal{C})} & -\lambda_{\mathcal{CW}} \mathbf{S}^{(\mathcal{CW})} & -\lambda_{\mathcal{CU}} \mathbf{S}^{(\mathcal{CU})} \\
-\lambda_{\mathcal{CW}} \mathbf{S}^{(\mathcal{CW})T} & (\lambda_{\mathcal{CW}} + \lambda_{\mathcal{WU}}) \mathbf{I}_{\mathcal{W}} & -\lambda_{\mathcal{WU}} \mathbf{S}^{(\mathcal{WU})} \\
-\lambda_{\mathcal{CU}} \mathbf{S}^{(\mathcal{CU})T} & -\lambda_{\mathcal{WU}} \mathbf{S}^{(\mathcal{WU})T} & (\lambda_{\mathcal{CU}} + \lambda_{\mathcal{WU}}) \mathbf{I}_{\mathcal{U}}
\end{bmatrix}.
$$

We can prove that $\mathbf{M}$ is positive semi-definite by referring to Eq. (1). We also define a diagonal matrix $\mathbf{\Omega} = \mathrm{diag}(\lambda_{\mathcal{C}} \mathbf{I}_{\mathcal{C}}, \lambda_{\mathcal{W}} \mathbf{I}_{\mathcal{W}}, \lambda_{\mathcal{U}} \mathbf{I}_{\mathcal{U}})$ which can be proved to be positive definite since $\{\lambda_{\mathcal{C}}, \lambda_{\mathcal{W}}, \lambda_{\mathcal{U}}\} > 0$.

By taking the derivative with respective to $\mathbf{f}$ and setting it to zero, we have $\partial \mathcal{F} / \partial \mathbf{f} = 2(\mathbf{M} + \mathbf{\Omega})\mathbf{f} - 2\mathbf{\Omega}\mathbf{y} = 0$. Since $(\mathbf{M} + \mathbf{\Omega})$ is positive definite and invertible, the closed form solution for Eq. (2) thus can be derived as $\mathbf{f}^* = (\mathbf{M} + \mathbf{\Omega})^{-1} \mathbf{\Omega} \cdot \mathbf{y}$. The objective function $\mathcal{F}$ is strictly convex since the second-order derivative of $\mathcal{F}$ is $(\mathbf{M} + \mathbf{\Omega})$, which is positive definite. Therefore, the convex optimization problem in Eq. (1) has the closed form solution as its *global minimum*.

4.4 An Efficient Iterative Algorithm

Directly using the closed-form solution to compute the scores may be very expensive, especially the matrix inverse operations on large matrix. Instead, people prefer iterative solution in such case, where only multiplication operations between sparse matrices are required.

With scores for all variables initialized by their labels, *i.e.*, $\mathbf{f}^{(0)} = \mathbf{y}$, for each variable in $\{\mathbf{f}_{\mathcal{C}}, \mathbf{f}_{\mathcal{W}}, \mathbf{f}_{\mathcal{U}}\}$, we iteratively update its score by using the current scores of all three variables, similar to [18]. The update formula can be derived by taking derivative of $\mathcal{F}$ with respect to each variable.

Table 3: Example entities used in evaluations.

Domain	Named Entities
Car	bmw z3, ford mustang, nissan gt r
Organization	special forces, nbc, 3m
Product	argus c3, adobe flash, canon 50mm lens
Sports	kobe bryant, john lackey, shane victorino
Location	atlanta, delaware, las vegas

For synonym candidates, we set $\partial\mathcal{F}/\partial\mathbf{f}_{\mathcal{C}} = 0$ and obtain the iterative update formula in the $(t+1)$-th step as follows:

$$\mathbf{f}_{\mathcal{C}}^{(t+1)} = \frac{\lambda_{\mathcal{CU}}}{\alpha}\mathbf{S}^{(\mathcal{CU})}\mathbf{f}_{\mathcal{U}}^{(t)} + \frac{\lambda_{\mathcal{CW}}}{\alpha}\mathbf{S}^{(\mathcal{CW})}\mathbf{f}_{\mathcal{W}}^{(t)} + \frac{\lambda_{ME}}{\alpha}\mathbf{S}^{(\mathcal{C})}\mathbf{f}_{\mathcal{C}}^{(t)} + \frac{\lambda_{\mathcal{C}}}{\alpha}\mathbf{y}_{\mathcal{C}}, \tag{3}$$

where we define $\alpha = (\lambda_{\mathcal{CU}} + \lambda_{\mathcal{CW}} + \lambda_{ME} + \lambda_{\mathcal{C}}) > 0$.

