Automatic Discovery of Attribute Synonyms Using Query Logs and Table Corpora

Yeye He[1], Kaushik Chakrabarti[1], Tao Cheng[2] *, Tomasz Tylenda[3] †

[1]Microsoft Research, Redmond, USA
[2]Pinterest, Inc., San Francisco, USA
[3]Max-Planck Institute for Informatics, Saarbrucken, Germany

[1]{yeyehe, kaushik}@microsoft.com
[2]taocheng@pinterest.com
[3]ttylenda@mpi-inf.mpg.de

ABSTRACT

Attribute synonyms are important ingredients for keyword-based search systems. For instance, web search engines, recognize queries that seek the value of an entity on a specific attribute (referred to as e+a queries) and provide direct answers for them using a combination of knowledge bases, web tables and documents. However, users often refer to an attribute in their e+a query differently from how it is referred in the web table or text passage. In such cases, search engines may fail to return relevant answers. To address that problem, we propose to automatically discover all the alternate ways of referring to the attributes of a given class of entities (referred to as attribute synonyms) in order to improve search quality. The state-of-the-art approach that relies on attribute name co-occurrence in web tables suffers from low precision.

Our main insight is to combine positive evidence of attribute synonymity from query click logs, with negative evidence from web table attribute name co-occurrences. We formalize the problem as an optimization problem on a graph, with the attribute names being the vertices and the positive and negative evidences from query logs and web table schemas as weighted edges. We develop a linear programming based algorithm to solve the problem that has bi-criteria approximation guarantees. Our experiments on real-life datasets show that our approach has significantly higher precision and recall compared with the state-of-the-art.

1. INTRODUCTION

Keyword-based search systems often need to understand synonyms that people use to refer to both entities and attributes in order to return the most relevant results. While discovering entity synonyms has been a common topic of research [10, 24], the problem of attribute synonym has so far received little attention. The lack of attribute synonyms often limits the efficacy of keyword search systems.

Figure 1: Example search results for e+a queries

In the following, we will use web search engine as a concrete example to illustrate the importance of attribute synonyms in keyword search, although its importance clearly extends beyond web search engine (e.g., for database keyword search and other types of schema search [9]).

Web search engines now answer certain types of queries directly using structured data and other sources [17, 23, 30]. For instance, for the query {barack obama date of birth}, both Bing and Google show the answer "August 4, 1961" prominently above its regular results. The above query is an example of an important class of web queries where the user specifies the name of entity and the name of an attribute and seeks the value of that entity on that attribute [30]. We refer to them as entity-attribute queries or "e+a" queries in short.

Web search engines answer many e+a queries using a curated knowledge base containing entities and their values on various attributes. It has been recognized that these knowledge bases have low coverage of tail entities and attributes [17, 28]. For example, Google does not answer the query {number of english speakers in china} using their knowledge base, because it likely does not have that attribute for the

*Work done during employment at Microsoft Research
†Work done during employment at Microsoft Research

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
http://dx.doi.org/10.1145/2872427.2874816.

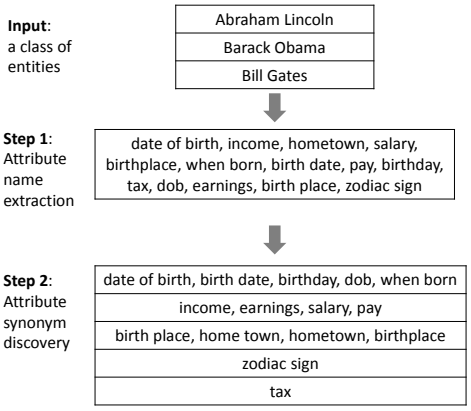

Figure 2: Two step framework for attribute synonyms

entity class Country. Such queries are answered using web tables and text passages as they cover the long tail of entities and attributes [17, 30]. For example, Google answers the above query using a web table as shown in Figure 1(a).

Users often refer to an attribute in their e+a query differently from how it is referred in the web table or text passage. In such cases, search engines will fail to return relevant answers. For example, if the user asks for {english literacy rate in china} (where 'english literacy rate' may be an alternate way of referring to '% english speakers'), Google fails to return the above answer as shown in Figure 1(b).

To address this problem, we propose to *automatically discover all the alternate ways of referring to the attributes of a given class of entities*. We refer to these alternative attribute names as *attribute synonyms*.[1] Like previous works on attribute extraction, we perform synonym discovery *for a given class of entities* [17, 18, 20, 21]. For example, our approach can discover that 'english literacy rate' is a synonym of '% english speakers' for the entity class Country. With this knowledge, the search engine will be able to return the relevant web table for the above query.

