Unsupervised Identification of Synonymous Query Intent Templates for Attribute Intents

Yanen Li[1], Bo-June (Paul) Hsu[2], ChengXiang Zhai[1]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
{yanenli2, czhai}@illinois.edu
[2]Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
paulhsu@microsoft.com

ABSTRACT

Among all web search queries there is an important subset of queries containing entity mentions. In these queries, it is observed that users are most interested in requesting some attribute of an entity, such as "Obama age" for the intent of age, which we refer to as the attribute intent. In this work we address the problem of identifying synonymous query intent templates for the attribute intent. For example, "how old is [Person]" and "[Person]'s age" are both synonymous templates for the age intent. Successful identification of the synonymous query intent templates not only can improve the performance of all existing query annotation approaches, but also could benefit applications such as instant answers and intent-based query suggestion. In this work we propose a clustering framework with multiple kernel functions to identify synonymous query intent templates for a set of canonical templates jointly. Furthermore, signals from multiple sources of information are integrated into a kernel function between templates, where the weights of these signals are tuned in an unsupervised manner. We have conducted extensive experiments across multiple domains in FreeBase, and results demonstrate the effectiveness of our clustering framework for finding synonymous query intent templates for attribute intents.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query Intent*

General Terms

Algorithms, Performance, Experimentation

Keywords

Attribute Intent, Synonymous Query Intent Templates, Clustering with Multiple Kernels

1. INTRODUCTION

Accurate understanding of the intent underlying a user query is a crucial problem in modern information retrieval. The understanding of user's intent not only can improve the accuracy of the search results, but also enables new types of applications that help the user make decision and finish tasks directly. For example, for a query "Tom Cruise age", an intelligent search engine would trigger the direct answer of "50 years" on top of the result page, which quickly fulfils the user's information need for finding this fact about Tom Cruise. Likewise, for a query "map of Chicago", it is more desirable to show a city map of Chicago directly. Another emerging application is the entity search, which returns the most relevant entities and attributes instead of relevant web pages. The examples mentioned above all require precise understanding of the intent of a user query.

In this work we focus on the subset of queries containing entity mentions since they are one of the most important subset of queries. One previous research reported that 71% of search queries contained named entities [1], another research found that 57% of queries have entities or entity categories [2]. We have observed from the query log of a major web search engine that among the entity queries, users are most interested in requesting an attribute of an entity. We refer to such intention as ***attribute intent***, which is the focus of this work. For example, "Obama age" for the intent of *age*, "The Museum of Modern Arts phone number" for the intent of *phone number* are attribute intents.

An ultimate goal of the attribute intent understanding is that assume we have a gigantic database of all entities and attributes in the world, map all possible search queries to the corresponding attributes. This goal shares similarities with other researches on query annotation [17, 22], in which they aim at annotating a query to a structured schema. Note that the query intents in these works are not necessarily attribute intents. However, these approaches either require a lot of labeled data, or rely on a highly developed and structured domain database. Most recently, unsupervised method does exist [11], yet it is limited in its ability to merge query patterns conveying same intent.

Not surprisingly, achieving the ultimate goal of attribute intent understanding automatically across many domains is very challenging. In some tail domain doing this mapping is not even possible because of the data sparsity. Thus in this work we aim at mapping the attribute intent at the template level, where individual queries are aggregated into query intent templates, partly reducing the data sparsity issue. For example, for the attribute "age" in Person domain, "[Person] age" is a query intent template for the "age" intent, while "Obama age", "Tom Cruise age" are individual queries conforming to this template. We refer to the query intent templates conveying same underlying intent as ***synonymous query***

intent templates. Table 1 shows two attribute intent templates as well as their synonymous templates. Specifically, in this work we seek to address the problem of identifying synonymous query intent templates for attribute intents. Successful identification of query intent templates to their corresponding attribute intents can provide benefits to several applications. First, it can potentially improve all exiting query intent annotation approaches by merging the synonymous templates. Second, in the application of instant answers, it enlarges the questions that a system can answer. Third, by knowing synonymous query intent templates, a search engine can recommend more queries based on rules between query templates for long-tailed queries [29].

Table 1: Synonymous Query Intent Templates

Domain	Attribute Intent Template	Synonymous Query Intent Templates
Person	[Person] age	[Person]'s age how old is [Person] age of [Person] what age is [Person]
Institution	[Institution] phone number	[Institution]'s phone phone number for [Institution] number to [Institution] [Institution] contact number

To the best of our knowledge, there are no existing methods that directly address the synonymous query templates identification problem. There might be several indirect solutions though. For example, the above mentioned query annotation methods could be used to solve the problem. However, due to the structure and semantic difference between queries and query templates, these methods might not be feasible. We could also apply approaches for question paraphrasing in the Question and Answer research community [30, 31]. However, [30] focuses on finding synonymous phrases on the question level, not on the template level. And [31] aims at identifying alternative expression of the answers, while in this paper we focus on alternative ways people query (ask) attribute intents. Thus these previous approaches can not be directly applied to the problem we are addressing.

In addition, we can treat it as a string synonym problem, which previous researches have addressed with several similarity features such as distributional similarity [18, 19], coclick similarity [10, 9] *etc*. However, from Table 1 we can see that sometimes the synonymous templates may be different from the canonical template (the attribute intent template itself) on small lexical variation and word orders. Yet sometimes they are very different on lexicon. Therefore, we can imagine that only relying on single similarity such as string similarity, it is very hard to discover templates that are synonymous to the canonical template in semantic level.

In fact, the synonymous query intent templates share similarities among several independent, sometime complementary signals. For example, other than the lexicon similarity, they would manifest high similarity on the distribution of entities that occur in the templates. In addition, people tend to click on same set of documents when they issue queries conforming to semantic equivalent templates. For instance, it's common that users who issue "Obama age" and "how old is Obama" will both land on the wikipedia page of Barack Obama. In this work we attempt to identify synonymous query intent templates by integrating multiple information sources. Among multiple information sources, some are more important in determining synonym relations than others. Moreover, the relative importance of these sources also depends on the domain: a feature that is crucial in the Person domain might be only marginal in the Location domain. Therefore automatic determination of the weights of different information sources is critical. Here we propose to automatically learn these weights via a novel clustering procedure with multiple kernel functions that cluster candidate query intent templates to the canonical templates and effectively determine the appropriate weights of the signals.

Other than the choice of information sources, there are different options for the mode of learning. We could identify the synonymous templates for a single canonical query intent template at a time, or learn the synonymous templates for all canonical templates jointly. Handling single template at a time might lead to the difficulty of differentiating synonymous templates of close related ones. For example, for template "[Person] height", "[Person] weight" could be identified as synonym mistakenly because they share a lot of coclicks. However if we learn synonymous templates from those two values jointly, this error could be corrected easily because of the awareness of each other. Joint learning of synonymous templates also provides the must-link and can-not-link constraints among canonical templates, which could effectively improve the results.

In this work we address the problem of finding synonymous query intent templates for a set of canonical query intent (attribute intent) templates jointly. We have proposed a clustering model with multiple kernel functions to take advantage of multiple heterogeneous information sources. Qualitative and quantitative results on attribute intent templates from a large knowledge demonstrate that our method can identify synonymous query intent templates accurately.