For keywords, we set $\partial\mathcal{F}/\partial\mathbf{f}_{\mathcal{W}} = 0$ and obtain the iterative update formula as follows:

$$\mathbf{f}_{\mathcal{W}}^{(t+1)} = \frac{\lambda_{\mathcal{WU}}}{\beta}\mathbf{S}^{(\mathcal{WU})}\mathbf{f}_{\mathcal{U}}^{(t)} + \frac{\lambda_{\mathcal{CW}}}{\beta}\mathbf{S}^{(\mathcal{CW})^T}\mathbf{f}_{\mathcal{C}}^{(t)} + \frac{\lambda_{\mathcal{W}}}{\beta}\mathbf{y}_{\mathcal{W}}, \tag{4}$$

where we define $\beta = (\lambda_{\mathcal{WU}} + \lambda_{\mathcal{CW}} + \lambda_{\mathcal{W}}) > 0$.

For web pages, we set $\partial\mathcal{F}/\partial\mathbf{f}_{\mathcal{U}} = 0$ and obtain the iterative update formula as follows:

$$\mathbf{f}_{\mathcal{U}}^{(t+1)} = \frac{\lambda_{\mathcal{WU}}}{\gamma}\mathbf{S}^{(\mathcal{WU})^T}\mathbf{f}_{\mathcal{W}}^{(t)} + \frac{\lambda_{\mathcal{CU}}}{\gamma}\mathbf{S}^{(\mathcal{CU})^T}\mathbf{f}_{\mathcal{C}}^{(t)} + \frac{\lambda_{\mathcal{U}}}{\gamma}\mathbf{y}_{\mathcal{U}}, \tag{5}$$

where we define $\gamma = (\lambda_{\mathcal{WU}} + \lambda_{\mathcal{CU}} + \lambda_{\mathcal{U}}) > 0$.

In the iterative solution, each object iteratively spreads its score to its neighbors in the constructed heterogeneous graph G following the proposed hypotheses in Sec. 3, until a global stable state is achieved. To our knowledge, this is the first work to perform graph-based semi-supervised learning on a *signed heterogeneous graph*. The proof procedure in [34] can be adopted to prove convergence for the proposed algorithm (to the closed form solution in Sec. 4.3). For lack of space, we do not include it here.

4.5 Post-Processing for Subset Output

In many applications, such as knowledge base enrichment, it is desirable to output a subset of high quality synonyms, instead of a ranked list of synonyms. Typical ways for creating subset output, such as cut-off by threshold and selecting top-K results, may generate low quality results because (1) different entities may have different scale of ranking scores due to variations of their input (*e.g.*, different number of existing synonyms); and (2) number of existing true synonyms varies a lot across different domains (see Table 5).

In our implementation, the cut-off is based on relative score changes in derived ranking list, since significant score gap is observed between true and false synonyms in the ranking list. More precisely, suppose $c^{(i)}$ is the i-th candidate in current ranking list, the list will be cut after candidate $c^{(i)}$ if $\frac{f_{\mathcal{C}}(c^{(i-1)}) - f_{\mathcal{C}}(c^{(i)})}{f_{\mathcal{C}}(c^{(i)})} > \delta$. This technique is applied in evaluations in Sec. 6.3 and in generating system output (Table 6).

4.6 Computational Complexity Analysis

In this section, we analyze the computational complexity of proposed StrucSyn framework using the term *flame* [14].

For construction of heterogeneous graph, the cost for computing $G^{(\mathcal{CU})}$ and $G^{(\mathcal{WU})}$ are around $|\mathcal{C}|d_{\mathcal{C}}$ flames and $|\mathcal{W}|d_{\mathcal{W}}$ flames, respectively. $d_{\mathcal{Q}}$ and $d_{\mathcal{W}}$ are the average numbers of web pages related to a candidate and a keyword, respectively $(d_{\mathcal{Q}}, d_{\mathcal{W}} \ll |\mathcal{U}|)$. Constructing $G^{(\mathcal{CW})}$ costs around $|\mathcal{C}|l_{\mathcal{Q}}l_q$ flames. $l_{\mathcal{Q}}$ is average number of support queries for a candidate, and l_q is average query length in $\mathcal{L}$ ($l_{\mathcal{Q}}l_q \ll |\mathcal{W}|$ and l_q is constant). Constructing $G^{(\mathcal{C})}$ costs around $3|\mathcal{C}|l_{\mathcal{Q}}$ flames, and instantiating the graph (*i.e.*, collect query evidence for keywords) costs around $|\mathcal{Q}|$ flames.

Table 4: Statistics of heterogeneous graphs.

