Learning Entity Types from Query Logs via Graph-Based Modeling

Jingyuan Zhang[†], Luo Jie[*], Altaf Rahman[*], Sihong Xie[†], Yi Chang[*] and Philip S. Yu[†,‡]
[†]Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA
[‡]Institute for Data Science, Tsinghua University, Beijing, China
[*]Yahoo! Labs, Sunnyvale, CA, USA
{jzhan8, sxie6, psyu}@uic.edu, luoj.roger@yahoo.com, {altafr, yichang}@yahoo-inc.com

ABSTRACT

Entities (e.g., person, movie or place) play an important role in real-world applications and learning entity types has attracted much attention in recent years. Most conventional automatic techniques use large corpora, such as news articles, to learn types of entities. However, such text corpora focus on general knowledge about entities in an objective way. Hence, it is difficult to satisfy those users with specific and personalized needs for an entity. Recent years have witnessed an explosive expansion in the mining of search query logs, which contain billions of entities. The word patterns and click-throughs in search logs are not found in text corpora, thus providing a complemental source for discovering entity types based on user behaviors. In this paper, we study the problem of learning entity types from search query logs and address the following challenges: (1) queries are short texts, and information related to entities is usually very sparse; (2) large amounts of irrelevant information exists in search logs, bringing noise in detecting entity types. In this paper, we first model query logs using a bipartite graph with entities and their auxiliary information, such as contextual words and clicked URLs. Then we propose a graph-based framework called ELP (**E**nsemble framework based on **L**able **P**ropagation) to simultaneously learn the types of both entities and auxiliary signals. In ELP, two separate strategies are designed to fix the problems of sparsity and noise in query logs. Extensive empirical studies are conducted on real search logs to evaluate the effectiveness of the proposed ELP framework.

Categories and Subject Descriptors

H.2.8 [**Database management**]: Database applications-Data mining

Keywords

Query; Entity; Graph

CIKM'15, October 19–23, 2015, Melbourne, Australia.

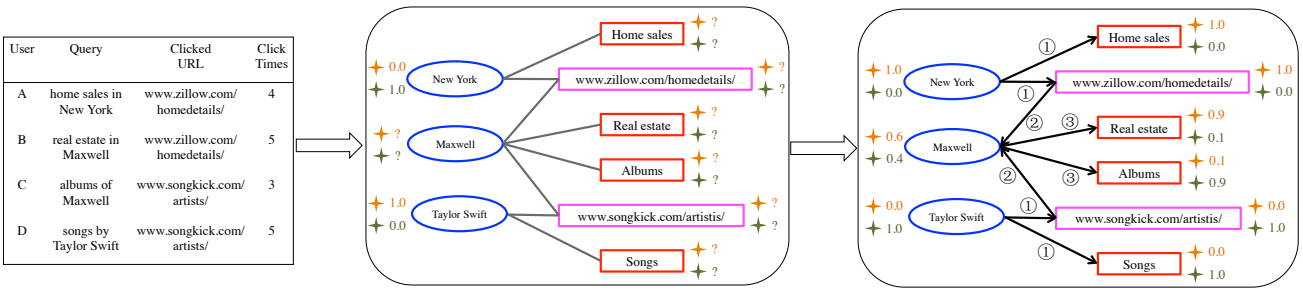
DOI: http://dx.doi.org/10.1145/2806416.2806498.

1. INTRODUCTION

An *entity* is something that exists in itself, actually or potentially, concretely or abstractly, physically or not [1]. Entities are forming the building block for various web applications. Yelp [2] is building on top of a corpus of *local* class entities (*e.g.*, the restaurant entity "The French Laundry", the Point of Interest entity "Golden Gate Bridge", etc.) associated with user reviews. IMDB [3] has a large corpus of *movie* and *actor* class entities. Modern search engines like Bing, Google and Yahoo! start building Knowledge Graph containing a large collection of diverse types of entities. When a user issues a question about an entity (*e.g.*, "net worth of Bill Gates" or "phone number of Gary Danko"), the search engine can retrieve results directly from the knowledge graph, satisfying the user's need and providing better user experience. Recent study shows that around 70% of the queries contain entity information [31, 22]. Hence, the coverage of entities is very important for these applications. Moreover, knowing the exact types of entities can help the application decide the best way in presenting results to users.

Various entity repositories, ranging from the more general collaborative knowledge bases such as Wikipedia and Freebase to the domain-specific corpora such as IMDB and Yelp, are widely used to extract entity information and aggregate the information into a comprehensive knowledge graph. However, there are several problems with this approach: (a) *coverage*: it is one of the key metric in measuring the quality of knowledge. Knowledge bases like Wikipedia and Freebase primary focus on popular entities from a few limited types, while other domain-specific corpora are more expensive to obtain. Plus, little information exists in knowledge bases for many less popular entities or newly generated entities, such as a new music title. It is difficult to identify and extract such entities in time; (b) *ambiguity*: multiple types of entity are often associated with the same string collected from the same or different sources. For example, the token "Chicago" is not only a city entity, but also a movie entity or a rock band entity. How to separate them apart in case little is known about the types of the entities, and how to rank these entities according to the *popularity* and/or user intent, are both quite challenging; (c) *discrepancy*: errors may exist due to user-generated contents via crowdsourcing, thus information extracted from these sources may be noisy and inconsistent.

[1] http://en.wikipedia.org/wiki/Entity
[2] http://www.yelp.com/
[3] http://www.imdb.com/



(a) Search query logs (b) Graph construction (c) Learning steps

Figure 1: An intuitive example of learning entity types from search query logs. In (b), the four-pointed orange and blue stars mean the person and place types, respectively. The number next to a star shows the probability of a node belonging to a type. "1.0" means the type is already known and "?" means that we need to learn the type from search query logs. In (c), each black arrowed line shows the propagation direction and the circled number on each line represents the order of the propagation process.

A lot of research work in the literature tries to overcome the above challenges from different perspectives. The existing knowledge bases could only cover a fraction of the whole entity space. In order to expand the size of knowledge bases, many automatic techniques have been proposed to discover entities and their types from different large corpora, such as news articles and web pages [3, 9, 30]. In addition, disambiguating entities from news reports is also studied in [14, 20]. Other sources such as search query logs can also be leveraged to extract and disambiguate entities. Since the search engine has become the main information source for most people to look for information, search query logs can be a nice complementary source for extracting new entity information as well as learning entity popularity and disambiguating entities. A few state-of-the-art approaches are proposed to classify and disambiguate entities in query logs [16, 26, 6]. For instance, intent-based Model (IM) [26] predicts entity type distributions by jointly modeling user intent and entity types via probabilistic inference in a graphical manner. Fast Entity Linker (FEL) is proposed in [6] to disambiguate entities by linking queries to entities in a knowledge base. However, these methods do not fully explore the importance of auxiliary signals in query logs, i.e. the structural language patterns (contextual word patterns) in queries and the clicked domains from relevant web URL results. For example, given the query "menu of Purple Pig" and a user's clicked domain URL "yelp.com", both the pattern "menu of" and the clicked URL help predicting "Purple Pig" as a local restaurant entity. Therefore, knowing the types of these important signals can help mining entity types from query logs more effectively.