2. RELATED WORK

Query understanding has long been an important topic of information retrieval. In terms of query structural analysis, there are early and recent research attempts to do query segmentation [7, 14], part-of-speech tagging [5], shallow linguistic structure parsing [6] *etc*. In this work we are interested in entity related queries, where named entity recognition inside a query is an important task that has been addressed by previous works [13, 12]. These works are focused on entities, while the focus of this work is on the attributes, especially the attribute intent. There are recent attempts for entity attribute identification based on query logs [32, 33]; and the results from these works could potentially enlarge the attributes on which we are looking for synonymous query intent templates.

Finding synonymous query intent templates for attribute intent is an important task toward the ultimate goal of mapping all entity queries to the associated attribute intents. This goal shares similarity with the query annotation tasks, which try to annotate a query to some structured schema based on the query intent. There has been research attempts to find groups of queries sharing same intents [28, 21], however in these works queries are not explicitly mapped to structured schema. To do explicit schema discovery from queries, people have been using a grammar-based approach, in which domain specific rules are defined and queries are matched to these predefined rules. However these rules are domain dependent, which preclude them from applying to a large number of domains. Also, defining such rules manually takes a lot of effort. To reduce the cost of defining rules, supervised and semi-supervised statistical models are developed [17]. But a fair amount of labeled data is still required in these approaches. There are recent attempts to discover a query's schema information in an unsupervised way. For example, Sarkas *et al.* [22] propose an unsupervised method to annotate queries with a highly developed and structured domain database. Unfortunately, these resources may not be available to domains where data is sparse. To overcome this limitation, Cheung *et al.* propose using sequence clustering with a large open source knowledge graph for query schema annotation [11]. However, a major limitation of this work is that it is unable to merge query templates conveying same underlying intent. The main difference of this paper and these query annotation works is that we focus on mapping synonymous templates to canonical intent template, while these works focus on mapping queries to templates. Thus the previous works in this line is orthogonal to our work. The successful identification of synonymous templates can potentially improve the query annotation quality by merging the synonymous query templates. Besides, synonymous templates can be also used to improve query recommendation for long-tail queries [29].

The problem setup in this paper is similar to finding question paraphrases in Question and Answer [31, 30]. [31] aims at finding alternative expression of the answers, thus their features are majorly from the documents. However in this paper we focus on alternative templates people query (ask) attribute intents, not the answers. In addition, [30] focuses on finding synonymous phrases on the question level, not on the template level. So it is not obvious whether the method in [30] can be applied to our problem setup. What's more, [30] uses supervised approach to identify question paraphrase pairs, while we are more interested in unsupervised methods that can scale to large set of entities and attributes.

Researchers have also employed several similarity metrics to other tasks such as finding string synonyms from web data. Such similarities include distributional similarity [18, 19], coclick similarity [10, 9], point wise mutual information [25], and co-occurrence statistics [4]. Differing from these works, our work introduces novel similarity metrics like entity distribution similarity and query trend similarity that are tailored for finding synonymous query intent templates.

In our approach we combine the metrics with weights learn the weights automatically in an unsupervised manner. In this sense our work is also related to the previous works on semi-supervised metric learning [27, 23, 3]. We differ from these works in that our metric learning approach is embedded in a clustering framework.

3. PROBLEM DEFINITION

In this work, we seek to identify synonymous query intent templates from a set of entity attribute intents of a domain. The precise representation of the entity attribute intent could be difficult, here we start from using the attribute names as the representation of an entity attribute intent. And the definition of a domain can be flexible, we only assume that a domain contain a set of homogeneous entities with attribute values. Without loss of generality, we focus on entity attributes from a large open source knowledge graph: FreeBase where a large amount of the real world entities and attributes are covered. In fact, our technique does not depend on a specific data source. For example, it is not hard to adapt to other knowledge graphs such as Wikipedia.

Formally, given a set of M semantically distinct entity attribute intent templates $\mathbb{V} = \{v_1, v_2, ..., v_M\}$ for a domain, where each template $v \in \mathbb{V}$ is represented by its canonical form (here we assume the attribute name), such as "[Person] age" for intent "finding age of a person", and "[Movie] cast" for the intent "finding cast of a movie". From a set of N candidate synonymous templates $\mathbb{X} = \{x_1, \ldots, x_N\}$, we define an oracle mapping $\mathcal{F} : X \to \mathbb{V} \cup \{v_0\}$, which assigns each candidate template x to its unique canonical synonym query intent template v, or if x is not a synonym of any value in $\mathbb{V}$, to the special background template v_0. Note that we assume each candidate query template maps to at most one canonical template. Now, we can define the synonymous query intent templates identification problem as follows:

Definition 1: For each canonical attribute intent templates $v \in \mathbb{V}$, find the subset $\mathbb{X}_v = \{x \in \mathbb{X} | \mathcal{F}(x) = v\}$, representing the set of synonymous query intent templates for value v.

For example, for the attribute intent template "[Person] age", its synonymous templates include "how old is [Person]" and "age of [Person]", but not include templates like "[Person] birthday" or "how high is [Person]" though they are related templates.

4. A CLUSTERING FRAMEWORK

Here we propose to identify synonymous query intent templates by a clustering model that has multiple similarity functions. In this model, query templates $\mathbb{X} = \{x_1, x_2, ..., x_N\}$ are modeled as data points. Points are connected to each other by similarity functions f (also called kernels). Data points form clusters such that points in the same cluster are considered synonyms. The canonical query (attribute) intent templates $v_1, v_2, ..., v_M$ also belong to $\mathbb{X}$, but they have hard assignments to their respective clusters.

We tackle the problem with a clustering framework in order to take advantage of modeling a set of canonical templates jointly. We further consider transitivity and compactness in our model, which means we choose a cluster assignment by considering a committee of points in a cluster rather than a single medoid. In addition, previous researches measure similarity with single or only a few similarity features, while we want to support arbitrary features. Thus, manual tuning of parameters is not sufficient. Though there are existing works in metric learning with supervision, we are also interested in unsupervised techniques. Hence, we employ the regularized metric learning approach to guide our parameter optimization. In this section we first describe the center initialization and how to obtain candidate templates. After that, we define the similarity kernel functions according to different information sources. Then we introduce a basic clustering model for finding synonymous templates. Further, by addressing its limitations, we propose several extensions that lead to the refined model. To facilitate the readability of the paper, we summarize the major notations in Table 2.