Domain	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{W}\rvert$	#Relationship
Car	2,026	∼600k	1,280	∼1.8M
Organization	6,403	∼1.4M	3,676	∼5.1M
Product	669	∼130k	299	∼383k
Sports	1,411	∼675k	916	∼2.1M
Location	4,081	∼1.7M	3,713	∼5.0M

For graph-based ranking, in each iteration, update calculation for $\mathbf{f}_{\mathcal{C}}$ costs around $|\mathcal{C}|(d_{\mathcal{C}} + l_{\mathcal{C}}l_q + 3l_q + 1)$ flames. Similarly, updating $\mathbf{f}_{\mathcal{W}}$ costs around $|\mathcal{W}|d_{\mathcal{W}} + |\mathcal{C}|l_{\mathcal{C}}l_q + |\mathcal{W}|$ flames, and updating $\mathbf{f}_{\mathcal{U}}$ costs around $|\mathcal{W}|d_{\mathcal{W}} + |\mathcal{C}|d_{\mathcal{C}} + |\mathcal{U}_e|$ flames. Suppose the algorithm can converge in t iterations ($t \approx 10$ based on our empirical studies), the total time complexity for the proposed framework is $\mathcal{O}\Big(t \cdot \big(|\mathcal{C}|d_{\mathcal{C}} + |\mathcal{W}|d_{\mathcal{W}} + |\mathcal{C}|l_{\mathcal{Q}}\big)\Big)$.

5. MODEL GENERALIZATION

In practice, many other structured attributes, besides source web pages and existing synonyms, may be available for entities. For example, Freebase stores for entity Kobe Bryant: place of birth, profession, sports teams, etc. Intuitively, a web page is more likely an entity page if it has strong association with a rich set of structured attributes of the entity. Compared to a news article on Kobe Bryant, a bibliography page is a better entity page for him since it contains more informative attributes such as his date of birth, place of birth and education. Therefore, structured attributes can help infer entity pages and assist the discovery of entity synonyms.

A straightforward way is to identify structured attributes in web document and use them to measure how likely the web page is an entity page. More precisely, in data model instantiation, entity page score of a web page can be initialized by relevance between entity and the web page, computed based on the occurrences of structured attributes within the web document. This simple method, however, assumes all attributes are equally informative for inferring entity pages, and ignores occurrences of attribute instances that are closely related to the entity.

More generally, we propose to map structured attribute into object type in our data model. Suppose there are d extra types of structured attributes, $\mathcal{A} = \{A_1, ..., A_d\}$. We denote by $A_e = \{a_e^{(1)}, ..., a_e^{(d)}\}$ the attribute instances (values) for entity e. In our generalized data model, each structured attribute $A_i \in \mathcal{A}$ corresponds to a new object type, and its instances correspond to objects of type-A_i, $a_j^{(i)}$ where $j = 1, ..., |A_i|$. Relations can be constructed between attribute objects and web pages to represent their association strength. Specifically, for each attribute type A_i, we form links between its attribute object $a_j^{(i)}$ and a web page u, if $a_j^{(i)}$ appears in u. The links can be weighted by frequency of $a_j^{(i)}$'s occurrences in u. By doing so, a subgraph $G^{(A_i\mathcal{U})}$, denoted by a bi-adjacency matrix $\mathbf{W}^{(A_i\mathcal{U})}$, can be constructed for each attribute A_i. With the generalized data model, by setting $a \in A_e$ as positive examples, we can derive importance scores for attribute instances of different attributes with respect to e, and propagate entity page scores $f_{\mathcal{U}}$.

6. EXPERIMENTS

6.1 Experimental Setup

1. Data Preparation. We use a sampled query log of a commercial search engine, which contains ∼136 million unique queries and ∼156 million unique URL. We consider clickthrough relationships if and only if the click frequency is larger than 4. Queries containing frequent spelling errors are further filtered out.

We collect named entities from five different domains (20 from each), *i.e.*, car, organization, product, sports and location (see Table 3 for examples). These domains are chosen because (1) many applications require entity extractions on them; and (2) entities in these domain may be ambiguous. Table 4 shows statistics of the constructed graphs for the entities. For structure attributes, we extract from Freebase "*common.topic.topic_equivalent_webpage*" as source web pages $\mathcal{U}_e$, and "*common.topic.alias*" as existing synonyms $\mathcal{C}_e$.

2. Gold Standard Synonyms. We collect a set of gold standard synonyms for entities in our evaluation set, by combining two sets of ground truth synonyms: existing entity synonyms in Freebase and manually labeled synonyms.

Over 88% of the test entities have existing alias in Freebase. However, the number of existing alias is limited, *e.g.*, ∼3.7 synonyms for the 100 entities on average, and it varies a lot across different domains. Also, most existing aliases in Freebase are widely known acronyms and nicknames, which have limited diversity. This motivates us to conduct manual labeling for further enriching the gold standard set.