In this paper, we model search query logs into a bipartite graph to encode relations between entities and important signals. Two kinds of nodes, entities and their auxiliary information, are contained in the constructed bipartite graph shown in Figure 1 (b). With such a bipartite graph, we can take advantage of the encoded relations [35] to learn entity types. Moreover, the type information can also be assigned to auxiliary nodes, thereby helping disambiguating entities via user-generated texts (*e.g.*, contextual words) and user feedbacks (*e.g.*, clicked URLs). In this paper, we apply a graph-based **L**abel **P**ropagation (LP) method to simultaneously learn types of both entities and auxiliary signals. Figure 1 (c) shows the steps of LP in an intuitive way. Given a small number of prior-known entities, the types of these entities are first propagated to the connected auxiliary nodes, and then the types are propagated back from auxiliary nodes to unknown entities. Despite the simple idea, mining entity types from the built graph is still a challenging task due to the following reasons:

- Queries are short texts, and information related to entities is usually very sparse. It is non-trivial to explore the hidden connections among entities and auxiliary information in search logs.
- Large amounts of irrelevant information exists in search logs, bringing noise in detecting entity types. It is difficult to discover and remove such noisy information from search logs.

In order to address these two issues, we propose an **E**nsemble framework based on **L**abel **P**ropagation (ELP) to simultaneously learn types of both entities and auxiliary signals. Specifically, we design two separate strategies to fix the problems of sparsity and noise in query logs, respectively.

In summary, our contributions are as follows:

- We represent query logs as a bipartite graph about entities and their auxiliary signals. We leverage such interconnected relationships between entities and their auxiliary signals to learn both entity types and auxiliary node types together.
- We propose an **E**nsemble framework based on **L**abel **P**ropagation (ELP) and design two separate strategies in ELP to effectively learn node types from search query logs.
- We conduct extensive empirical studies on search logs from a real-world search engine to demonstrate the effectiveness of the proposed ELP framework. In addition, some case studies show that ELP can learn important word patterns for different types of entities, as well as disambiguating entities via the connected auxiliary informaiton.

2. BACKGROUND

In this section, we first introduce several related concepts and notations. Then, we will formally define the problem of

learning node types from a bipartite entity-auxiliary graph extracted from query logs.

Definition 1. **A Bipartite Entity-Auxiliary (EA) Graph**: A bipartite entity-auxiliary (EA) graph is represented as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is the set of nodes (objects), including two types of objects, i.e., entities $E = \{e_1, ..., e_M\}$ and auxiliary signals $A = \{a_1, ..., a_N\}$. $\mathcal{E} \subseteq E \times A$ is the set of links (relations) between the nodes in V, which involves the *associatedWith* link between entities and auxiliary signals. Let $\mathcal{W}$ denote an $M \times N$ weight matrix, in which element w_{ij} equals the frequency associating e_i and a_j.

Figure 1 (b) shows an example of a bipartite EA graph extracted from the search query logs in Figure 1 (a). Three entities are connected with six auxiliary signals, including four contextual words and two clicked URLs.

In EA graph, each entity has at least one type (label) in reality. We assume there are K labels for entities ($K \geq 2$) and represent entity types as $\mathcal{Y} \in \mathbb{R}^{M \times K}$. $y_{ik} \in \mathcal{Y}$ is a non-negative real number indicating the probability that entity e_i belongs to label k. In practice, a small set of entities (seed entities) in the graph may be manually labeled with their types. We denote the labeled entity set as E_L. In Figure 1 (b), both "New York" and "Taylor Swift" are considered as seed entities. We use $\mathcal{Y}^0$ to denote an instantiation of $\mathcal{Y}$ that is consistent with the seed labels. Given an entity $e_i \in E_L$ with n labels ($n \geq 1$), we set y^0_{ik} as $1.0/n$ if e_i has label k, otherwise 0. Given the entity $e_i \in E \backslash E_L$ without labels, we have $y^0_{ik} = 0$ for any label k.

From the existing search logs D, we can extract a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and get $\mathcal{Y}^0$ according to some seed entities E_L. Our goal is to learn entity types $\mathcal{Y}$ from $\mathcal{G}$. Since the auxiliary nodes can also carry labels with them to indicate the important interconnections between entities and the auxiliary signals, another goal is to assign labels to those auxiliary nodes in $\mathcal{G}$. We use $\mathcal{Z} \in \mathbb{R}^{N \times K}$ as labels of auxiliary nodes, where the element z_{jk} is a non-negative real number indicating the probability that a_j relates to label k. Thus our ultimate goal becomes to estimate $\mathcal{Y}$ and $\mathcal{Z}$ given $\mathcal{G}$ and E_L. In order to solve this problem, we apply a graph-based **L**abel **P**ropagation (LP) method to leverage these important auxiliary signals via their connections with the target entities as shown in Figure 1 (c).

3. PROPOSED METHOD

In this section, we propose an **E**nsemble framework based on **LP** (ELP) to simultaneously learn types of both entities and auxiliary signals from query logs. Before proceeding, we first introduce how to build the entity-auxiliary graph from real-world search logs.

3.1 Graph Construction

Given search logs, we first have to extract entities from queries. Several methods are applied to find entities in this paper. First, we use a part-of-speech tagger [1] to extract contiguous words of proper nouns, common nouns and capitalized words [16, 17] to form noun phrases. Second, we match the extracted noun phrases according to a dictionary of entities built from knowledge bases, such as Wikipedia, Freebase and Yelp. We do not use the type information in those knowledge bases. We assume that the types of entities are unknown in the experiments. These methods help us detect entities in high precision. Besides, we can use a more complex model in [10] to identify the entity and the background part (i.e., contextual words). In the example of the search logs in Figure 1 (a), we extract "New York", "Maxwell" and "Taylor Swift" as entities. Thus, "home sales", "real estate", "albums" and "songs" are considered as contextual words. In our experiments, we use both the uni-gram and binary-grams of contexts as auxiliary nodes. The stop-word nodes are removed from our graph.

In search logs, clicked URLs are also very important for learning entity types. Since each clicked URL may have several levels of domain names to point to a certain webpage, there will be too many redundant nodes of clicked URLs in the constructed graph. Therefore, we group a set of URLs into a single auxiliary node if they have exactly the same top- and second-level domain names. In Figure 1 (a), we only show the first two domain names for the clicked URLs. We use the frequencies of entities and auxiliary nodes appearing together in the query logs as weights of corresponding edges.