We adopt a two-step framework: *attribute name extraction* and *attribute synonym discovery*. Given an entity class and some entity instances of that class, the first step extracts all attribute names for that class from sources like query click log and web table corpus. The second step then identifies synonyms among all such attribute names. Figure 2 shows an example input and output of the two steps for the class Person.

Attribute name extraction has been studied extensively in prior works [17, 18, 20, 21]; we use a variant of those techniques in this paper. On the other hand, attribute synonym discovery has received little attention in the literature, and is the focus of this paper. We briefly describe two baseline techniques for synonym discovery and their limitations; a more detailed discussion can be found in Section 6.

• *Thesaurus*: We consider a pair of attribute names as synonyms if they occur as synonyms in a manually compiled thesaurus like Wiktionary [5] or Merriam-Webster [2]. The limitation of this approach is that synonymity of attribute names often depend on the entity class (e.g., "megapixels" and "mp" are synonyms of "resolution" only for the class Camera); the thesaurus is context-independent and hence does not contain such synonyms.

[1] Whenever we refer to synonyms in this paper, we refer to attribute synonyms as opposed to other types of synonyms like entity synonyms.

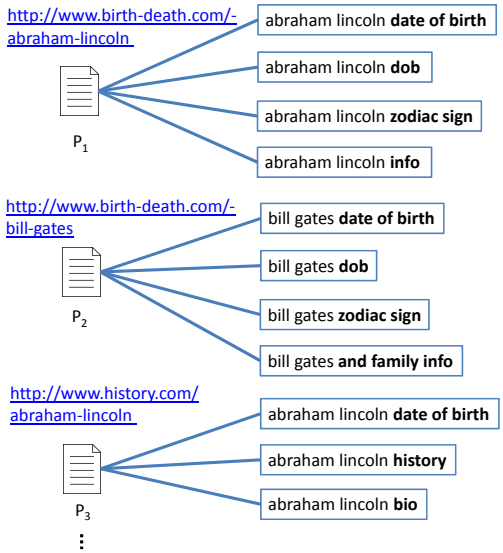

Figure 3: e+a Queries and Clicks (Edges are Clicks)

• *ACSDb*: Cafarella et al. leverages the attribute name correlation statistics (called ACSDb) computed from a large corpus of web tables to compute attribute synonyms [9]. The main positive evidence of synonymity this approach uses is based on the fact that synonymous attribute names are likely to co-occur with the similar context attributes in web table schemas. Our experiments show that such positive evidence is often inadequate as many non-synonyms also co-occur with same attributes. This results in low precision.

Main insights: We propose to derive *positive evidence of synonymity from the query click logs* of a web search engine. Users often issue e+a queries to the search engine. Also, there are many pages on the web that contain information about entities and their attributes (we refer to them as e+a pages). For example, http://www.birth-death.com contains millions of e+a pages with birth date, death date and zodiac sign information of famous people. E+a queries click on e+a pages as they contain the desired information. Figure 3 shows some examples of e+a queries, e+a pages and clicks. Consider an entity e and two attribute names a_1 and a_2 of the given class. *Our key insight is that if a_1 and a_2 are synonyms, users will issue queries $\{e+a_1\}$ and $\{e+a_2\}$ (where "+" denotes string concatenation) and click on the same pages; this is because both queries seek the same information and the same pages contain that information.* For example, in Figure 3, {bill gates dob} clicks on the same page (http://www.birth-death.com/-bill-gates) as {bill gates date of birth}. Since we are given a set of entities, we can further aggregate the positive evidence for a pair of attribute names across all the entities.

However, the positive evidence from query alone is not adequate, because e+a pages often contain information on multiple attributes of an entity. For example, each page on http://www.birth-death.com contains information on birth date, death date and zodiac sign of a person. Thus, a page clicked by {bill gates date of birth} will also be clicked by {bill gates zodiac sign} as shown in Figure 3. Even if we are given a set of entities, this happens for all or many of the entities. This approach will likely produce "zodiac sign" as a synonym of "date of birth".

To overcome the above limitation, we complement the positive evidence from the query click logs with *negative evidence derived from web tables*. In particular, because two attributes that co-occur frequently in same web tables are

unlikely to be synonyms (e.g., it would be meaningless to include both "dob" and "date of birth" in the same table), we use attribute co-occurrence in tables as negative evidence.

Lastly, we observe that attribute synonyms in the context of an entity class are *transitive*, which is a useful constraint to further improve synonym discovery. The main technical challenge we address is to formulate a principled problem that can effectively utilize both positive and negative evidences, while also leveraging the global transitivity property.