In our labeling process, for each candidate n-grams in $\mathcal{N}$ (see Sec 2.2), we manually judge whether it is true synonym for e, and collect all positive ones to form the second part of the gold standard set. In the gold standard set, there may be synonyms of different categories, *e.g.*, "*terminus ga*" and "*atl*" for entity *Atlanta*, as well as spelling variants within one category, *e.g.*, "*terminusga*" and "*terminus*".

We summarize statistics of the five evaluation sets in Table 5. Naturally, the difficulty of discovering synonyms is different due to variance in domain specialty and extent of entity ambiguity. For example, product domain has more true synonyms but less candidates while car domain has more candidates but less true synonyms.

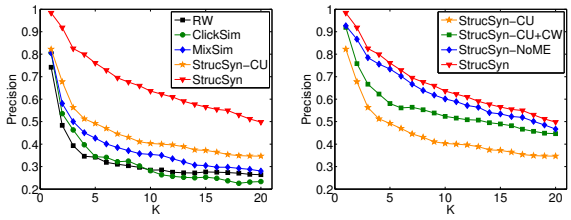
3. Evaluation Metrics. There are mainly two ways for effectiveness evaluation: (1) output as a ranked list of synonyms; and (2) output as a subset of synonyms with automatic cutoff technique. By ranking synonyms in terms of entity synonym score, we can employ evaluation metrics for ranking problem, such as Precision at position K (P@K), to evaluate the method performance. P@K is defined as the percentage of true synonyms which appear in the top-K positions.

In many applications, it is more desirable to have a subset of high quality synonyms as output. The quality of output can be evaluated by Precision and Recall. However, similar to the case in [5], it is very difficult to know the universal set of synonyms for each entity, and therefore hard to report the traditional recall number. We thus use average size of output subset per entity as an alternative for Recall.

Table 5: Statistics of the evaluation sets.

Domain	Car	Org	Product	Sports	Loc
Avg #Candidates	2,579	8,711	797	1,914	6,534
%True synonyms	1.78	0.87	8.27	1.98	1.32

4. Compared Methods. We compare the proposed method (StrucSyn) with its variants in order to validate different hypotheses in our data model—**StrucSyn** is the proposed full fledged model with all relations ($\mathcal{C}_e$ are excluded from method input); **StrucSyn**$_{CU}$ considers only candidate-page relation; **StrucSyn**$_{CU+CW}$ considers only candidate-page and candidate-keyword relations; and **StrucSyn**$_{NoME}$ excludes mutual exclusion relation. Based on the parameter study in Sec. 6.5, we set $\{\lambda_{\mathcal{C}}, \lambda_{\mathcal{W}}, \lambda_{\mathcal{U}}\} = \{0.1, 0.01, 0.008\}$ and $\{\lambda_{\mathcal{CU}}, \lambda_{\mathcal{CW}}, \lambda_{\mathcal{WU}}, \lambda_{ME}\} = \{0.33, 0.4, 0.08, 0.05\}$ for StrucSyn and its variants throughout the experiments. Several state-of-the-art methods are also implemented for comparison—**RW** [9] performs backward random walk on click graph;



(a) Compared w/ baselines (b) Compared w/ variants

Figure 7: Performance comparison in terms of Precision at different positions (K).

ClickSim [7] considers intersecting page count and click ratio between entity name and candidate queries; **MixSim** [5] is a hybrid method that leverages both pseudo documents and query contexts.

6.2 Performance Comparison

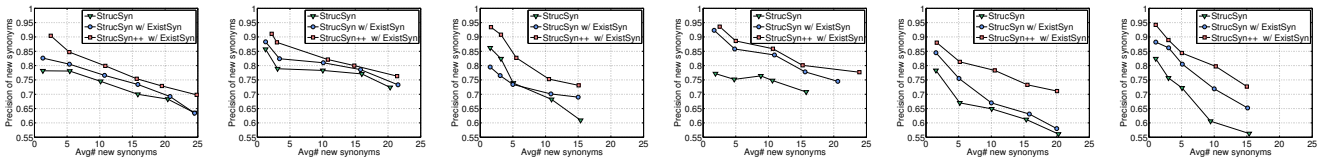
First, the proposed method is compared to existing methods using Precision at different positions (1 to 20). Fig. 7(a) summarizes the aggregated comparison results over all five domains. Overall, StrucSyn significantly outperforms other methods on different Ks, *e.g.*, 53% improvement over best performing existing method MixSim on P@10, which shows the performance gains from leveraging source web pages, exploring sub-queries and studying heterogeneous relationships between the objects. In terms of leveraging query context, StrucSyn explores entity-specific contexts while MixSim simply considers all contexts equally in the entire query log. Thus, it will not suffer from noisy contexts or ambiguous contexts of different entities. With entity pages and entity contexts, StrucSyn can uncover high quality synonyms without strict assumption like two-way checking in MixSim.