Such a bipartite graph helps encode relations between entities and important auxiliary signals from search query logs. We can take advantage of the encoded relations to discover entity types by applying the graph-based LP method. However, directly applying LP may not be satisfying due to the following issues in query logs:

1. Queries are short texts, and information related to entities is usually very sparse. LP may not propagate labels adequately. Therefore, it is necessary to explore the hidden connections in EA graph.

2. Large amounts of irrelevant information exists in search logs, bringing noise in detecting entity types. LP may propagate errors out and enlarge the error information due to the noise. Hence, it is imperative to discover and remove such noisy information from the EA graph.

In the following, we first focus our attention on how to apply LP on the built graph to learn types of both entities and auxiliary signals simultaneously. Then we introduce two separate strategies LPA and LPD to address the problem of sparsity and noise in the EA graph respectively. After that, we describe the proposed ELP framework that takes advantage of the LPA and LPD strategies.

3.2 Methodology

3.2.1 The LP Method

The problem of learning with labeled and unlabeled data from graphs has been investigated in [36, 37, 18, 33, 11]. The objective and algorithm of the LP method used in this paper are heavily influenced by the works of [36, 18]. Given the collection of search log data D, we can extract an entity-auxiliary graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a weight matrix $\mathcal{W}$ as introduced in Section 2. With a small set of seed entities E_L, we can initialize $\mathcal{Y}^0$. Our goal is to automatically estimate $\mathcal{Y}$ for entities and $\mathcal{Z}$ for auxiliary nodes according to $\mathcal{W}$ and $\mathcal{Y}^0$. We define a normalized frequency matrix as follows:

$$\mathcal{N} = \mathcal{D}^{-1/2}\mathcal{W}, \tag{1}$$

where $\mathcal{D}$ is a diagonal matrix and each element $d_{ii} \in \mathcal{D}$ is the sum of all the elements in the ith row (or column) of $\mathcal{W}\mathcal{W}^\top$. Intuitively, d_{ii} can be interpreted as the volume of all length-of-two paths that start at e_i. The reason we use such a normalization is to guarantee the convergence of LP as shown in [18].

Algorithm 1 The LP algorithm

Input: Search query log D, a set of seed entities E_L and a trade-off parameter α
Output: Label matrices $\mathcal{Y}$ and $\mathcal{Z}$

```
//graph construction step
Build an entity-auxiliary graph G = (V, E) from D
//initialization step
Initialize Y as Y^0 according to E_L
Compute the weight matrix W from G
Compute N from W according to Equation (1)
//iterative computation step
while NOT converged do
    //propagation step from entities to auxiliary nodes
    Compute Z^t according to Equation (2)
    //propagation step from auxiliary nodes to entities
    Compute Y^t according to Equation (3)
end while
Normalize Y and Z according to Equation (5)
```

With the above definitions and notations, LP iteratively updates $\mathcal{Y}$ and $\mathcal{Z}$. For the t-th iteration, it first propagates the types of entities to the connected auxiliary nodes:

$$\mathcal{Z}^t = \mathcal{N}^\top \mathcal{Y}^{t-1}. \tag{2}$$

Then it propagates the types back to entities from the auxiliary nodes as follows:

$$\mathcal{Y}^t = \alpha \mathcal{N} \mathcal{Z}^t + (1-\alpha)\mathcal{Y}^0, \tag{3}$$

where α is a parameter to trade off the label consistency between the intrinsic graph structure and the seed entities. It has been shown in [18] that the sequence of $\mathcal{Y}^t$ asymptotically converges to:

$$\mathcal{Y}^* = (1-\alpha)(\mathbf{1} - \alpha \mathcal{D}^{-1/2}\mathcal{W}\mathcal{W}^\top \mathcal{D}^{-1/2})^{-1}\mathcal{Y}^0. \tag{4}$$

The time complexity of LP is $O(T|\mathcal{E}|)$, where T is the iteration number and $|\mathcal{E}|$ is the number of connections in the EA graph. Through our experiments, the algorithm converges after no more than 20 rounds in most cases. The LP method is summarized in Algorithm 1.

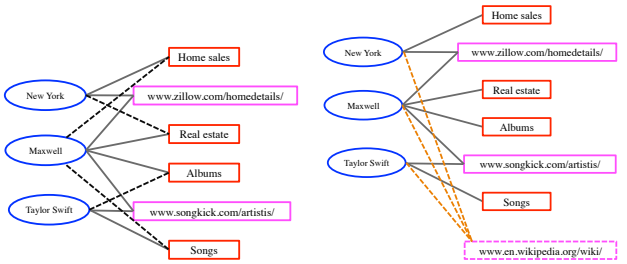
Once $\mathcal{Y}$ and $\mathcal{Z}$ are obtained, we normalize their elements to get the posterior probabilities $p(k|e_i)$ for $i = 1, ..., M$ and $p(k|a_j)$ for $j = 1, ..., N$ as follows:

$$\begin{aligned} p(k|e_i) &= y_{ik} / \sum_{l=1}^{k} y_{il}, \\ p(k|a_j) &= z_{jk} / \sum_{l=1}^{k} z_{jl}. \end{aligned} \tag{5}$$

3.2.2 The Proposed LPA Strategy

Directly applying LP may not be satisfying because the connections extracted from query logs are very sparse and LP cannot propagate labels adequately. Therefore, we propose a strategy LPA (**L**abel **P**ropagation after **A**dding more connections) to explore the hidden connections in the EA graph. We take advantage of the word2vec tool [4] to connect entities with more contextual words and help the LP model propagate labels more effectively.

[4] `https://code.google.com/p/word2vec/`



(a) The LPA strategy (b) The LPD strategy

Figure 2: Two seperate intuitive examples of the updated graphs obtained from the LPA and LPD strategies. The dashed black edges in (a) represent the hidden connections explored by LPA. In (b), the dashed box of the auxiliary node means that the node is multi-type, and the dashed orange edges represent the connections we should get rid of according to LPD.

The intuition behind LPA is that, if one entity e connects with one auxiliary node a_1 and a_1 has a high similarity with another auxiliary node a_2, we should connect e with a_2 to expand the connections in the bipartite EA graph. Hence, we need to measure the similarities among auxiliary nodes first. In this paper, we focus on contextual words and measure their similarities according to the semantic meanings by the word2vec tool. The word2vec tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words [23, 24, 25]. By calculating the distance between two vector representations, we can obtain the similarity value for two words. Hence, given the auxiliary node set $A = \{a_j\}_{j=1}^N$, we could get a similarity matrix $\mathcal{S} \in \mathbb{R}^{N \times N}$, where each element $s_{ij} \in \mathcal{S}$ denotes the similarity value between a_i and a_j. The above exploration of connections can be formulated as follows:

$$\mathcal{W}_\mathcal{A} = \mathcal{W} \times \mathcal{S}, \tag{6}$$

where $\mathcal{W}_\mathcal{A}$ is the updated weight matrix according to LPA. Intuitively, $w_{ij} \in \mathcal{W}_\mathcal{A}$ can be interpreted as the weight aggregation of all length-of-two paths from e_i to a_j via every $a_{j'} \in A$. Given the search logs in Figure 1 (a), we show an intuitive example of the updated graph according to LPA in Figure 2 (a). We assume that the contextual words "home sales" and "real estate" are very similar so we connect "New York" with "real estate" in Figure 2 (a). With such a denser graph, we could run the LP algorithm to propagate labels more effectively. We denote the node types learned from LPA as $\mathcal{Y}_\mathcal{A}$ and $\mathcal{Z}_\mathcal{A}$ for entities and auxiliary nodes, respectively.