Contributions: Our main contributions are as follows:

• We formalize the attribute synonym discovery problem as a holistic optimization problem on a graph with weighted edges. Our problem formulation effectively combines positive and negative signals from web tables and query logs, and optimize globally taking into account *synonym transitivity* observed by class-specific attribute synonyms.

• We develop a linear program-based algorithm for the optimization problem, and we formally show that the proposed algorithm has bi-criteria approximation guarantees.

• We study a variant of the attribute synonym discovery problem, namely *attribute synonym discovery with anchors*, where the goal is to find synonyms of *a given set of distinct attributes*. This can be used to discover attribute synonyms for an entity class in a knowledge base, or any given web table. We propose an algorithm to solve this problem variant.

• We perform extensive experiments on diverse entity classes (Section 5), and draw signals from a large-scale query click log from the Bing search engine as well as corpus of 50 million web tables. Our experiments show that our approaches (i) discover attribute synonyms with high precision and recall, (ii) significantly outperforms the thesaurus lookup approach and the ACSDb approach proposed in [9].

2. PRELIMINARIES AND FRAMEWORK

We first introduce the definitions of entity class and instances, query log and web table corpus. We then present the two-step framework of attribute name extraction and attribute synonym discovery.

2.1 Definitions

Entity class and instances: We assume that the entities of the world are organized into classes (e.g., Country, Person). We perform attribute synonym discovery for a given entity class. We assume a set of entity instances of the particular class to be provided as input. For example, the entity instances can be obtained from a knowledge base.

Query log: The query click log collects the click behavior of millions of web searchers over a long period of time. We use two years' worth of click logs from Bing. We assume the query log $\mathcal{Q}$ to contain records of the form (q, u, cid) where q is a query string, u is a web document, represented by its unique url, and cid is a unique click-id. We say a query q is a co-clicked query (or simply co-click) of a query q' if q and q' click on the same url.

Web table corpus: The web tables corpus $\mathcal{T}$ contains all the HTML tables extracted from the web. We use a corpus of 50 million tables extracted from a part of Bing's web crawl. Each web table $T \in \mathcal{T}$ has (i) a schema S_T which is an ordered list of column names $[h_1, h_2, ..., h_n]$ and (ii) a set of subject entities E_T which are the values in the "subject column" of the table [26]. One or more of the column names can be empty strings. A single site often have many tables with the same schema; we retain a schema only once per url domain in order to prevent a single schema swamping the co-occurrence statistics [9].

Matching Patterns
a e
e a
e's a
a in e
a of e
a for e

Table 1: Example e+a query patterns

2.2 Two-step Framework

We adopt a framework consisting of two steps:

Step 1: Attribute name extraction: Given an entity class and a set $\mathcal{E}$ of instances of that class, this step extracts all the attribute names for that class. We use a variant of prior techniques for this step [17, 18, 20]. We identify e+a queries in the query log containing one of the input entities. Such queries typically follow one of the patterns listed in Table 1. For each such query, we extract the attribute name from the remainder of the query. For example, if "barack obama" is an input entity, the query {barack obama date of birth} matches the query pattern and hence we extract "data of birth" as a candidate attribute name.

However, the candidates so generated often contain many noisy non-attributes. For example, for the entity "barack obama", there are many queries like {barack obama news} or {barack obama twitter}, where "news" and "twitter" do not correspond to attribute names.

We eliminate such non-attribute names by applying two simple but effective filtering techniques:

• *Web Table Column Name Filtering:* We first identify the web tables containing entities of the given class. We do this by checking for sufficient overlap of the entities in its subject column with the set $\mathcal{E}$ of input entities. In our experiments, we use a threshold of 4 overlapping entities. We can then use column names of these tables to filter out non-attribute candidates. For example, we can accept only those candidate attribute names that occur as a column name in these tables. This will filter our candidates like "news" and "twitter" as they are unlikely to be column names in web tables. Other options include accepting candidate attribute names that approximately match table column names.

• *Question Pattern Filtering:* Since users often use question style queries to ask for certain attribute of an entity (e.g., {when was barack obama born}), such question patterns are also useful attribute synonyms (e.g., "when born"). Because such question patterns typically do not occur as columns names in web tables, we additionally include candidate attributes matching predefined question patterns. For example, we accept candidate names that begin with "how", "what", "who", "when", "where", "which", etc.

Step 2: Attribute Synonym Discovery: This step identifies the synonyms using the attribute names extracted from the step above. For example, as shown in Figure 2, given the extracted attribute names {date of birth, income, hometown, salary, birthplace, when born, birth date, pay, birthday, tax, dob, earnings, birth place, zodiac sign}, this step might identify the following sets of synonyms: {date of birth, birth date, birthday, dob, when born}, {income, earnings, salary, pay}, {birth place, home town, hometown, birthplace}, {zodiac sign} and {tax}. This synonym discovery step is the main focus of this work, which we will discuss in the rest of this paper.