In particular, we notice that click-based variant StrucSyn$_{CU}$, which leverages only candidate-page relation, achieves superior performance over both click-based methods (RW and ClickSim) and hybrid method (MixSim). In fact, StrucSyn$_{CU}$ and RW are similar in weighting query-page relationship and method input, and are only different in that StrucSyn$_{CU}$ aggregates statistics of sub-query candidates across multiple support queries. This validates our claim that exploring sub-queries can help uncover more synonyms.

In Fig. 7(b), we further compare StrucSyn with its variants to validate our hypotheses in the data model. By comparing StrucSyn$_{CU}$ and StrucSyn$_{CU+CW}$, one can clearly observe the performance gain (*e.g.*, over 36% enhancement on P@10) by exploring co-occurrence between candidates and keywords. This shows evidence for Hypothesis 2 and demonstrates the effectiveness of entity context for deriving synonyms. Compared to StrucSyn$_{CU+CW}$, StrucSyn$_{NoME}$ further incorporates keyword-page relation (see Hypothesis 1) when deriving entity context scores and entity page scores, and obtains performance enhancement over StrucSyn$_{CU+CW}$, *e.g.*, around 27% improvement on P@5. Finally, performance gain by leveraging candidate mutual exclusion relation can be seen from the gap between StrucSyn and StrucSyn$_{NoMe}$, demonstrating the effectiveness of Hypothesis 3 in detecting entity mention boundary.

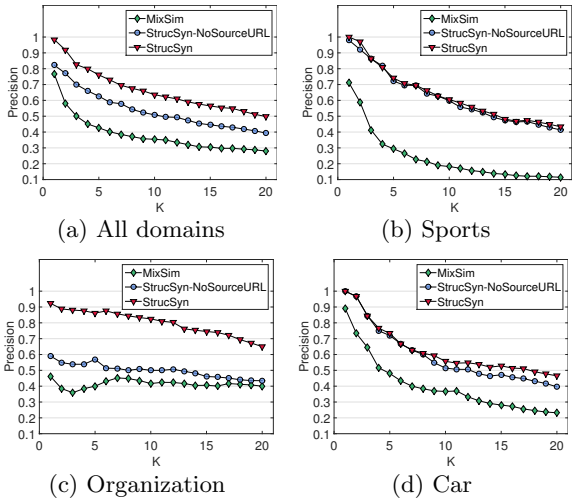
In particular, StrucSyn$_{CU+CW}$ can be seen as an advanced version of MixSim. It not only explores sub-queries, but also leverages source web pages and local contexts using a label propagation framework. Gap between StrucSyn$_{CU+CW}$ and StrucSyn demonstrates that the proposed method still outperforms the state-of-the-art method MixSim though it is augmented to an advanced version, *i.e.*, StrucSyn$_{CU+CW}$.

In terms of method input, StrucSyn takes source web pages as extra information compared to state-of-the-art meth-



(a) All domains (b) Car (c) Sports (d) Organization (e) Location (f) Product

Figure 9: Enriching entity knowledge base.



(a) All domains (b) Sports (c) Organization (d) Car

Figure 8: Influence of entity source web pages.

ods. To conduct more comprehensive comparison, we exclude source web pages from StrucSyn to form a new variant, StrucSyn$_{NoSourceURL}$. Fig. 8 summarizes the study results across different domains. In Fig. 8(a), StrucSyn$_{NoSourceURL}$ outperforms the best performing existing method MixSim in terms of aggregated results over all domains. This shows that our proposed method, even without source web pages as input, can still obtain superior performance since it models relations between candidates, keywords and web pages in a principled way and resolves limitations such as tailed synonyms and sub-query synonyms in previous methods.

We further look into three different domains (*i.e.*, sports, organization, car) to study influence of source web pages on the proposed method. As shown in Figs. 8(b) and 8(d), source web pages have little influence on entities of sports or car domain. This is because in these two domains popular web pages in query log usually are good entity pages. Thus, the performance gain is mainly produced by heterogeneous graph-based framework. On the other hand, in organization domain (see Fig. 8(c)), significant performance boost can be observed after including source web pages in the method. This is because popular web pages often focus on certain aspects of organization entities, *e.g.*, lottery and tourism of Delaware, and thus are less likely high quality entity pages.

6.3 Enriching Entity Knowledge Bases

Existing synonym in entity knowledge bases usually have limited coverage and diversity. It is desirable that our method can complement existing synonyms in knowledge bases, by outputting high quality subsets of synonyms.