3.2.3 The Proposed LPD Strategy

Another issue of directly applying LP is that noise may exist in the built EA graph so that LP may propagate errors out and enlarge the error information. Therefore, we propose a strategy LPD (**L**abel **P**ropagation after **D**eleting noisy nodes) to discover and get rid of noisy information in the EA graph.

The basic idea of LPD is to discover some multi-type auxiliary nodes and delete them with their corresponding connections in the constructed EA graph. Here multi-type nodes mean contextual words or clicked URLs that cover several types of entities. For example, "picture" relates to

several entity types, such as media, location, and person. Hence, it is not informative to take such contextual word into consideration. We apply a similar approach in [2] to get rid of some multi-type auxiliary nodes and update the graph accordingly. Specifically, we start by calculating the similarity (e.g., cosine similarity) between two entities according to the bag-of-word representations of their auxiliary nodes. Low similarity pairs are more likely to represent entities with different types. Hence, auxiliary nodes involved with such entities are not likely to be very specific. So we can consider low similarity pairs as voters and let the auxiliary nodes be the candidates. Each pair votes for its auxiliary nodes they share. The more votes an auxiliary node gets, the higher probability of multi-type it is. We can then apply a threshold to get rid of some auxiliary nodes and update the EA graph accordingly. Figure 2 (b) gives an intuitive example of the updated graph according to LPD. In this figure, we assume the clicked URL "www.en.wikipedia.org/wiki/" is a multi-type node and delete it with its corresponding edges from the graph. Then we can run LP on such a cleaner graph so that the error information can be propagated out as little as possible. We denote the node types learned from LPD as $\mathcal{Y}_{\mathcal{D}}$ and $\mathcal{Z}_{\mathcal{D}}$ for entities and auxiliary nodes, respectively.

3.2.4 The Proposed ELP Framework

Given the proposed LPA and LPD strategies, we can simply combine them together to derive another two strategies, LPAD and LPDA. LPAD updates the graph by first adding more connections and then deleting noisy nodes. The node types learned from LPAD are denoted as $\mathcal{Y}_{\mathcal{AD}}$ and $\mathcal{Z}_{\mathcal{AD}}$ for entities and auxiliary nodes, respectively. LPDA updates the graph in an opposite way, i.e., first deleting noisy nodes and then adding connections based on the remaining nodes. We denote the node types learned from LPDA as $\mathcal{Y}_{\mathcal{DA}}$ and $\mathcal{Z}_{\mathcal{DA}}$ for entities and auxiliary nodes, respectively.

Since each strategy has its advantage, we propose an **E**nsemble framework based on **LP** (ELP) to combine them together and maximize the margin [12]. We run each strategy separately and select the best one as the final result for each node as follows:

$$\begin{aligned} \mathcal{Y} &= max\{\mathcal{Y}_{\mathcal{A}}, \mathcal{Y}_{\mathcal{D}}, \mathcal{Y}_{\mathcal{AD}}, \mathcal{Y}_{\mathcal{DA}}\}, \\ \mathcal{Z} &= max\{\mathcal{Z}_{\mathcal{A}}, \mathcal{Z}_{\mathcal{D}}, \mathcal{Z}_{\mathcal{AD}}, \mathcal{Z}_{\mathcal{DA}}\}. \end{aligned} \tag{7}$$

In practice, we can also use the weighted results of the four strategies as the final solution. Since it would bring several weight parameters for these strategies, we calculate the results of ELP according to Equation (7) for simplicity in the experiments.

4. EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the proposed ELP framework. After introducing the datasets and the experimental settings, we compare different baseline methods.

4.1 Data Processing

We collect a large set of click-through data (denoted as a system set) over a continuous period of time from a real-world search engine. Then a small number of click-through data are sampled from the system set and denoted as a gold set. We manually labeled entities from queries of the gold set with correct types. The labeled data are only used for seed selections and performance evaluations in the experiments. The basis statistics of these two datasets are shown in Table 1.

Table 1: Statistics of the collected query data.

Dataset	#Queries	#Clicked URLs
Gold	16,903	2,369,618
System	217,223,831	1,556,499,551

Table 2: Statistics of the entity-auxiliary graphs. "EC Links" means the entity-context links and "EU Links" means the entity-URL links.

		#Auxiliary nodes		#Links	
Dataset	#Entities	#Contexts	#URLs	#EC Links	#EU Links
Gold	934	1,445	10,059	3,323	36,475
System	10,722	39,279	24,107	514,489	221,668

Table 3: Distributions of labels for entities.

	Labels		
Dataset	Local	Media	Person
Gold	36.6%	36.4%	27.0%
System	33.5%	33.7%	32.8%

In the experiments, we focus on 3 target types of classes, namely Local, Media and Person. By following the extraction rules in Section 3.1, we get entities and related auxiliary signals belonging to these 3 target types. In order to build a compact and reliable graph, we apply a threshold to get rid of some infrequent nodes. For example, we set the threshold as 1 for the gold dataset and filter out those nodes appearing only 1 time. The basic statistics of nodes and links in the entity-auxiliary graphs are represented in Table 2. The distributions of the 3 target labels for entities are shown in Table 3.

4.2 Compared Methods

In order to show that the LP model fits the constructed graph very well, we compare LP with several traditional classification methods. Given the bipartite EA graph, we consider the connected auxiliary nodes as features for each entity and the frequencies (the edge weights) as feature values. We focus on the following methods:

- SVM: **S**upport **V**ector **M**achine (SVM) is a popular classification method. Since non-linear RBF kernel is widely used in SVM models, we apply SVM (RBF) on the constructed bipartite EA graph.
- KNN: We compare with the **K**-**N**earest **N**eighbors method (KNN) to show the effectiveness of the LP method. We denote the KNN method using n neighbors as KNN-n.
- DT: The **D**ecision **T**ree method (DT) is applied on features extracted from the EA graph.
- NB: We apply the **N**aive **B**ayers (NB) on features of entities extracted from the built EA graph.
- LP: The original **L**abel **P**ropagation method (LP) is applied on the EA graph without the feature extraction.