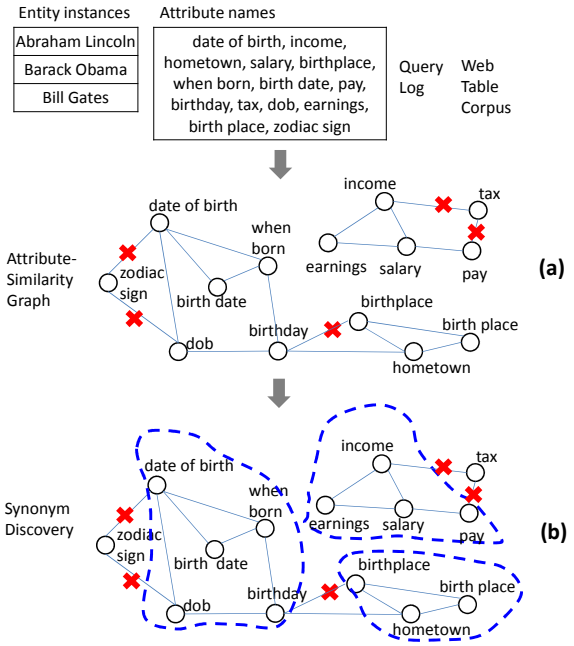

Figure 4: (a): Attribute similarity graph (middle), and (b): attribute synonym discovery (bottom)

3. ATTRIBUTE SYNONYM DISCOVERY

In this section, we discuss the second step of our framework, which is to discover synonyms given valid attribute names (from step 1). We will first describe how we build an attribute-similarity graph to model the likelihood of synonymity, and then discuss our holistic optimization formulation that finds synonyms using the graph.

3.1 Attribute Similarity Graph

Given all attributes of an entity class, we model these attributes and their similarity relationships as a graph, where each vertex corresponds to an attribute, and each edge represents the similarity relationship between a pair of attributes for synonymity (Figure 4(a) shows one such graph, which will be discussed in detail).

To compute similarity relationship between any two attributes, we combine positive similarity signals derived from query logs, and negative similarity signals from web tables. We will describe these two signals next in turn.

Positive Similarity: To determine what attributes are likely to be similar, we leverage the rich user interactions in the query logs. Search engine users often use different ways to seek for the same attribute-level information, and ultimately click on the same pages with the desired information. For example, users searching for {bill gates date of birth} and {bill gates dob} often click on the same pages (e.g., http://www.birth-death.com/-bill-gates, as shown in Figure 3).

Let $q(e, a)$ be an e+a query with entity e and attribute a. In general, if $q(e, a_i)$ and $q(e, a_j)$ co-click on a similar set of pages, then a_i and a_j are more likely to be synonyms. We use this intuition to determine the positive similarity between two attributes.

Let $\mathcal{Q} = \{(q, u, cid)\}$ be the query logs where each entry has a triple consisting of a user query q, a page url u, and a unique query-url click-id cid. We can write the multi-set of pages clicked by query q as $M(q)$:

$$M(q) = \{u | (q, u, cid) \in \mathcal{Q}\}$$

Let $PosSim(a_i, a_j|e)$ be positive similarity of a_i, a_j for a given entity e that we need to compute, we observe that the similarity of the multi-set of pages clicked by $q(e, a_i)$ and $q(e, a_j)$ is often a good proxy. Namely

$$PosSim(a_i, a_j|e) = Sim(M(q(e, a_i)), M(q(e, a_j)) \quad (1)$$

where Sim can be any similarity function for two multi-sets. For example, we can instantiate Sim as Cosine similarity, Jaccard similarity, or distributional metrics like Jensen-Shannon distance, etc. We use Cosine for positive similarity in this work.

Since we are give a set of entities $\mathcal{E}$ as input, we can further aggregate the similarity of a_i and a_j across all input entities.

$$PosSim(a_i, a_j|\mathcal{E}) = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} (PosSim(a_i, a_j|e)) \quad (2)$$

This aggregation generates a robust signal of positive similarity between a_i and a_j, especially when given a large set of entities (e.g., all entities of a class from knowledge base). For simplicity we will omit $\mathcal{E}$ and simply write $PosSim(a_i, a_j)$ when $\mathcal{E}$ is clear from the context.