We study the performance of StrucSyn and its two variants. Compared to StrucSyn, StrucSyn$_{w/ExistSyn}$ incorporates entity existing synonyms $\mathcal{C}_e$ in Freebase to instantiate the data model (Sec. 4.1), and StrucSyn++$_{w/ExistSyn}$ uses both extended graph (Sec. 3.5) and existing synonyms. Subsets of synonyms are output using automatic cut-off technique. We compute precision and average number of new synonyms after excluding existing synonyms from the output synonym subset. By varying the threshold δ of automatic cut-off technique, we can generate output subsets of different sizes and precisions. Fig. 9 summarizes the performances over different domains using different thresholds.

One can see from Fig. 9(a) that, in general, the proposed methods can discover subsets of new synonyms with high quality and reasonable size to complement existing ones in Freebase, *e.g.*, a subset of 10 new synonyms with Precision over 82% by StrucSyn++$_{w/ExistSyn}$. By using existing synonyms to instantiate the data model, StrucSyn$_{w/ExistSyn}$ obtains precision boost over StrucSyn. This is mainly because that existing synonyms provide positive labels on candidates and thus help derive entity contexts based on candidate-keyword relation, as well as entity pages based on candidate-page relation. With graph extension, StrucSyn++$_{w/ExistSyn}$ obtains further improvement over StrucSyn$_{w/ExistSyn}$ since extended graph covers more true synonyms and contains richer information between objects.

When looking at each domain specifically, one can clearly see the performance differences across different domains. For example, location entities reach relatively low precision when output size is larger than 10 (see Fig. 9(e)), while car entities still preserve good precision with output size of 20, as shown in Fig. 9(b). This demonstrates the varying domain specialty, *e.g.*, number of true synonyms (Table 5), across different domains. It can be further observed that performance gaps between three methods vary, showing that existing synonyms and extended graph bring different degrees of enhancement in different domains. For example, by comparing results in Fig. 9(e) with those in Fig. 9(b), one can see that existing synonyms significantly boost precision for location entities but have limited influence on car entities. One major reason is the limited number of existing synonyms in car domain. In particular, with extended graph, we can significantly enhance the precision while providing better coverage for location entities (see Fig. 9(e)).

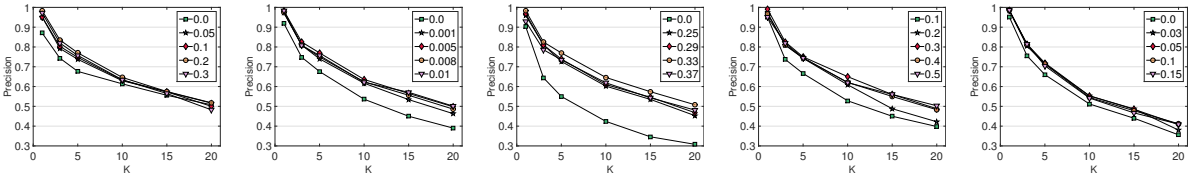
6.4 Case Studies

To provide a clearer look at the derived entity synonyms, we select six entity from five different domains, and use StrucSyn++$_{w/ExistSyn}$ to generate a subset of new synonyms for each entity, where we set the automatic cut-off threshold $\delta = 0.22$. Table 6 summarizes our results and provides existing synonyms in Freebase for comparison. First, we can see that the proposed method is able to generate reasonable size of new synonym subset with high quality, *i.e.*, only two synonyms (in grey color) are incorrect. Also, we observe that different kinds of entity synonyms are covered in the output. For entity *Kobe Bryant*, its returned new synonyms include atypical synonym "*kobe the mamba*", superstring "*kobe 24 bryant*", acronym "*kb*" and spelling variant "*coby bryant*". We find that most of the returned new synonyms are atypical—sharing little text similarity with the entity name (*e.g.*, see results for *Volkswagen Type 2*). This demonstrates the effectiveness of our method in discovering these interesting yet challenging atypical synonyms.

We observe that entities with ambiguous names benefit greatly from source web pages when comparing StrucSyn

Table 6: Example output of our method.

Entity	Freebase synonyms	New synonyms	Entity	Freebase synonyms	New synonyms
Volkswagen Type 2	transporter kombi camper bus	microbus combi thesamba type vw combi minibus	**BMW**	bayerische motoren werke	bavarian motor works beamer automobile bmwgroup bimmer beamer car
Atlanta	atlanta georgia atlanta ga city of atlanta atl a town	terminus ga ga capital city atl georgia antanta wiki capital of ga	**Kobe Bryant**	black mamba kobe bean bryant	kobe the mamba bean bryant coby bryant kobe 24 bryant kb
8th Armored Division	N/A	thundering herd 8th eighth armored div	**Skydrive**	onedrivecom windows live skydrive microsoft skydrive	windows live storage windows drive ms storage online