In addition, in order to show the effectiveness of the proposed ELP framework, we compare with different variations of the LP model. Since both contextual words and clicked URLs can be considered as auxiliary information for entities

in search query logs, we can construct three different bipartite graphs. They are the **E**ntity-**C**ontext (EC) graph, the **E**ntity-clicked **U**RL (EU) graph and the **E**ntity-**Auxiliary** (EA) graph. The EA graph considers both the contexts and clicked URLs as the auxiliary information in search query logs, so it contains more information than the EC and EU graphs. We can apply our proposed strategies on these different graphs and we summarize them as follows:

- LP: There are three versions for LP. They are LP (C), LP (U) and LP (A). LP (C) focuses on the EC graph. Similarly, LP (U) applies on the EU graph and LP (A) runs on the EA graph.

- LPA: We derive two baselines from LPA. The first one is LPA (C), which applies the LPA strategy on the bipartite EC graph. The other one is LPA (A) on the EA graph. The effectiveness of using more auxiliary information from search query logs can be demonstrated by comparing LPA (C) with LPA (A). Since the LPA strategy focuses on expanding the hidden connections between entities and contextual words, we cannot apply it on the EU graph.

- LPD: Three baselines can be generated from LPD. They are LPD (C), LPD (U) and LPD (A) on the EC, EU and EA graphs respectively. In particular, LPD (C) means that we first get rid of the top k multi-type contextual words and then apply the LP method on the updated EC graph. LPD (U) is derived in a similar way. For LPD (A), we first find the top k multi-type contextual words from the EC graph and the top k multi-type clicked URLs from the EU graph. After that we delete these contextual words and clicked URLs from the EA graph and apply LP on the updated graph. In this way, we guarantee that the most ambiguous auxiliary nodes (both contextual words and clicked URLs) are removed from the EA graph.

- LPAD: There are two baselines based on LPAD. They are LPAD (C) and LPAD (A). Specifically, LPAD (C) contains three steps: (1) expand connections and update the EC graph; (2) delete multi-type nodes in the denser EC graph; (3) run LP on the latest EC graph. LPAD (A) executes in a similar way.

- LPDA: We generate LPDA (C) and LPDA (A) from LPDA. LPDA (C) updates the EC graph by first deleting multi-type contextual words and then expanding hidden connections between entities and the remaining contextual words. LPDA (A) updates the EA graph in a similar way.

- ELP: We also derive two versions for ELP. They are ELP (C) and ELP (A). ELP (C) combines LPA (C), LPD (C), LPAD (C) and LPDA (C) in an ensemble way while ELP (A) ensembles LPA (A), LPD (A), LPAD (A) and LPDA (A) together to achieve a better performance.

For a fair comparison, we use the same parameter settings for the baselines related to the LP method. Specifically, we test with different α values for LP and find that $\alpha \in (0.5, 0.9)$ yields similar good results. So we set the parameter α to be 0.75 as in [18]. In order to get the similarities among contextual words, we use a pre-trained vectors[5] on about 100 billion words and phrases from various news articles. For the number of auxiliary nodes that should be deleted, we set it to be 10 in the experiments. In addition, we use SVM (RBF) with optimized parameters and other traditional classifiers with default parameters in our experiments. For each node, we can get a list of non-negative real numbers from LP indicating the posterior probabilities that the node relates to a label. We clamp these probabilities to 0/1 values for simplicity.

In order to evaluate the results, we focus on the labeled data and use accuracy and weighted average of the F1 score of each class (abbreviated as "weighted-F1") as the performance measures for entities. Weighted-F1 means that we calculate the F1 score for each label and find their average value weighted by the number of true instances for each label. This metric takes the label imbalance into consideration. For an entity with multi-labels, if the learned label matches with one of its multiple labels, we consider it as a correct prediction. Since we do not have ground truth for the auxiliary information, we will not present the quantitative analysis on the auxiliary information. We only show some qualitative analysis in Section 4.4. In the experiments, we randomly select a certain portion (e.g., 10%) of the entities as seeds for 10 times and report the average performances for models related to LP. We use the same seed entities as the training data for the traditional classification models.

4.3 Performance Evaluation

In this subsection, we show the performances of the proposed ELP framework. We first demonstrate that how the LP method takes advantage of the constructed bipartite EA graph compared with some traditional classification models. Due to space limit, we only show the performances on the EC graph of the gold dataset in Table 4. Similar performances can be obtained for other graphs.

It can be observed from Table 4 that LP consistently outperforms other classification methods on accuracy and weighted-F1 scores for different amounts of seed entities (training data). It illustrates that the constructed graph helps the LP method propagate the label information out very well. Since all the other classifiers ignore the graph structure, important information may be missing and the performances are not so well compared with the LP method that takes advantage of the graph structure. In addition, when the amount of seed entities increases, the performances become better for almost all classifiers except the SVM method with the RBF kernel. It seems that more training data does not help SVM (RBF) very much. However, in reality, more seed entities means more annotations and human labelings. With large volumes of new queries, extracting such supervised information from search query logs can be very expensive and time consuming. In Table 4, LP can only achieve 45% of accuracy when 1% of data are selected as seeds. The performance should be improved if we explore the hidden connections and get rid of multi-type nodes in the constructed graph as in the proposed ELP framework. So in the following, we focus on the gold dataset with 1% of seed entities to show the effectiveness of ELP.

[5]freebase-vectors-skipgram1000-en.bin.gz. It can be downloaded from `https://code.google.com/p/word2vec/`.

Table 4: Average performances with different amounts of seed entities for 10 times on the EC graph of the gold dataset. The results are reported as "average performance + (rank)". "↑" indicates that the larger the value the better the performance.

Metric	Method	Percentages of seed entities					
		1%	10%	20%	30%	40%	50%
Accuracy ↑	SVM (RBF)	0.3720 (3)	0.3876 (5)	0.3853 (5)	0.3763 (5)	0.4139 (5)	0.4206 (5)
	KNN-1	0.3610 (5)	0.4617 (4)	0.5100 (4)	0.5406 (4)	0.5591 (4)	0.5779 (4)
	DT	0.3688 (4)	0.4995 (3)	0.5568 (3)	0.5818 (3)	0.5989 (3)	0.6036 (3)
	NB	0.3752 (2)	0.5713 (2)	0.6106 (2)	0.6317 (2)	0.6271 (2)	0.6318 (2)
	LP	**0.4544** (1)	**0.6959** (1)	**0.7204** (1)	**0.7313** (1)	**0.7350** (1)	**0.7386** (1)
Weighted-F1 ↑	SVM (RBF)	0.2278 (3)	0.2445 (5)	0.2353 (5)	0.2206 (5)	0.2858 (5)	0.2918 (5)
	KNN-1	0.2248 (4)	0.4190 (4)	0.4987 (4)	0.5304 (4)	0.5522 (4)	0.5755 (4)
	DT	0.2231 (5)	0.4608 (3)	0.5404 (3)	0.5702 (3)	0.5903 (3)	0.5977 (3)
	NB	0.2789 (2)	0.5643 (2)	0.6066 (2)	0.6295 (2)	0.6233 (2)	0.6293 (2)
	LP	**0.3904** (1)	**0.6910** (1)	**0.7188** (1)	**0.7308** (1)	**0.7346** (1)	**0.7385** (1)

The results of different methods based on LP are shown in Table 5. It can be observed that ELP (A) outperforms other baseline methods on both accuracy and weighted-F1 and ELP (C) can also achieve a very good performance. ELP (A) outperforms ELP (C) with an improvement of 21% on the accuracy. It shows that more auxiliary nodes help ELP achieve a better performance on learning entity types from search query logs.