Table 2: Major Notations in this Paper

Symbols	Descriptions
$\mathbb{V}$	canonical attribute intent templates
M	number of distinct canonical attribute intent templates
$\mathbb{X}$	candidate query intent templates
$\mathbb{X}_v$	synonyms of a target canonical template v
$f_t(.,.)$	similarity kernels
$d(.,.)$	distance between two attribute intent templates
W	kernel weights
R	partition matrix of the templates to clusters
Z	medoids of the clusters
γ	distance between a clusters to background
L	size of the pseudo-medoid
μ	proportion of random templates selected from background
S, U	should-link and should-not-link constraints

4.1 Cluster Center Initialization

Cluster center initialization includes two major steps: (1) allocate initial values to the centers; (2) choose the number of clusters. For step 1, we first assign each attribute names as the initial value of the cluster centers. This approach is generally effective; however, there is a limitation when the attribute names are long or written in a way that is uncommon in the query log where we look for candidates. For example, in the Person domain, the attribute names “date of the birth" and “country of nationality" are written in such forms. We employ an automatic reformulation method to handle the initial center allocations as follows: first, we check whether a canonical attribute name is above a threshold of query counts in the query log. If not, we use a state-of-the-art query log-based string synonym finder [9] to obtain a synonym that is most popular in the query log, and then replace it with this string synonym as the initial cluster center. For instance, in the above example, after proper reformulation, the initial value of these two centers are “[Person] date of birth" and “[Person] nationality". Finally, if we couldn't find any valid synonyms for an attribute name, we drop this cluster from the cluster list.

In step 2, we can set the number of clusters as the number of valid canonical attribute intent templates after reformulation in step 1. But we can do better by adding more clusters due to the reasons as follows: we observe that usually the current knowledge graph does not cover all attribute intents in a domain, and some of these uncovered intents will affect the clustering results. For example, for intent templates “[Person] age" and “[Person] height", we find that intents “[Person] wiki" and “[Person] biography" are ranked high in both clusters, which is not desirable. However, they are not covered by attribute names in Person domain in FreeBase, precluding us from assigning them as canonical templates. Hence, the final results can be improved by adding a few auxiliary clusters initialized by these popular intent templates. We identify these intent templates by two criteria: (1) they are most popular intents among all intents about the entities in a given domain; (2) they are not string synonyms of the valid canonical templates after processing in step 1. Furthermore, in addition to the auxiliary clusters, we add a background cluster to attract random templates. And the background cluster is initialized by a set of random candidate templates whose generation process is described in the following section.

After these steps, finally we obtain $K+1$ clusters (the background cluster being the 0^{th} cluster) in total, among which M' ($M' \leq M$) clusters correspond to the valid canonical attribute intent templates. There are also $K-M'$ auxiliary clusters, and 1 background cluster.

4.2 Candidate Query Intent Templates

Our clustering framework assumes we have a set of candidate query intent templates $\{x_1, x_2, ..., x_N\}$ as input. Here we describe how we obtain them efficiently. The search space of potential candidate templates is huge: any template in the query log that contains the target entities could potentially be a synonymous template. Therefore reducing the template search space is critical. Specifically, for a set of valid attribute intent templates $\mathbb{V} = \{v_1, v_2, ..., v_K\}$, we identify candidate templates as follows:

1. For each $v \in \mathbb{V}$, we first get the set of entities under intent templates v from FreeBase. And for each of these entities, we instantiate the template to a query by the entity mention. For example, “[Person] age" can be instantiated into “Obama age" or “Tom Cruise age". Then, for each of instantiated query, we obtain a set of most related queries by a query similarity function. For instance, “how old is Obama", “how old is Tom Cruise" are within the most related queries. By iteration all entities, we can get a set of related queries for v.

2. From the set of most related queries, we only keep queries that contain at least one entity in the entity set of v. Then, we generate candidate templates from such queries. In the above example, a candidate template “how old is [Person]" is generated accordingly.

3. In order to cover most of the candidates, other than a similarity function, we also employ other similarities to select template candidates. Particularly, for each similarity, we keep only top 50 candidates, and merge them and remove duplicates to form the set of candidate templates.

4.3 Freebase as a Knowledge Graph

Before diving into the similarity kernel definitions we briefly describe the knowledge graph we use to facilitate the query intent template initialization and kernel function computation. Freebase is a large knowledge graph that consists of over 20 million entities across over 10,000 concepts. There are multiple entities under a concept such as “People" or “Location", and each entity is consist of multiple attributes. We use the attribute name of the entities under a concept to nominate the query intent template, such as “[People] age" or “Phone number of [Institution]". In addition, we utilize the entities under a concept to compute the similarity kernel functions between two query intent templates.

4.4 Similarity Kernels

Defining proper similarity measures between data points is an essential part of a clustering framework. In our problem setup, we can measure the similarity between two query intent templates in different aspects. For example, two similar templates may share similar lexicon, with only minor difference in word order. Likewise, two templates are similar if people click on same set of documents after issuing queries conforming to these templates. Multiple heterogeneous information sources, in the forms of query log, anchor text, query document pairs, temporal and spatial query trends, can be used to calculate the similarity between query intent templates. In this section we describe 4 similarity functions, which are also called similarity kernels, in detail. Our clustering framework is not limited to these kernels; in fact our framework can accommodate arbitrary number of similarity kernels, whose weights are automatically tuned.

1. **Entity distribution similarity**. This similarity measures the entity frequencies under different query intent templates. Our hypothesis is that the distribution of entities for which user query about a particular intent should be similar among synonymous templates of that intent, while should be very different from templates conveying different intents. Fig. 1 illustrates an example of entity distribution similarity. We can see from Fig. 1 that query templates that follow the same underlying intent manifest similar query counts correlation across entities (color comparison in the same row). On the other hand, templates from different intents show very different entity distribution (row comparison). Formally, for a query intent template x_i, let its entity distribution be:

$$\Omega_i = \{\, \omega_1, \omega_2, ..., \omega_{n_e} \} \,, \tag{1}$$

where $\omega_t, 1 \leq t \leq n_e$ is the normalized query counts of a query intent template instantiated with an entity. And n_e is the number of entities in consideration. In the example shown in Fig. 1, ω_t could be the query counts of “Obama age", normalized by the total query counts of queries containing “Obama". And the similarity between templates x_i, x_j is the cosine similarity of Ω_i and Ω_j:

$$f_1(x_i, x_j) = \frac{\Omega_i \cdot \Omega_j}{||\Omega_i|| \cdot ||\Omega_j||} \tag{2}$$

2. **Coclick similarity.** This similarity measures how similar the set of documents users click on when they issue queries conforming to the query intent templates in consideration. We refer to such

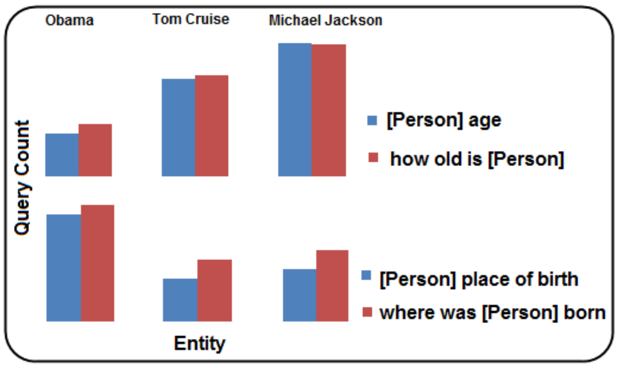

Figure 1: Entity distribution similarity

queries as proxy queries. A proxy query is generated by injecting an entity into the intent template. For example, "The Museum of Modern Art phone number" is a proxy query for template "[Institution] phone number". Given an intent template x_i, let the set of proxy queries of x_i be $\mathbb{Q}_i = \{q_1^i, q_2^i, ..., q_{n_i}^i\}$. For each query q_l^i, assume that the users have clicks (≥ 0) on a set of documents represented by vector: $\Delta^l = \{\delta_1^l, \delta_2^l, ..., \delta_{n_d}^l\}$, where n_d is the total number of documents. And let the accumulation of these clicks be:

$$\Delta = \sum_l \delta^l = \left\{ \sum_l \delta_1^l, \sum_l \delta_2^l, ..., \sum_l \delta_{n_d}^l \right\} \tag{3}$$