(a) $\lambda_{\mathcal{C}}$ (b) $\lambda_{\mathcal{U}}$ (c) $\lambda_{\mathcal{CU}}$ (d) $\lambda_{\mathcal{CW}}$ (e) λ_{ME}

Figure 10: Model parameter study.

with its variant StrucSyn$_{NoSourceURL}$. For example, while the returned synonyms by StrucSyn are all correct for NBA player *JR Smith*, those return by StrucSyn$_{NoSourceURL}$ (*e.g.*, "*smith nuggets*", "*smith plumbing*", "*smith drain*") are actually a mix of two different entities. For *CBS*, we find that source web pages also help focus on entire entity (*e.g.*, "*tiffany network*", "*columbia broadcasting system*" by StrucSyn), instead of aspects of it (*e.g.*, "*cbs games*","*cbstv*" by StrucSyn$_{NoSourceURL}$). These examples demonstrate the importance of source web pages for better entity modeling.

6.5 Parameter Study

We study the effect of parameters for StrucSyn on a validation set which consists of 100 entities from five domains. To study each parameter, we vary its value while fixing the values of other parameters as 0.1 and perform five-fold cross validation to obtain the Precision at different Ks. Due to space limit, only a part of results are presented in Fig. 10.

Overall, we find that all the relations are useful since setting any of them to zero will cause performance drop (*e.g.*, see Fig. 10(c) for $\lambda_{\mathcal{CU}} = 0$). Similarly, it can be seen that label information is also critical to the performance (*e.g.*, see Fig. 10(b) for $\lambda_{\mathcal{U}} = 0$). On the other hand, we find StrucSyn shows robust performance over a large range of values on each parameter. For example, in Fig. 10(c) when $\lambda_{\mathcal{CU}} \geq 0.25$, the performance changes slightly. Similar trend can be observed for $\lambda_{\mathcal{U}}$ in Fig. 10(b). Such robustness allows us to use a fix set of parameters while achieving high effectiveness across different domains.

In our experiments, StrucSyn usually converges (*i.e.*, relative change of objective value in Eq. (1) is smaller than 10^{-5}) within 10 iterations. For the test entities, StrucSyn finishes processing each entity in 2.51 seconds on average on a single server with 2.27Ghz Xeon CPU and 32GB memory.

7. RELATED WORK

Previous efforts on automatically discovering entity synonyms consider a variety of information. Given a corpus, one can leverage document-level co-occurrence [30, 2] and distributional similarity [22, 25] to extract synonyms. These corpus-based methods, however, may suffer from low accuracy due to noisy entity context, and sparse co-occurrences between entities. They also require sophisticated linguistic features and human labels for model learning. On the other hand, people start exploring query log, which provide more focused entity information compared to document, and propose a variety of features for synonym discovery, including query click similarity [33, 6, 7], query context similarity [5, 21], and pseudo-document similarity [5]. However, both lines of work take only entity name as input but ignore the inherent ambiguity nature of many entity names.

Recent studies [19, 21] use a set of homogeneous entities, *e.g.*, entities from the same domain, as input to help resolve name ambiguity and collect more accurate information on entity context. Such limited input schema, however, has difficulties in handling ad-hoc entities since identifying homogeneous entities for ad-hoc entity is already non-trivial.

To our knowledge, the proposed method is the first to study synonym discovery for *structured* entities. With entity-focus information provided by structured attributes, our method can conduct effective and efficient synonym discovery for single entity (see StrucSyn vs. MixSim [5], Figs. 8 and 9). Compared to many query-based methods [7, 5, 21], we further explore sub-queries as candidates and incorporate many types of information used by previous methods in a principled, heterogeneous graph-based framework.

In terms of identifying various ways to reference an entity, our work is related to entity resolution (coreference resolution), which aims to resolve various entity mentions, found in documents [26, 31] or structured data records [4, 13], into real-world entities. Entity linking [12, 29], on the other hand, focuses on matching entity mentions to entities in structured knowledge bases, which relies on surface forms (synonyms) of entities to generate candidates. Therefore, entity synonym discovery is orthogonal to entity linking and can complement it by providing a richer set of synonyms. Our work is also related to a rich body of works on identifying similar queries such as query clustering [28, 1, 17], intent mining [27, 32], query alternation [3, 8] and query suggestions [24, 23], in terms of measuring query similarity based on web data. There exists work on finding word (lexical) synonyms from document [20, 16], but we focus on discovering entity synonym from query log.

8. CONCLUSION

In this paper, we study the problem of synonym discovery for structured entities where structured attributes are leveraged to handle entities with ambiguous names and enrich the mining process. We propose to explore sub-queries as synonym candidates, and design a novel heterogeneous graph-based framework to model properties of entity synonym, by studying interplay between candidate, web page and keyword. Experimental results demonstrate the effectiveness of our method.