In particular, due to the noisy information in search query logs, directly applying the LP method on the constructed EA graph may reduce the performance as shown in Table 5. However, the results can be improved if we better build the graph as introduced in LPA and LPD. We can observe that the performance of LPA (LPD) on the EA graph are better than those on the EC and EU graphs. It demonstrates that using more high-quality auxiliary nodes can provide more important information and facilitate the process of LPA or LPD. Moreover, compared with the original LP method, both LPA and LPD can improve the performances for all the constructed bipartite graphs (i.e., EC, EU and EA). For example, LPA (A) significantly outperforms LP (A) with improvements of 45% and 104% on accuracy and weighted-F1, respectively. Furthermore, LPA seems more powerful than LPD on both the EC and EA graphs. It shows that exploring hidden connections among the sparse graph plays a more important role in learning entity types from search query logs.

Though LPA and LPD perform better than LP, our proposed ELP framework achieves better results than the LPA and LPD strategies. Specifically, ELP (A) outperforms LPA (A) with an improvement of 22% on the weighted-F1 score. In addition, ELP also performs better than LPAD and LPDA with an average improvement of 18% on the weighted-F1 score as shown in Table 5. It implies that ELP can maximize the effectiveness of combining different strategies together. Simply combining LPA and LPD together (e.g., LPAD and LPDA) may not make full use of the operations of adding more connections and deleting the noisy nodes.

In summary, with the help of exploring hidden connections and getting rid of noisy nodes, the proposed ELP framework can achieve an accuracy of 68% on the EA graph with only 1% of entities as seeds. From Table 4, we can see that the LP method needs around 10% of seed entities to get the same accuracy score. Therefore, ELP can help significantly reduce the cost of human labeling in learning entity types from search query logs.

We further show the effectiveness of the proposed ELP framework on the larger system set. Only 0.1% of seed entities are used in the experiments to test the power of ELP. Since we only labeled entities in the gold set and these entities are included in the system set, we calculate the accuracy and weighted-F1 scores on the labeled entities in the system set. The performances are presented in Table 6. We can get similar observations for the system set.

Table 5: Average performances with 1% of seed entities for 10 times on the gold dataset. The results are reported as "average performance + (rank)". "↑" indicates that the larger the value the better the performance.

Graph	Method	Metric	
		Accuracy ↑	Weighted-F1 ↑
EC	LP (C)	0.45 (8)	0.39 (8)
	LPA (C)	0.53 (5)	0.49 (5)
	LPD (C)	0.50 (7)	0.45 (6)
	LPAD (C)	0.53 (5)	0.49 (5)
	LPDA (C)	0.54 (4)	0.50 (4)
	ELP (C)	0.56 (3)	0.56 (2)
EU	LP (U)	0.39 (10)	0.26 (10)
	LPD (U)	0.52 (6)	0.44 (7)
EA	LP (A)	0.40 (9)	0.27 (9)
	LPA (A)	0.58 (2)	0.55 (3)
	LPD (A)	0.53 (5)	0.45 (6)
	LPAD (A)	0.58 (2)	0.55 (3)
	LPDA (A)	0.58 (2)	0.55 (3)
	ELP (A)	**0.68** (1)	**0.67** (1)

Table 6: Average performances with 0.1% of seed entities for 10 times on the system dataset. The results are reported as "average performance + (rank)". "↑" indicates that the larger the value the better the performance.

Graph	Method	Metric	
		Accuracy ↑	Weighted-F1 ↑
EC	LP (C)	0.44 (6)	0.32 (8)
	LPA (C)	0.45 (5)	0.37 (5)
	LPD (C)	0.50 (4)	0.41 (4)
	LPAD (C)	0.45 (5)	0.37 (5)
	LPDA (C)	0.45 (5)	0.37 (5)
	ELP (C)	0.57 (2)	0.56 (2)
EU	LP (U)	0.40 (9)	0.27 (10)
	LPD (U)	0.43 (7)	0.34 (7)
EA	LP (A)	0.42 (8)	0.30 (9)
	LPA (A)	0.55 (3)	0.51 (3)
	LPD (A)	0.44 (6)	0.36 (6)
	LPAD (A)	0.55 (3)	0.51 (3)
	LPDA (A)	0.55 (3)	0.51 (3)
	ELP (A)	**0.59** (1)	**0.58** (1)

4.4 Case Study

In this subsection, we present several case studies to show the effectiveness of the proposed ELP framework. We first show the most popular auxiliary nodes with their labels learned from ELP and explain how such auxiliary information can help detect new entities from search query logs. Then we give some examples of the hidden connections we explored in LPA. After that, we list several multi-type auxiliary nodes discovered in LPD. At the end, we will analyze

Table 7: The most popular auxiliary information learned from ELP (A) on the gold dataset.

Label	Top k	Auxiliary Information	
		Contextual Word	Clicked URL
Local	1	high school	hamptoninn3.hilton.com/en/
	2	Sale in	www.homes.com/Real_Estate
	3	IL	www.wunderground.com/weather-forcast/
	4	Orlando	www.accuweather.com/en/
	5	Coupons	www.city-data.com/city/
Media	1	Watch	www.tv.com/shows/
	2	Cast	tv.yahoo.com/shows/
	3	Season	tv.msn.com/tv/
	4	Songs	tv.yahoo.com/news/
	5	Episode	yidio.com/show/
Person	1	Biography	www.theguardian.com/film/
	2	Site	marquee.blogs.cnn.com/2014/
	3	Naked	images.fanpop.com/images/
	4	Nude	movies.msn.com/movies/
	5	Divorce	www.tmz.com/2014/

the potential to disambiguate multi-label entities in the proposed ELP framework.

4.4.1 Popular Auxiliary Information

Given the gold dataset, we first randomly select 1% of entities as seeds and run ELP on the EA graph. Then we apply a threshold (e.g., larger than 10) to select the most popular contextual words and clicked URLs separately. After that, we group the auxiliary nodes according to their learned types and rank nodes in each group by the learned probability value in a decreasing order. Due to space limit, we only show the top 5 related auxiliary information learned from ELP (A) for the gold dataset in Table 7. We can observe that people care about the education, real estate and weather very much when they search about local entities.