EXAMPLE 1. *We use the example in Figure 3 to illustrate positive similarity. For simplicity suppose all query-url click edges are of frequency 1. Suppose we want to compute the positive similarity of a_1 = "date of birth" and a_2 = "dob". Let entity e = "abraham lincoln", then the two e+a queries are $q(e, a_1)$ = {abraham lincoln date of birth}, $q(e, a_2)$ = {abraham lincoln dob}, and the multi-set of pages clicked by the two queries are $M(q(e, a_1)) = \{P_1, P_3\}$, $M(q(e, a_2)) = \{P_1\}$. Using Equation (1) and Cosine similarity, we can compute $PosSim(a_1, a_2|e) = 0.7$, or the similarity of "date of birth" and "dob" given "abraham lincoln" is 0.7.*

Similarly, let entity e' = "bill gates", we can compute $PosSim(a_1, a_2|e') = 1$, because $M(q(e', a_1)) = \{P_2\}$, and $M(q(e', a_2)) = \{P_2\}$. Averaging across these two entities using Equation (2), we have $PosSim(a_1, a_2) = 0.85$.

While page co-click information provides valuable positive signals, we observe that in reality the same page often contains information about different attributes, rendering co-click positive similarity inadequate for high quality synonyms. For instance, page P_1 in Figure 3 contains a variety of attribute information for the same person. As a result, not only are queries with attributes "dob" and "date of birth" clicking on these pages, but also queries with attributes such as "zodiac sign". Since "zodiac sign" and "date of birth" also share co-clicks they will generate non-trivial positive similarity scores.

One approach to mitigate this effect is to identify these overly-broad pages (e.g., Wikipedia page) and discard them. Empirically we discard pages that are frequently clicked by entity-only queries $e \in \mathcal{E}$ (e.g., "bill gates"), as well as pages that are clicked by a significant fraction of distinct attributes for the entity class. Such pages are likely to be overly generic and unsuitable for positive score computation.

While this page-filtering approach alleviates the problem of noisy signals from page co-click to some extent, it does not fully address it. We introduce another set of negative signals obtained from web tables to help synonym discovery.

Negative Similarity: We observe that a pair of true synonyms will rarely co-occur as column names in the same web table schema, since it would be useless to duplicate identical columns in a single table (e.g., "dob" and "date of birth"

are unlikely to co-occur in the same table). Utilizing this observation, we derive negative signals if attributes a_i and a_j co-occur sufficiently frequently as column names in same tables.

We use the standard point-wise mutual information (PMI) to measure the strength of correlation. Let $p(a)$ denote the fraction of those tables containing a as a column name, and $p(a_i, a_j)$ be the fraction of tables containing both a_i and a_j as column names. The PMI score between a_i and a_j is defined as follows.

$$\text{PMI}(a_i, a_j) = log \frac{p(a_i, a_j)}{p(a_i)p(a_j)} \quad (3)$$

A positive PMI score indicates that the co-occurrence is more frequent than coincidence, which is a strong negative evidence indicating that the two attributes are unlikely to be synonyms. On the other hand, negative PMI score in this case does not necessarily give much positive evidence for synonymity. We thus use PMI as negative signal only when it is positive, defined as follows.

$$NegSim(a_i, a_j) = \min(-\text{PMI}(a_i, a_j), 0)$$

Combining positive and negative similarities: We use both positive and negative similarity scores via a simple linear combination.

$$w_{ij} = \beta PosSim(a_i, a_j) + (1 - \beta) NegSim(a_i, a_j) \quad (4)$$

Here w_{ij} denotes the combined similarity score of attributes a_i and a_j. A parameter β is used to weight the relative importance of the two components. Empirically we use $\beta = 0.5$, which produces good quality results in our experiments (Section 5).

With the combined similarity score, we can now completely model the relationship between attributes as a graph.

Definition 1. **Attribute-similarity graph**. We use a graph $G = (V, E)$ to model attribute similarity, where each vertex $v_i \in V$ corresponds to an attribute a_i, and each edge $e_{ij} \in E$ has a weight w_{ij} as defined in Equation (4) to represent the similarity between attributes a_i and a_j.

EXAMPLE 2. *Figure 4(a) shows an example of the attribute-similarity graph computed for the entity class* Person*. Each vertex corresponds to an attribute, and the edge represents the combined similarity. We omit the exact edge weights in the graph for simplicity, but use edges with red crosses to indicate negative edges (the two attributes co-occur frequently in same web tables), and solid edges for positive edges if the two attributes have significant query log co-clicks. There are no edges between attribute pairs if they have insignificant positive co-click similarity and insignificant web table co-occurrence.*

3.2 Holistic optimization for attribute synonyms

Edge-based synonyms vs. cluster-based synonyms. Given the attribute-similarity graph, a natural approach is to generate synonyms by finding pairs of attributes that have high similarity scores. This is equivalent to edges in the attribute-similarity graph, and we call this *edge-based synonyms*.