Under this assumption, for templates x_i and x_j, we define their coclick similarity as the cosine similarity of Δ_i and Δ_j:

$$f_2(x_i, x_j) = \frac{\Delta_i \cdot \Delta_j}{||\Delta_i|| \cdot ||\Delta_j||} \tag{4}$$

3. **Pseudo-document similarity.** This similarity is related to the coclick similarity, and it is a more robust measure on sparse data such as clicked documents. This similarity has been successfully applied to finding entity synonyms [9]. For a document D, the document title and body are not always easy to get; and they are usually long, thus might not be the best representation of D. Instead, the queries which have user clicks on D (referring queries) provide a succinct representation of D. We define the document augmented by all referring queries as the pseudo-document of D: P_D. We then define the one-way similarity of two proxy queries given the pseudo-document. For proxy queries q_a, q_b, let $\mathbb{D}_b$ be the set of documents users click after issuing q_b. The one-way pseudo-document similarity from q_a to q_b is:

$$\varphi(q_a \rightarrow q_b) = \frac{\text{Count}\left\{q_a \in P_{D_j} | D_j \in \mathbb{D}_b\right\}}{\text{Count}\left\{D_j | D_j \in \mathbb{D}_b\right\}}, \tag{5}$$

That is, the percentage of pseudo-documents that contain query q_a. Likewise, the symmetric, two-way pseudo-document similarity between q_a and q_b is:

$$\varphi'(q_a, q_b) = \frac{\varphi(q_a \rightarrow q_b) + \varphi(q_b \rightarrow q_a)}{2}. \tag{6}$$

Finally, for two query intent templates x_i, x_j, and given a set of entities E under the templates, we generate proxy queries q_e^i and q_e^j for x_i and x_j according to $e \in E$. We define the pseudo-document similarity between x_i and x_j as the average pseudo-document similarity of their corresponding proxy queries:

$$f_3(x_i, x_j) = \frac{1}{|E|} \sum_{e \in E} \varphi'(q_e^i, q_e^j) \tag{7}$$

4. **Query trend similarity.** Query templates with same intent should share similar query volume patterns over time. To illustrate this, we show 4 queries on GoogleTrend to display their query volumes from the year of 2005 to 2011 in Fig. 2. These 4 queries are of 2 distinct intents, one is "[Location] map", the other is "[Location] time zone". The temporal trends of these queries demonstrate that query patterns from same underlying intents are more correlated than from different intents. To measure it quantitatively, we collect a large query log with temporal query volumes. This query trend log contains about 80 millions unique queries. For each query, we record the monthly query volume from June 2008 to June 2011. Given a query q, let its monthly query volumes be $\Phi = \{\phi_1, \phi_2, ..., \phi_{n_t}\}$, where n_t is the total number of months in the records. Then for two proxy queries, we define their query trend similarity as the cosine similarity of Φ_i and Φ_j:

$$\varsigma(q_a, q_b) = \frac{\Phi_a \cdot \Phi_b}{||\Phi_a|| \cdot ||\Phi_b||} \tag{8}$$

And for two query intent templates x_i, x_j, and given a set of entities E under the templates, we generate proxy queries q_e^i and q_e^j for x_i and x_j according to e. Finally the query trend similarity between x_i and x_j is the average query trend similarity of the proxy queries:

$$f_4(x_i, x_j) = \frac{1}{|E|} \sum_{e \in E} \varsigma(q_e^i, q_e^j) \tag{9}$$

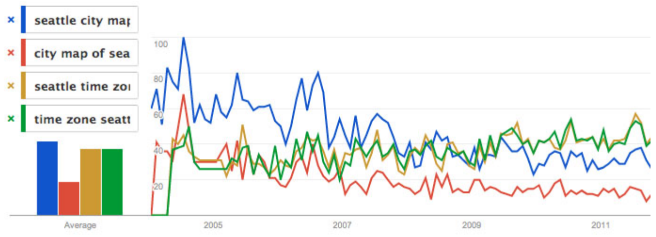

Figure 2: Query trend similarity

4.5 Integrating Multiple Kernels

The similarity kernels defined above have these properties: (1) symmetric; (2) $f_t(x_i, x_j) \in [0, 1]$, for $1 \leq t \leq T$. We seek to integrate multiple kernels into a new function that the weights of individual kernel can be automatically learned. Since our model resembles the K-medoids clustering [15], here we follow the nomenclature in the clustering literature and define the distance between query intent templates x_i and x_j as a convex combination of the similarity kernels:

$$\begin{aligned} d(x_i, x_j) &= \sum_{t=1}^{T} w_t \cdot d_t(x_i, x_j) \\ &= \sum_{t=1}^{T} w_t \cdot (1 - f_t(x_i, x_j)) \end{aligned} \tag{10}$$

where $f_t(x_i, x_j) \in [0, 1]$ is the similarity kernel of x_i and x_j calculated based on evidence from information source $t \in \{1, ..., T\}$. Likewise $d_t(x_i, x_j) = 1 - f_t(x_i, x_j) \in [0, 1]$ is the distance between x_i and x_j. And w_t are the weights needed to be learned, following non-negative constraints:

$$w_t \geq 0, 1 \leq t \leq T \tag{11}$$

In the following, we describe our clustering model that identifies synonymous query intent templates and learns the appropriate kernel weights iteratively.

4.6 Initial Model

We first define an objective function for our model and refine it by overcoming its limitations. Overall, we aim at minimizing the following objective function:

$$\begin{aligned} & g_0(R, Z, W) \\ =& \sum_{k=1}^{K}\sum_{i=1}^{N} r_{i,k} \cdot d(x_i, z_k) + \sum_{i=1}^{N} r_{i,0} \cdot d(x_i, z_0) \\ =& \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{t=1}^{T} r_{i,k} \cdot w_t \cdot d_t(x_i, z_k) + \sum_{i=1}^{N} r_{i,0} \cdot \gamma \end{aligned} \quad (12)$$

subject to:

$$\begin{cases} \sum_{k=0}^{K} r_{i,k} = 1, 1 \le i \le N \\ r_{i,k} \in \{0, 1\}, 1 \le i \le N, 0 \le k \le K \\ w_t \ge 0, 1 \le t \le T \end{cases} \quad (13)$$

The above objective function minimizes the sum of with-cluster dispersions. In this formulation, the first term is the overall within-cluster distances of the normal clusters, and the second term is the within-cluster distances in the background cluster. Such formulation is to make the resulting clusters compact. Note that in our model there is no need to represent data points with explicit feature vectors or coordinates, instead, we only require that $d(x_i, x_j) \ge 0$. The notations of variables are listed below:

- $d(x_i, z_k)$ is the overall distance function between x_i and z_k, as defined in *Eq.* (10);
- $Z = \{z_0, z_2, ..., z_K\}$ are the medoids of the clusters. The first K medoids are fixed to the valid canonical intent templates $\{v_1, v_2, ..., v_K\}$ for which we look for synonymous templates;
- R is an $N \times (K + 1)$ partition matrix, where N is the total number of points and $K + 1$ is the number of clusters; $r_{i,k} \in \{0, 1\}$ is a binary variable; $r_{i,k} = 1$ indicates object x_i is in k^{th} cluster;
- $W = \{w_1, w_2, ..., w_T\}'$ are the weights of different distance kernels;
- γ is a constant measuring the distance of $x \in X$ to the background cluster.