4.4.2 New Entity Discovery

With the learned types of auxiliary nodes in Table 7, we can discover new entities easily. For example, if a new TV series is released, we can detect it as a new entity when people search with the word "episode". In our experiments, we consider those entities appearing few times (≤ 2) as new entities and ELP can learn their types correctly. For instance, "Pogo" (an online game) appears only twice and ELP detects it as a media entity because its connected contextual words are "app" and "ipad". However, "Pogo" refers to a musical artist and a comic strip in Wikipedia. Therefore, the ELP method helps us discover "Pogo" as a new media entity, and we can add such information to the current knowledge graph.

4.4.3 Hidden Connections

Now we analyze how the hidden connections we explored help LPA fully propagate labels. We focus on those entities with few connected contextual words and give some examples of the entities with their hidden connections we discovered from the gold dataset as shown in Table 8. It can be observed that the hidden connections provide complementary and discriminative information for learning entity types. For example, given the entity "Big Brother" and its connected contextual word "CBS", the word "showtime" help predict "Big Brother" as an entity of media with more confidence. In addition, the hidden connections can help learn entity types more correctly. Take the entity "George Clooney" as an instance. The connected contexts "movie" and "new" are a bit ambiguous and "George Clooney" may be considered as an entity of media because "movie" is more related to media.

Table 8: Some entities and their connected contextual words from query logs. The hidden connected contextual words explored by LPA are also shown in the last column.

Label	Entity	Contextual Word	
		Existing word	Hidden word
Media	Big Brother	CBS	Showtime
Person	George Clooney	Movie, New	Wedding
Media	Haunted	Taylor Swift	Music, Video
Person	Sarah McLachlan	Song	Single
Local	Starbucks	free, coffee	open, free shipping, zip code

Table 9: The multi-type auxiliary information discovered from LPD on the gold dataset.

Top k	Auxiliary Information	
	Contextual Word	Clicked URL
1	Online	en.wikipedia.org/wiki/
2	Lyrics	geo.yahoo.com/t/
3	News	video.search.yahoo.com/video/
4	Free	video.search.yahoo.com/search/
5	Hotels	images.search.yahoo.com/search/
6	Movie	search.yahoo.com/
7	Newspaper	news.search.yahoo.com/
8	Weather	www.imdb.com/title/
9	Map	www.youtube.com/
10	Jobs	news.yahoo.com/photos/

However, if we connect "George Clooney" with "wedding", we can easily learn that "George Clooney" is an entity of person. Therefore, exploring the hidden connections in query logs can help learn entity types more accurately.

4.4.4 Multi-type Auxiliary Information

Here we analyze the effectiveness of discovering and removing the multi-type auxiliary information from LPD. We list the 10 most ambiguous auxiliary nodes discovered from the gold dataset in Table 9. It can be observed that these auxiliary nodes are related to different types of entities and getting rid of them can help propagate labels more accurately in LP. For example, the contextual word "Online" can refer to the online information of a place, a movie and a person. In addition, the clicked URL "en.wikipedia.org/wiki/" is related to a navigational website that contains diverse information. It is difficult to detect the type of an entity if such URLs are connected with the target entity. Therefore, we identify the ambiguous auxiliary nodes and remove them from our constructed graph.

4.4.5 Entity Disambiguation

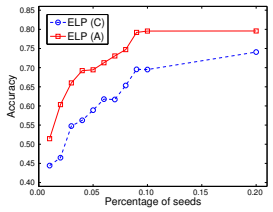
According to the proposed ELP framework, we can get a list of non-negative real numbers indicating the probabilities that an entity relates to a type. We clamp these probabilities to 0/1 values for simplicity in the performance evaluation. However, in reality, a lot of entities have more than one type. In this subsection, we analyze the potential of ELP to disambiguate multi-label entities.

We first analyze those entities without ambiguity. From our experimental results, ELP gives high probability values to the types of these entities. For example, given the entity "Gold Digger", ELP learns a probability value of 0.94 for the media type. Similarly, "'Walt Disney World" has a probability value of 0.80 for the local type.

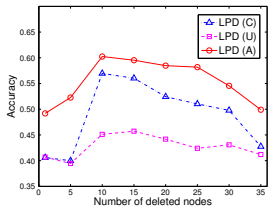
We also focus on those multi-type entities to see whether it is easy to disambiguate them in our experiments. We take the entity "Maxwell" as an example. ELP learns it as a local entity with a probability of 0.48 and a person entity with a probability of 0.37. Table 10 shows some connected

Table 10: Some auxiliary nodes for a multi-label entity "Maxwell". We group them according to their types learned from ELP.

Label	Top k	Auxiliary Information: Contextual Word	Auxiliary Information: Clicked URL
Local	1	Real estate	www.zillow.com/homedetails/
	2	Weather forecast	www.realtor.com/realestateandhomes/
	3	Map of	www.healthgrades.com/physician/
	4	Sale in	www.homes.com/Real_Estate/
	5	Homes for sale	www.wunderground.com/weather-forecast/
Person	1	New album	www.allmusic.com/artist/
	2	Discography	www.jango.com/music/
	3	Songs by	www.oldies.com/artist-songs/
	4	Albums	www.songkick.com/artists/
	5	Full album	www.mtv.com/artists/



(a) Effect of seed entites



(b) Effect of deleted nodes

Figure 3: Parameter analysis on the gold dataset.

auxiliary nodes for "Maxwell". We group them according to their labels learned from ELP and rank them by their probability values in a decreasing order. It can be observed that the auxiliary nodes for "Maxwell" as a local entity and a person entity are very different. Hence, ELP provides the potential for us to further split the entity node "Maxwell" into two nodes to better build the entity-auxiliary graph from search query logs.

4.5 Sensitivity Analysis

In this subsection, we assess the benefit of ELP with different amounts of seed entities. We focus on the EC and EA graphs extracted from the gold dataset and fix other parameters. Figure 3 (a) shows the classification accuracies. We find that the performances become better when we increase the seed entities. Moreover, the results stabilize when we use more than 10% of seed entities.

We also demonstrate the effect of the numbers of deleted nodes in Figure 3 (b). Here we fix 1% of seed entities and other parameters, but vary the numbers of deleted nodes. It can be observed that the best accuracy is achieved when we remove 10 multi-type auxiliary nodes. The noise may still exist if we get rid of too few nodes and the performance cannot be improved dramatically. However, information may be imcomplete if we delete too many nodes as shown in Figure 3 (b). So in our experiments, we set the number of deleted nodes as 10.