In practice, however, this edge-based approach suffers due to sparse and noisy web data. First, due to query log sparsity, certain synonymous attributes may not share enough co-clicks. Attribute "dob" and "birth date" in Figure 4(a), for instance, do not have enough co-clicks, and an edge-based approach will miss out on such pairs. Furthermore, because the log is often noisy, certain non-synonym attributes may have high co-click similarity. For example, "birthday" has high co-click similarity with "hometown" as in Figure 4(a) (thus the edge between them). An edge-based approach will mistake such pairs as synonyms.

The key problem here is that the edge-based approach only looks at local edge information between a pair of attributes at a time. We can in fact exploit a global property, that attribute synonyms for a given entity class is generally *transitive*, defined as follows.

PROPERTY 1. *Synonyms are transitive, if both of the following two hold true for any distinct $a_i, a_j, a_k \in A$:*

(1) if a_i is a synonym of a_j, a_j is a synonym of a_k, then a_i and a_k must be synonyms; and

(2) if a_i is a synonym of a_j, but a_j is not a synonym of a_k, then a_i and a_k must not be synonyms.

We emphasize that transitivity does *not* hold in general for other types of synonyms without a specific context. For example, an ambiguous term like "mp" can have synonyms like "military police", "member of parliament", or "mega-pixels", etc. Imposing transitivity would require all these expanded forms to be synonyms, which is clearly not true. As such, commercial entity-synonyms offerings like Bing Entity Synonym API [1] do not assume transitivity and makes predictions only on a per-pair basis, which is effectively edge-based synonyms.

However, transitivity does hold in almost all cases for attribute synonyms, mainly because the meaning of attributes are unambiguous given the context of an entity class. For example, for Camera, even short abbreviated attributes like "mp" is unambiguously referring to "mega-pixels".

Because of transitivity, we can actually produce *cluster-based synonyms*, by grouping different synonyms of the same attribute together into clusters. Exploiting this property allows us to optimize globally across all attributes, instead of making local decisions one pair at a time. This often leads to better predictions, as shown in the example below.

EXAMPLE 3. *We revisit the example in Figure 4(a). Although "dob" has low direct similarity with the input attribute "birth date" (thus no edge between them), it does have high similarity with "date of birth", which in turn has high similarity with attribute "birth date". If we look at the graph globally and enforce transitivity, we may predict "dob" and "birth date" as synonyms despite their low co-click similarity (thus mitigating data sparsity).*

Similarly, even though "birthday" has co-clicks with "hometown", because we know "hometown" and "birthplace" are highly likely to be synonyms, and "birthplace" and "birthday" are highly unlikely to be synonyms (due to web table co-occurrence), we may no longer predict 'birthday" and "hometown" as synonyms because of transitivity (thus mitigating noise in data).

Using cluster-based synonyms, we can produce clusters of attributes as synonyms. For example using the attribute-similarity graph in Figure 4(a), we can produce clusters like the one in Figure 4(b). Note that although we would like to produce clusters, standard clustering techniques are not

suitable for this specific problem (Section 6 has more discussions on this). We develop a holistic optimization problem formulation for this task of attribute synonym discovery.

Holistic global optimization. Given that we want to output clusters of attribute synonyms to exploit transitivity, we can define our problem as follows.

Definition 2. (**Attribute Cluster Discovery**). Given the attribute-similarity graph $G = (V, E)$, we want to find a disjoint partitioning of V, denoted as $\mathcal{S} = \{S_1, \ldots, S_m\}$, such that each cluster contains the attribute synonyms of a unique attribute, and no two clusters correspond to the same attribute.

Given that there are many possible partitionings, we need to determine which partitioning is more desirable by defining the "quality" of clusters of attributes. A natural approach is to use the sum of all edges weights inside the cluster. Let $g(S)$ be the sum-of-all-pair score defined as follows:

$$g(S) = \sum_{v_i, v_j \in S, i \neq j} w_{ij} \tag{5}$$

We can then simply aggregate the sum of $g(S)$ scores across all clusters as our objective function, or $\sum_{S_i \in \mathcal{S}} g(S_i)$. Intuitively, this quality score measures the overall similarity of all attributes within a cluster. This is a suitable metric, because given transitivity, a true synonym should have some positive pairwise similarity with most attributes in the same cluster. Thus including a synonym in the right cluster will likely improve our quality metric, leading us to the right cluster for attributes. On the other hand, incorrectly including a non-synonym will introduce negative similarity with most attributes in the cluster, reducing the quality score. As a result, by maximizing this objective function we are likely to find high quality synonym clusters.