The objective function of the initial model is similar to the formulation of K-medoids[15]. The advantage of employing the K-medoids framework rather than K-means is that we only need to define distance functions (kernels) between data points, while explicit feature vectors are not needed. This is desirable because feature vectors are sometime hard to represent explicitly. In addition, there are important differences between our initial model and the K-medoids model: firstly, the first K medoids in our model are fixed to the canonical templates, assuming they are best representatives of these clusters. This also implies that there is no need to update the medoids. Secondly, in our model the distance between points is a weighted kernel function, which is very different from the standard K-medoids model. Such weights measure the relative contribution of the kernels, and they are estimated in an unsupervised manner. Thirdly, in our model we add a background cluster in order to attract the random points. Here we assume that the distance of any point to the background cluster is a constant.

Although this initial model can partition the data points into clusters efficiently, it suffers from the following limitations: (1) Using a single fixed representative for a cluster may be problematic. First, the canonical template is not always the most representative one. It may have idiosyncrasies that are not shared by other members of the cluster. Second, because the similarity features are noisy, if we only compare a candidate against the canonical template, a noisy feature may bias it towards an incorrect cluster. (2) Manually setting the constant γ is very difficult. Nominating a good γ at the beginning is hard, and further, since the distance between data points depends on the weights, it makes it even harder to choose the appropriate γ inside the algorithm. Therefore an automatic estimation of this constant is necessary. (3) Not enough guidance for learning the kernel weights. The initial model doesn't make use of the constraints in the forms of must-link and can-not link, to learn the optimal weights.

4.7 Refined Model

We propose to address Limitation 1 by introducing pseudo-medoids instead of fixed centers; and we address Limitation 2 by estimating γ with random points. In addition, we tackle Limitation 3 by adding constraints that regularize the value of kernel weights. Now we look for cluster assignments and kernel weights that minimize the new objective function:

$$\begin{aligned} & g_1(R, Z', W) \\ =& \sum_{k=1}^{K}\sum_{i=1}^{N} r_{i,k} \cdot d(x_i, z'_k) + \sum_{i=1}^{N} r_{i,0} \cdot d(x_i, z_0) + ||W|| \\ &+ \beta_1 \sum_{S_k \in S} \sum_{(x_i,x_j)\in S_k} d(x_i, x_j) - \beta_2 \sum_{U_k \in U} \sum_{(x_i,x_j)\in U_k} d(x_i, x_j) \end{aligned} \quad (14)$$

subject to: *Eq.* (13), where z'_k is the pseudo-medoid, A is the subset of random points in the background cluster. $\beta_1 \ge 0, \beta_2 \ge 0$ are the weights for regularization terms. And

$$d(x_i, z'_k) = \sum_{x_j \in z'_k} \frac{1}{|z'_k|} d(x_i, x_j), \quad (15)$$

$$d(x_i, z_0) = \sum_{x_j \in A} \frac{1}{|A|} d(x_i, x_j), \quad (16)$$

$$||W|| = \sqrt{\sum_{t=1}^{T} w_t^2} \quad (17)$$

This new formulation addresses all limitations of the initial model. First, limitation 1 can be addressed by having a flexible representative or a small set of representatives for each cluster. However it's not desirable to have flexible medoids since in our problem setup the canonical values are good representatives and it is more robust to include them into the medoids. Thus we propose to use a small subset of points, including the canonical value, to form a new pseudo-medoid z'_k. This subset is viewed as a committee that determines the distance from a point to the cluster. By carefully choosing pseudo-medoid members, it will reduce the risk of having unpopular canonical values and noisy features, making the clusters more compact. In fact a similar idea of clustering with committees of points has been successfully applied to the document clustering problem [20]. As the optimal solution for K-medoids cluster is NP-hard, we can only find the local optimum of medoids by algorithms such as PAM [16]. However, this algorithm takes $O(m^2)$ time to update a medoid where m is the number of points in a cluster, which is inefficient. Also it doesn't take the advantage of the canonical templates. In our new formulation, we form the new pseudo-medoid by including the top $L-1$ most similar templates to the canonical template as well as the canonical template itself. The advantage of this nearest neighbors approach is that it forms a compact pseudo-medoid around the canonical value efficiently, which takes only $O(m)$ time.

Second, by assuming random points follow similar properties as the background, we address the limitation 2 by to randomly selecting μ proportion of points to form the set A from the background cluster. Thus γ can be estimated by taking the average of the distance to this random subset. Results show that the final synonyms are stable with respect to different setting of μ.

Third, we propose to guide the search of optimal kernel weights in a regularization framework. The first regularization term $||W||$ is to prevent the weights from becoming too large. The second regularization term acts as a must-link constraint in which each set S_k contains templates that should be close to each other. In this work we include templates of each pseudo-medoid z'_k to S_k, which tries to make the pseudo-medoid more compact. On the contrary, the third regularization term acts as a can-not-link constraint in which U_k contains templates that should be far away from each other. Actually for each cluster, the canonical intent template should be away from the other clusters. In that way, we include K pairs of points in each U_k, resulting in $\frac{K\cdot(K-1)}{2}$ such pairs in total.

4.8 Solving the Model

In the refined model there are three set of unknown variables: R, Z' and W, which are dependent on each other. There is no exact solution to solve all of them at the same time. Instead we solve this optimization problem by iteratively solving the following minimization problem:

1. Fix $Z' = \hat{Z}'$ and $W = \hat{W}$; find the best R that minimizes $g_1(R, \hat{Z}', \hat{W})$

2. Fix $W = \hat{W}$ and $R = \hat{R}$; find the best medoids Z' that minimizes $g_1(\hat{R}, Z', \hat{W})$

3. Fix $Z' = \hat{Z}'$ and $R = \hat{R}$; solve the best parameters W that minimizes $g_1(\hat{R}, \hat{Z}', W)$

Sub-problem 1 can be solved by:

$$\begin{cases} r_{i,k} = 1 \quad \text{if} \quad d(x_i, z'_k) \le d(x_i, z'_l), 0 \le k, l \le K \\ r_{i,k} = 0 \quad \text{otherwise} \end{cases} \tag{18}$$