5. RELATED WORK

Entity extraction and classification have rapidly developed over the past few years [1, 5, 26]. Several methods are proposed to extract entities from web documents [3, 30] and the disambiguation of entities from news articles is studied in [14, 20]. In recent years, the extraction and classification of entities over query logs receive a lot of attention [27, 26, 16, 1, 17] and disambiguating entities in queries is also investigated in [6]. However, all these existing methods do not fully explore the importance of the auxiliary information related to entities. Our study is different since we encode entities and the important auxiliary information together into a bipartite graph and learn the types of both entities and auxiliary nodes simultaneously.

The graph-based label propagation method is also very popular in information retrieval tasks [18, 33]. Li et al. use click graphs, a bipartite-graph representation of click-through data from search query logs, to improve query intent classifiers [18]. Spam webpages are detected using the link structure of the click-through bipartite graph in [33]. It propagates spam scores iteratively between queries and URLs from a few seed pages/sites. In our work, we focus on learning entity types from search query logs, which differs from the task in [18, 33]. In addition, these models do not consider the sparsity and noise issues in search query logs. Our work proposes two separate strategies to explore hidden connections in the constructed bipartite graph and detect noisy information in query logs.

Besides the label propagation, many other learning methods, including Markov random walks [15], learning with local and global consistency [36, 8] and manifold regularization [4], are based on graphs. Furthermore, Chang et al. [7] proposed an unsupervised embedding scheme on graphs with heterogeneous components. Their method systematically captures network similarity between pairwise nodes by a deep learning framework. Though these models differ in their optimization objectives, they all share the same underlying assumption that if two samples are close in the intrinsic geometry of an input space, their conditional distributions will be similar.

The entity-oriented analysis of query data is also related to our work [13, 32]. For example, class attributes are extracted from search query logs for entities in [29, 28] as a complement source for existing knowledge bases. Based on the attributes, synonymous query intent templates are identified in [19]. In addition, entity-related search actions, annotations, and recommendation systems are studied in [22, 21, 34], respectively. However, our work is different from them since we study the problem of detecting entity types from search query logs.

6. CONCLUSION

In this paper, we study the problem of discovering entity types from search query logs. In order to take advantage of word patterns and user feedbacks (e.g., clicked URLs) from query logs, we construct a bipartite graph to encode entities and the important auxiliary information together. Based on this, the framework ELP is proposed to simultaneously learn types of both entities and auxiliary signals. In order to effectively learn node types, two separate strategies LPA and LPD are proposed and incorporated into ELP. Extensive empirical studies are conducted on real-world search logs to evaluate the effectiveness of the proposed ELP framework.

There are several interesting directions for future work. Since the constructed bipartite graph from search query logs reflects the most popular trending of entities, it can provide complementary information for the current knowledge bases. One direction of our future work is to explore the possibility of incorporating the built graph to the current knowledge bases. Another potential direction is to further disambiguate multi-label entities effectively based on the built bipartite graph, which is a hot but challenging problem in recent years.

Acknowledgement

This work is supported in part by NSF through grant CNS-1115234, Yahoo, and the Pinnacle Lab at Singapore Management University.

7. REFERENCES

[1] A. Alasiry, M. Levene, and A. Poulovassilis. Detecting candidate named entities in search queries. In *SIGIR*, 2012.

[2] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *ACM SIGKDD*, 2007.

[3] M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction for the web. In *IJCAI*, 2007.

[4] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7, 2006.

[5] K. Bellare, C. Curino, A. Machanavajihala, P. Mika, M. Rahurkar, and A. Sane. Woo: A scalable and multi-tenant platform for continuous knowledge base synthesis. *VLDB Endowment*, 6(11), 2013.

[6] R. Blanco, G. Ottaviano, and E. Meij. Fast and space-efficient entity linking in queries. In *WSDM*, 2015.

[7] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *ACM SIGKDD*, 2015.

[8] S. Chang, G.-J. Qi, C. C. Aggarwal, J. Zhou, M. Wang, and T. S. Huang. Factorized similarity learning in networks. In *ICDM*, 2014.

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

[10] N. Dalvi, M. Olteanu, M. Raghavan, and P. Bohannon. Deduplicating a places database. In *WWW*, 2014.

[11] C. Ding, T. Li, and D. Wang. Label propagation on k-partite graphs. In *ICMLA*, 2009.

[12] A. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *AAAI*, 1998.

[13] D. Hakkani-Tür, A. Celikyilmaz, L. Heck, and G. Tür. A weakly-supervised approach for discovering new user intents from search query logs. In *INTERSPEECH*, 2013.

[14] J. Hoffart, Y. Altun, and G. Weikum. Discovering emerging entities with ambiguous names. In *WWW*, 2014.

[15] M. Jaakkola and M. Szummer. Partially labeled classification with markov random walks. In *NIPS*, 2002.

[16] A. Jain and M. Pennacchiotti. Open entity extraction from web search query logs. In *ACL*, 2010.

[17] A. Jain and M. Pennacchiotti. Domain-independent entity extraction from web search query logs. In *WWW*, 2011.

[18] X. Li, Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *SIGIR*, 2008.

[19] Y. Li, B. Hsu, and C. Zhai. Unsupervised identification of synonymous query intent templates for attribute intents. In *CIKM*, 2013.

[20] Y. Li, C. Wang, F. Han, J. Han, D. Roth, and X. Yan. Mining evidences for named entity disambiguation. In *ACM SIGKDD*, 2013.

[21] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *VLDB Endowment*, 3(1-2), 2010.

[22] T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active objects: Actions for entity-centric search. In *WWW*, 2012.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR Workshop*, 2013.

[24] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[25] T. Mikolov, W. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, 2013.

[26] P. Pantel, T. Lin, and M. Gamon. Mining entity types from query logs via user intent modeling. In *ACL*, 2012.

[27] M. Paşca. Weakly-supervised discovery of named entities using web search queries. In *CIKM*, 2007.

[28] M. Paşca. Attribute extraction from conjectural queries. In *COLING*, 2012.

[29] M. Paşca and B. V. Durme. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI*, 2007.

[30] M. Pennacchiotti and P. Pantel. Entity extraction via ensemble semantics. In *EMNLP*, 2009.

[31] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *WWW*, 2010.

[32] J. Reisinger and M. Paşca. Fine-grained class label markup of search queries. In *ACL*, 2011.

[33] C. Wei, Y. Liu, M. Zhang, S. Ma, L. Ru, and K. Zhang. Fighting against web spam: a novel propagation method based on click-through data. In *SIGIR*, 2012.

[34] X. Yu, H. Ma, B. Hsu, and J. Han. On building entity recommender systems using user click log and freebase knowledge. In *WSDM*, 2014.

[35] J. Zhang, X. Kong, L. Jie, Y. Chang, and P. Yu. Ncr: A scalable network-based approach to co-ranking in question-and-answer sites. In *CIKM*, 2014.

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

[37] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, 2002.