Notice that by computing quality scores as sum-of-all-pairs in clusters, we have implicitly factored in transitivity as part of our objective function.

With this objective function, we can write the synonym discovery problem as the following optimization problem:

$$\text{(MAX-AP)} \qquad \max \sum_{i=1}^{m} g(S_i) \tag{6}$$

$$\text{s.t.} \quad S_i \cap S_j = \emptyset, \forall i \neq j \tag{7}$$

$$\bigcup_{i=1}^{m} S_i = V \tag{8}$$

While this formulation is intuitive, there are two shortcomings. First, this is a fixed optimization problem can be solved once to produce only one set of attribute clusters, without offering the flexibility to tweak for a desired level of precision and recall. In practice, we often need to trade-off precision and recall depending on the requirement of an application. The second is a technical reason that this particular formulation is difficult to optimize, as shown in the following theorem using a reduction from Independent Set.

THEOREM 1. *The* MAX-AP *problem described above is NP-hard. Furthermore, the cluster quality score cannot be approximated within a factor of $|V|^{1-\epsilon}$ for some fixed $\epsilon > 0$, unless $P = NP$.*

With these considerations in mind, we slightly change the formulation as follows. Instead of using the all-pair-similarity $g(S)$, we differentiate between *positive edge scores* $g^+(S)$ and *negative edge scores* $g^-(S)$, namely

$$g^+(S) = \sum_{v_i, v_j \in S, i \neq j, w_{ij} > 0} w_{ij} \tag{9}$$

$$g^-(S) = \sum_{v_i, v_j \in S, i \neq j, w_{ij} < 0} w_{ij} \tag{10}$$

Intuitively, $g^+(S)$ and $g^-(S)$ represent the sum of all positive edge scores and all negative edge scores in a cluster S, respectively, and they sum up to the original quality score $g^+(S) + g^-(S) = g(S)$. The score in $g^+(S)$ is similar to our original quality score $g(S)$ where a higher value is more desirable; while $g^-(S)$ measures the sum of "undesirable" edges inside S.

In principle, we would like to maximize $g^+(S)$ while minimize $g^-(S)$, but these are conflicting goals. As clusters grow in size, both $g^+(S)$ and $g^-(S)$ will monotonically increase. A larger $g^+(S)$ typically means that more synonyms are captured, thus better recall. At the same time the precision is likely to suffer as $g^-(S)$ increases. So we try to maximize $g^+(S)$ while limiting $g^-(S)$ to some pre-determined level, using the following formulation.

$$\text{(MAX-CS)} \qquad \max \sum_{i=1}^{m} g^+(S_i) \tag{11}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} g^-(S_i) \leq t \tag{12}$$

$$S_i \cap S_j = \emptyset, \forall i \neq j \tag{13}$$

$$\bigcup_{i=1}^{m} S_i = V \tag{14}$$

Note that we use $g^+(S)$ as the new objective function and $g^-(S)$ as a new constraint. This can be loosely interpreted as we want to maximize recall, while controlling the loss in precision to a certain limit. The parameter t used in constraint (12) limits total $g^-(S)$, which essentially gives us a "knob" to trade-off precision and recall.

We use the following example to illustrate MAX-CS.

EXAMPLE 4. *We revisit the example shown in Figure 4(a). Assume for simplicity all positive (solid) edges are of weight +1, and all negative (crossed) edges are of weight -1.*

Let us first consider the case where our precision threshold t has $t = 0$ in Equation (12)*. This ensures that no negative edges can be ever included in any clusters produced. Given that we need to maximize Equation* (11)*, which sums all intra-cluster positive edges, the best set of clusters possible are depicted in Figure 4(b), namely, we have 5 attributes in the "birth date" cluster (with a $g^+(S_i)$ score of 6 since there are 6 edges), 4 attributes in the "income" cluster (score 4), 3 in "birth place" (score 3), and two other singleton clusters, "tax" and "zodiac sign". It can be verified that this solution has a score of 13 as defined in Equation* (11)*, and is in fact the optimal solution to* MAX-CS *with $t = 0$.*

Suppose we change the threshold to $t = 1$ instead, which allows us to include one negative edge inside a cluster. It can be verified that the best solution is to merge the "birth date" cluster with the "income" cluster (which will include

one more intra-cluster edge between "birthday" and "birth place" compared to the previous solution), to produce a total score of 14 as defined in Equation (11)*. As can be seen here, higher t tends to produce results with lower precision (but with higher recall).*

Although the MAX-CS formulation captures what we need, this problem is also hard as shown in the following theorem.