For **sub-problem 2**, we update the pseudo-medoids of first K clusters by including up to the top $L-1$ most similar values to the canonical value as well as the canonical template itself:

$$z'_k \leftarrow v_k \cup \{L-1 \text{ nearest neighbors of } v_k \text{ in cluster } k\} \tag{19}$$

For the background cluster, there is no need to calculate the updated medoid. For **sub-problem 3**, we follow the basic ideas for solving the regularized metric learning problem [23]. Note that after fixing R and Z, solving *Eq.* (14) becomes a Semi-Definite Programming (SDP) problem. To see this, we rewrite *Eq.* (14) as:

$$\begin{aligned} &g_1(R, Z', W) \qquad (20)\\ &= \sum_{k=1}^{K}\sum_{i=1}^{N} r_{i,k}\cdot d(x_i, z'_k) + \sum_{i=1}^{N} r_{i,0}\cdot d(x_i, z_0) + y \\ &+ \beta_1 \sum_{S_k\in S}\sum_{(x_i,x_j)\in S_k} d(x_i,x_j) - \beta_2 \sum_{U_k\in U}\sum_{(x_i,x_j)\in U_k} d(x_i,x_j) \end{aligned}$$

subject to:

$$\begin{cases} \sqrt{\sum_{t=1}^{T} w_t^2} \le y \\ \\ w_t \ge 0, 1 \le t \le T \end{cases} \tag{21}$$

Now the objective function is linear in both y and W. It has two convex constraints: the first is a second order cone constraint, while the second is a positive semi-definite constraint. There exist efficient solutions that guarantee to solve this problem in a polynomial time. In this work we use the method implemented in SeDuMi [24] to solve W efficiently.

4.9 Algorithm Procedure

The optimal allocation of points to clusters and the best kernel weights can be found by iteratively solving the above sub-problems. These iterative updates are summarized in **Algorithm 1**. Algorithm 1 is guaranteed to converge to a local minimum after several iterations. The time complexity of **Algorithm 1** is $O(P\cdot(N\cdot L\cdot(K+1)\cdot T + T^3))$, where P is the number of iterations, N is the number of candidate templates, L is the number of nearest neighbors, $K+1$ is the number of clusters, and T is the number of similarity kernels. The SDP solver for W takes T^3 time complexity.

Algorithm 1: Clustering with Multiple Kernels

```
input : A set of canonical attribute values $\{v_1, v_2, ...v_K\}$
        and a set of candidate strings $\{x_1, x_2, ..., x_N\}$
output: Optimal membership matrix $R$, pseudo-medoids $Z'$
        and distance kernel weights $W$
1 Init: init $Z'^0 = \{v_1, v_2, ..., v_k\}$ and choose uniform
    $W^0 = \{\frac{1}{T}, ..., \frac{1}{T}\}$.
2 for $q \leftarrow 0$ to $q_{max}$ do
3     Let $\hat{Z}' = Z'^q$ and $\hat{W} = W^q$, find the best $R^{q+1}$ in
      sub-problem $g_1(R, \hat{Z}', \hat{W})$ according to Eq. (18). If
      $R^{q+1} = R^q$, output $R^q$, $\hat{Z}'^q$, $\hat{W}^q$ and stop.
4     Let $\hat{R} = R^{q+1}$ and $\hat{W} = W^q$, update $Z'^{q+1}$ in
      sub-problem $g_1(\hat{R}, Z', \hat{W})$ according to Eq. (19).
5     Let $\hat{R} = R^{q+1}$ and $\hat{Z}' = Z'^{q+1}$, find the best $W^{q+1}$ in
      sub-problem $g_1(\hat{R}, \hat{Z}', W)$ according to Eq. (20) and Eq.
      (21)
```

5. EXPERIMENTS AND RESULTS

To test the effectiveness of our proposed model, in this section we have carried out a set of experiments on datasets across multiple domains from a large knowledge graph. We first present the qualitative results obtained by all methods in comparison. Then we make direct comparison of our model to the baselines quantitatively. After that we investigate the model performance at top $\bar{K}$

results. Finally we analyze the model sensitivity subject to different parameter settings.

5.1 Datasets

For evaluation purpose, we have collected a large set of canonical templates from Freebase. These templates are formed by combining the entity type and attribute names, such as "[Person] place of birth", "[Educational Institution] phone number". They are collected from three major domains: People, Location, and Organization, which have many sub-relations. We have selected 113 canonical templates from People domain, 102 from Location domain, and 93 from Organization domain respectively. Because it's very difficult to come up with all true synonymous templates of the selected canonical templates, we employ the TREC style pooling strategy to obtain the initial pool of candidate ground-truths. That is, for each canonical template, we use all competing methods to produce up to 50 best synonyms. And then these results are pooled, producing another list of top 50 templates by a simple ensemble method. Finally domain experts are asked to label these candidate templates into true synonyms or not (1/0 labeling), resulting in 15450 labels in total. All of the labeled data as well as our experiment results will be freely available to the public.

We also describe the data sources from which the similarity functions are computed. Firstly, all of the similarity kernels are computed on a large query log and a query-click log from a major commercial search engine. There are more than 100 millions unique queries in the query log and about 600 millions query-clicked document pairs in the query-click log. Part of the queries has monthly query volume records from June 2008 to June 2011, which are used to compute the query trend similarity. All these query and click logs are preprocessed and indexed in a compression trie data structure so that the similarities can be computed efficiently.

5.2 Baselines

To the best of our knowledge, there is no existing work directly addressing the problem of finding synonymous query intent templates for attribute intents. Thus we choose a set of baselines that are based on individual features. We also add another baseline by string synonym induction.

1. ***Individual feature***. Individual feature can identify synonymous query intent templates effectively. Here we include baselines using individual feature we defined in section 4.4. Synonymous templates are identified by single input canonical template at a time.

2. ***Clustering with Fixed Weights***. In order to reveal the effectiveness of the kernel weights learning in our model, we add a baseline that uses the same clustering model, with fixed and equal kernel weights (0.25).

3. ***String Synonym Induction***. Synonymous query templates can be also identified by inducing the string synonyms of proxy queries. Specifically, for a canonical query intent template, we first obtain a set of proxy queries by instantiating the associated entities. Next, string synonyms of these proxy queries are found by a start-of-the-art synonym finding approach [9]. After that, the resulting string synonyms are induced back to query templates. Finally synonymous templates are found by keeping the most popular induced templates.

5.3 Evaluation Metrics

We evaluate our system using metrics of Mean Average Precision (MAP) and Precision @ $\bar{K}$, which are based on the precision and recall measures. Specifically, for an canonical query intent template, $v \in V$, $O(v) = \{o_1, o_2, ..., o_{K_v}\}$ is the set of synonym outputs in ranked order (decreasing). Let $S(v)$ denote the set of true synonymous templates annotated by the human experts for v. Precision for a particular input template v is computed as:

$$\textit{precision} = \frac{1}{|O(v)|} \sum_{o \in O(v)} I_p(o, v)$$

where $I_p(o, v) = 1$ if $o \in S(v)$, and 0 otherwise. When evaluating methods with ranked synonyms, MAP is a good measure to distinguish the approaches across different recall levels. MAP is the mean of the average precision for multiple templates. Suppose for canonical template v_j, the set of true synonyms of v_j is $S(v_j) = \{s_1, s_2, ..., s_{m_j}\}$, and R_{jk} is the set of ranked synonymous templates obtained by taking the top results until we reach true synonym s_k, then

$$\text{MAP}(V) = \frac{1}{|V|} \sum_{j=1}^{|V|} \frac{1}{m_j} \sum_{k=1}^{m_j} \textit{precision}(R_{jk})$$

Note that when a true synonymous template is not found in the ranked list, the corresponding precision in the above equation is set to 0. Besides MAP, we also use the Precision @ $\bar{K}$ measure to assess the quality of top ranked results down to position $\bar{K}$, which has practical significance.