9. REFERENCES

[1] L. M. Aiello, D. Donato, U. Ozertem, and F. Menczer. Behavior-driven clustering of queries into topics. In *CIKM*, 2011.

[2] M. Baroni and S. Bisi. Using cooccurrence statistics and the web to discover synonyms in a technical language. In *LERC*, 2004.

[3] M. Bendersky, D. Metzler, and W. B. Croft. Effective query formulation with multiple information sources. In *WSDM*, pages 443–452, 2012.

[4] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *VLDB*, 18(1):255–276, 2009.

[5] K. Chakrabarti, S. Chaudhuri, T. Cheng, and D. Xin. A framework for robust discovery of entity synonyms. In *SIGKDD*, 2012.

[6] S. Chaudhuri, V. Ganti, and D. Xin. Exploiting web search to generate synonyms for entities. In *WWW*, 2009.

[7] T. Cheng, H. W. Lauw, and S. Paparizos. Entity synonyms for structured web search. *TKDE*, 24(10):1862–1875, 2011.

[8] N. Craswell, B. Billerbeck, D. Fetterly, and M. Najork. Robust query rewriting using anchor data. In *WSDM*, 2013.

[9] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR*, 2007.

[10] M. Danilevsky, C. Wang, N. Desai, X. Ren, J. Guo, and J. Han. Kert: Automatic extraction and ranking of topical keyphrases from content-representative document titles. *SDM*, 2014.

[11] A. El-Kishky, Y. Song, C. Wang, C. R. Voss, and J. Han. Scalable topical phrase mining from text corpora. *VLDB*, 2015.

[12] A. Gattani, D. S. Lamba, N. Garera, M. Tiwari, X. Chai, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. Entity extraction, linking, classification, and tagging for social media: a wikipedia-based approach. *VLDB*, 6(11):1126–1137, 2013.

[13] L. Getoor and A. Machanavajjhala. Entity resolution: theory, practice & open challenges. *VLDB*, 5(12):2018–2019, 2012.

[14] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[15] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *SIGIR*, 2009.

[16] G. Halawi, G. Dror, E. Gabrilovich, and Y. Koren. Large-scale learning of word relatedness with constraints. In *SIGKDD*, 2012.

[17] Y. Hu, Y. Qian, H. Li, D. Jiang, J. Pei, and Q. Zheng. Mining query subtopics from search log data. In *SIGIR*, 2012.

[18] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao. Graph regularized transductive classification on heterogeneous information networks. In *ECMLPKDD*, 2010.

[19] L. Jiang, P. Luo, J. Wang, Y. Xiong, B. Lin, M. Wang, and N. An. Grias: an entity-relation graph based framework for discovering entity aliases. In *ICDM*, 2013.

[20] H. Kim, X. Ren, Y. Sun, C. Wang, and J. Han. Semantic frame-based document representation for comparable corpora. In *ICDM*, pages 350–359, 2013.

[21] Y. Li, B.-J. P. Hsu, C. Zhai, and K. Wang. Mining entity attribute synonyms via compact clustering. In *CIKM*, 2013.

[22] D. Lin, S. Zhao, L. Qin, and M. Zhou. Identifying synonyms among distributionally similar words. In *IJCAI*, 2003.

[23] H. Ma, H. Yang, I. King, and M. R. Lyu. Learning latent semantic relations from clickthrough data for query suggestion. In *CIKM*, 2008.

[24] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM*, 2008.

[25] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, and V. Vyas. Web-scale distributional similarity and entity set expansion. In *EMNLP*, 2009.

[26] H. Poon and P. Domingos. Joint unsupervised coreference resolution with markov logic. In *ACL*, 2008.

[27] X. Ren, Y. Wang, X. Yu, J. Yan, Z. Chen, and J. Han. Heterogeneous graph-based intent learning with queries, web pages and wikipedia concepts. In *WSDM*, pages 23–32, 2014.

[28] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *WWW*, 2010.

[29] W. Shen, J. Wang, and J. Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *TKDE*, (99):1–20, 2014.

[30] P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *ECML*. 2001.

[31] C. Wang, K. Chakrabarti, T. Cheng, and S. Chaudhuri. Targeted disambiguation of ad-hoc, homogeneous sets of named entities. In *WWW*, 2012.

[32] X. Wang, D. Chakrabarti, and K. Punera. Mining broad latent query aspects from search sessions. In *SIGKDD*, 2009.

[33] X. Wei, F. Peng, H. Tseng, Y. Lu, and B. Dumoulin. Context sensitive synonym discovery for web search queries. In *CIKM*, 2009.

[34] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. In *NIPS*, 2004.