5.4 Parameter Settings

The performance of our models will vary with respect to different parameter setting. Table 3 lists the optimal parameter setting we use to report results.

Table 3: Parameter Setting

Parameter	Setting	Meaning
$L-1$	2	number of nearest neighbors
μ	0.05	proportion of random points from the background
β_1, β_2	1	coefficients of the regularization terms

5.5 Qualitative Analysis of Results

In order to demonstrate what the results really look like and reveal the strength and weakness of the methods in comparison, we conduct a qualitative analysis in a concrete example. Table 5 shows the results of an input template "[Person] place of birth". Due to space limitation, for individual features we only present the results of Pseudo-document Similarity and Query Trend Similarity, which represent a strong feature and a weak feature. Based on the example, we have the following observations: First, among all individual features, the results computed by Pseudo-document Similarity is good. However there are noisy templates such as "autobiography [Person]" and "wiki [Person]" which affect the results. On the other hand, the results obtained by Query Trend Similarity is poor; there are several related, but not true synonymous, templates at the top results. Second, although String Synonym Induction demonstrates effectiveness in producing good results, it is unable to identify other true synonyms like "where was [Person] born", which might due to the limitation of its focus on finding string synonyms. Thirdly, results produced by Clustering with Fixed Weights are very good though it is still inferior to our clustering model that produces most clean results.

5.6 Overall Effectiveness

In this experiment we investigate the overall effectiveness of our clustering model. We present the results measured in Mean Average Precision (MAP) from all three domains in Table 4. The results indicate that (1) among all individual features, the Pseudo-document Similarity is the most effective feature, achieving MAP of 0.588; while the weakest one is Query Trend Similarity, only getting MAP of 0.332. (2) finding synonymous templates by String Synonym Induction is effective, achieving intermediate performance. (3) learning synonymous query templates jointly clearly demonstrates advantages: it achieves a competitive MAP of 0.608 even by simply fixing the weights to be all equal. And by learning the weight with regularization, our clustering model achieves 0.670 in MAP, which is significantly better than all other methods. We have conducted the statistical significance test (T-Test) and it is shown that there is statistically significant difference between our model and two baselines (Clustering with Fixed Weights and String Synonym Induction) at confidence level $p = 0.01$. The quantitative results in Table 4 are also consistent with our findings in the qualitative analysis.

Table 4: Synonymous Templates in All Domains

Domain	Method	MAP
	Entity Dist. Similarity	0.469
	Coclick Similarity	0.490
	Pseudo-doc. Similarity	0.588
All	Query Trend Similarity	0.332
	Clustering with Fixed Weights	0.608
	String Syn. Induction	0.543
	Our Model	**0.670**

5.7 Analysis of Top Ranked Results

Previous experiment measures the average precision across all levels of recall. In this experiment we zoom in the results at top ranked positions since high precision at top positions is highly demanded in several applications. We summarize the Precision @ $\bar{K}$ results at Fig. 3. Results in Fig. 3 clearly show the superior performance of our model, especially at top positions. For example, our model achieves precision of 0.899 at position 1 and 0.778 at position 2, which is much better than the baselines. These results are consistent with the MAP results reported in previous experiment, suggesting that methods that perform well in top ranked positions are generally superior to the methods that do poorly in top positions.

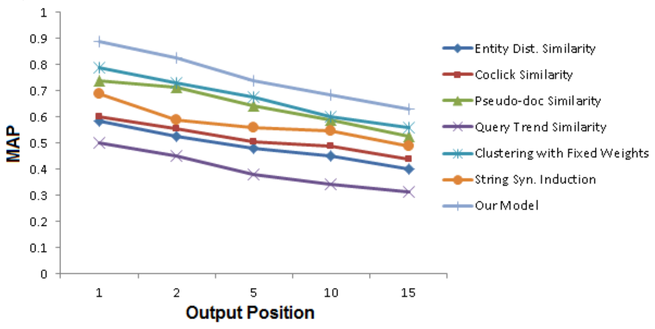

Figure 3: Precision @ $\bar{K}$ Results

5.8 Result Breakdown by Domains

Further, we break down the results by domains to investigate how well our model does in different domains. The breakdown results are shown in Fig. 4. We have two observations on the breakdown results: First, baselines powered by individual features achieve consistent results across domains, suggesting that the features we used are stable across different categories. Second, our model achieves best performance in all three domains, indicating that our clustering framework with multiple kernels may be applied a large number of domains without relying on domain specific labels that are expensive to collect in large scale.

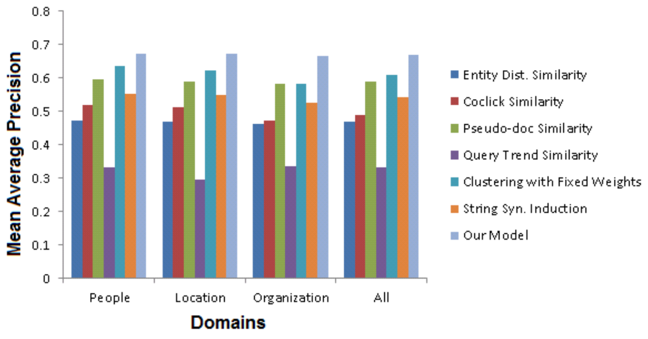

Figure 4: Result Breakdown by Domains

5.9 Contribution of the Kernels

Our proposed model is able to learn the weights of similarity kernels. In this experiment we look into the learnt weights to see whether they reflect the relative importance of the similarity kernels. For this purpose, we have conducted the ablation test, in which we **remove** one similarity kernel at a time and run the model. We also report the weights learnt without removing any kernels. Results on a subset of data in each domain are shown in Table 6. These results indicate that Pseudo-document Similarity seems to play relatively higher importance than other kernels. For example, in all three domains, it carries the highest weights; and the F1 measures drop to the lowest when removing this kernel (the lowest F1 is marked in bold). On the contrary, the Query Trend Similarity seems to play a marginal role in the model, reflected by the low weights in all domains. And removing Query Trend Similarity has a negligible effect on the MAP.

Table 6: Relative Importance of Similarity Kernels

Domain	W/ MAP	Entity Dist. Sim.	Coclick Sim.	Pseudo -doc. Sim.	Qry Trend Sim.
People	W	0.23	0.26	0.35	0.16
	MAP	0.640	0.633	**0.625**	0.659
Location	W	0.20	0.28	0.37	0.15
	MAP	0.621	0.637	**0.620**	0.671
Organization	W	0.25	0.30	0.32	0.13
	MAP	0.625	0.622	**0.606**	0.668

5.10 Model Sensitivity Analysis

Furthermore, we discuss the sensitivity of our model's performance according to the change of parameters. We focus on two parameters: one is $L - 1$, the number of nearest neighbors; the other is μ, the proportion of random points drawn from the background. According to Fig. 5, the Mean Average Precision of our model does change with respect to different $L - 1$. The MAP increases when $L - 1$ increases with a small step, while it decreases when the $L - 1$ becomes relatively big. This suggests that while

Table 5: Qualitative Results of the Synonymous Query Intent Templates

Canonical Template	Our Model	Pseudo-doc Similarity	Query Trend Similarity	Clustering with Fixed Weights	String Syn. Induction
[Person] place of birth	[Person] birth place	[Person] birth place	[Person] birth date	[Person] birth place	[Person] birth place
	birth place of [Person]	where was [Person] born	birthplace of [Person]	birth place of [Person]	[Person] birthplace
	birthplace of [Person]	[Person] birth	[Person] birth	[Person] birth	birth place of [Person]
	where was [Person] born	birth place of [Person]	[Person] born	where was [Person] born	birthplace of [Person]
	[Person]'s birth place	autobiography [Person]	[Person] bibliography	fun facts about [Person]	birthplace [Person]
	where did [Person] live	the birth of [Person]	autobiography of [Person]	birthplace of [Person]	[Person]'s birth place
	where was [Person] really born	birthplace of [Person]	wiki [Person]	[Person]'s birth place	birthplace [Person]
	the birth of [Person]	wiki [Person]	birth place of [Person]	where was [Person] really born	[Person] born place
	[Person] birth	[Person] death date	interesting facts about [Person]	the birth of [Person]	[Person] birth
	[Person] born	[Person] date of birth	birth place of [Person]	info on [Person]	[Person] s birth date

the pseudo-medoid is effective, if it becomes too large, the inclusion of noisy data will hurt the result. Thus, a small size of $L-1$, like 1 or 2, is a good choice. On the other hand, Fig. 6 indicates

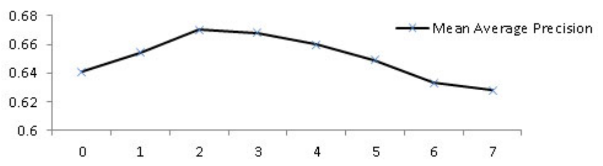

Figure 5: Results according to different $L-1$.

that results are most stable with different values of μ. It suggests that using a small μ is reliable, which makes our clustering model running faster.

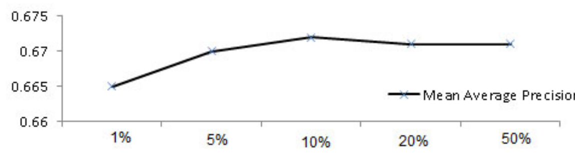

Figure 6: Results according to different μ.

6. CONCLUSION AND FUTURE WORKS

In this work we address the problem of finding synonymous query intent templates for a set of canonical query intent templates simultaneously. To solve this problem, we propose a clustering model that enables the integration of multiple heterogeneous information sources. The weights of these signals are tuned by a regularized metric learning strategy. We have conducted extensive experiments on a large set of attribute intent templates from FreeBase. Qualitative and quantitative results both demonstrate that the performance of our model is superior to the baselines. There are some interesting problems remained as future works. For example, given a small amount of labeled data, how to integrate the labeled data into our model to further improve the accuracy is an interesting problem to investigate. Another interesting problem is how to take advantage of the synonymous query templates to improve the query annotation quality.

7. REFERENCES

[1] J. Guo, G. Xu *et. al.* Named Entity Recognition in Query. In *SIGIR '09*.

[2] T. Lin, P. Pantel *et. al.* Active Objects: Actions for Entity-Centric Search. In *WWW '12*.

[3] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *J. Mach. Learn. Res.*, 2005.

[4] M. Baroni and S. Bisi. Using cooccurrence statistics and the web to discover synonyms in technical language. In *In Proceedings of LREC 2004*, 2004.

[5] C. Barr, R. Jones, and M. Regelson. The linguistic structure of english web-search queries. In *EMNLP'08*.

[6] M. Bendersky, W. B. Croft, and D. A. Smith. Structural annotation of search queries using pseudo-relevance feedback. In *CIKM '10*.

[7] S. Bergsma and Q. I. Wang. Learning noun phrase query segmentation. In *In EMNLP-CoNLL '07*.

[8] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, Sept. 2002.

[9] K. Chakrabarti, S. Chaudhuri, T. Cheng, and D. Xin. A framework for robust discovery of entity synonyms. In *In KDD '12*.

[10] T. Cheng, H. W. Lauw, and S. Paparizos. Fuzzy matching of web queries to structured data. In *ICDE'10*.

[11] J. C. K. Cheung and X. Li. Sequence clustering and labeling for unsupervised query intent discovery. In *WSDM '12*.

[12] J. Du, Z. Zhang, J. Yan, Y. Cui, and Z. Chen. Using search session context for named entity recognition in query. In *SIGIR '10*

[13] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *SIGIR '09*

[14] M. Hagen, M. Potthast, B. Stein, and C. Bräutigam. Query segmentation revisited. In *WWW '11*.

[15] L. Kaufman and P. Rousseeuw. *Clustering by Means of Medoids*. Delft University of Technology. Fac., Univ., 1987.

[16] L. Kaufman and P. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Applied probability and statistics. Wiley, 1990.

[17] X. Li, Y.-Y. Wang, and A. Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *SIGIR'09*.

[18] D. Lin. Automatic retrieval and clustering of similar words. In *ACL '98*.

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

[20] P. Pantel and D. Lin. Document clustering with committees. In *SIGIR'02*.

[21] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *WWW'10*.

[22] N. Sarkas, S. Paparizos, and P. Tsaparas. Structured annotations of web queries. In *SIGMOD '10*.

[23] L. Si, R. Jin, S. C. H. Hoi, and M. R. Lyu. Collaborative image retrieval via regularized metric learning, 2006.

[24] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones, 1998.

[25] P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *EMCL '01*.

[26] M. J. Welch and J. Cho. Automatically identifying localizable queries. In *SIGIR '08*.

[27] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *NIPS 2002*.

[28] X. Yin and S. Shah. Building taxonomy of web search intents for name entity queries. In *WWW '10*.

[29] I. Szpektor, A. Gionis and Y. Maarek. Improving Recommendation for Long-tail Queries via Templates. In *WWW '11*.

[30] S. Zhao, M. Zhou and T. Liu. Learning Question Paraphrases for QA from Encarta Logs. In *IJCAI '07*.

[31] D. Ravichandran and E. Hovy Learning Surface Text Patterns for a Question Answering System. In *ACL '02*.

[32] M. Pasca, and B. V. Durme What You Seek is What You Get: Extraction of Class Attributes from Query Logs. In *IJCAI '07*.

[33] M. Pasca, and B. V. Durme Weakly-Supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. In *ACL '08*.