THEOREM 2. *The* MAX-CS *optimization problem described above is NP-hard.*

In order to better solve this problem, we change the formulation using standard metric embedding methods as follows. We introduce a new set of binary variables, $d_{ij} \in \{0, 1\}$, to represent the distance between any two vertices v_i and v_j with $i < j$, and $i, j \in [n]$, where distance 0 indicates that v_i and v_j are in the same cluster, and 1 otherwise. We can transform MAX-CS to the following problem MAX-ECS.

$$\text{(MAX-ECS)} \quad \max \sum_{w_{ij}>0} (1 - d_{ij}) w_{ij} \tag{15}$$

$$\text{s.t.} \quad \sum_{w_{ij}<0} (1 - d_{ij}) w_{ij} \leq t \tag{16}$$

$$d_{ij} + d_{jk} >= d_{ik}, \forall i < j < k \tag{17}$$

$$d_{ij} \in \{0, 1\}, \forall i < j \tag{18}$$

Note that using the change of variables described above, the objective function in Equation (15) and the budget constraint in Equation (16) directly correspond to Equation (11) and Equation (12) in MAX-CS, respectively. Furthermore, the triangle inequality in Equation (17) ensures that for any feasible solution to MAX-CS, we will have a corresponding feasible solution to MAX-ECS, and the same is true also in the other direction. This guarantees that MAX-ECS has the same optimal solution as MAX-CS.

We can further define MIN-ECS as the loss minimization version of MAX-ECS, by changing the objective function as follows.

$$\text{(MIN-ECS)} \quad \min \sum_{w_{ij}>0} d_{ij} w_{ij} \tag{19}$$

$$\text{s.t.} \quad \sum_{w_{ij}<0} (1 - d_{ij}) w_{ij} \leq t \tag{20}$$

$$d_{ij} + d_{jk} >= d_{ik}, \forall i < j < k \tag{21}$$

$$d_{ij} \in \{0, 1\}, \forall i < j \tag{22}$$

Using MIN-ECS, we can design an LP-based algorithm with bi-criteria approximation guarantees. The algorithm works as follows. We first replace the integral variables d_{ij} with fractional variables $\overline{d}_{ij}$, and replace the corresponding integrality constraint in Equation (22) with fractional constraints $\overline{d}_{ij} \in [0, 1]$. This gives us a linear program, which we can solve optimally using standard LP-solvers in polynomial time, and obtain optimal fractional solutions denoted by $\overline{d}^*_{ij}$.

We then apply the classical region growing technique [7, 15, 16, 25] to round the resulting fractional solution $\overline{d}^*_{ij}$ into an integral solution without losing too much in quality. This procedure is described in Algorithm 1.

In Algorithm 1, we start by solving the fractional MIN-ECS for $\overline{d}^*_{ij}$. We then initialize the set of unassigned vertices U as V. We iteratively pick a random vertex v_i from U. Let

Algorithm 1 LP to approximate MIN-ECS

```
Region_Grow (Attribute-similarity graph G = (V, E))
solve the LP of fractional MIN-ECS for G
initialize an unassigned set of vertices U = V
initialize result set B = ∅
while U ≠ ∅ do
    randomly pick v_i ∈ U, r = 0
    while cutwgt(b(i, r)) > cln(n + 1)vol(b(i, r)) do
        r = r + Δr
    end while
    U = U \ b(i, r)
    B = B ∪ b(i, r)
end while
return B as the set of vertex clusters
```

$b(i, r)$ denotes the ball of radius r around vertex v_i, which is defined as the union of all vertices v_j whose distance to v_i is within radius, or $b(i, r) = \{v_j | v_j \in U, d_{ij} <= r\}$. Let the cut of a ball b be the set of positive edges with one endpoint in b. The sum of weights of such cut edges is denoted as cutwgt(b). Lastly, let the volume of a ball $vol(b(i, r))$ be the sum of weighted distance of positive edges belonging to the ball. For a positive edge with both endpoints v_j and v_k in the ball, it contributes $d_{jk} w_{jk}$ to $vol(b(i, r))$. For an cut edge (v_j, v_k) where $d_{ij} < r$, it contributes $w_{jk}(r - d_{jk})$ to $vol(b(i, r))$. In addition, an intial volume $\frac{F}{n}$ is included in each ball, where $F = \sum_{v_i, v_j \in V} w_{ij} d_{ij}$.

For the randomly picked vertex v_i, we use v_i as the center of a ball and iteratively increase the radius r, until cutwgt($b(i, r)$) <= $cln(n + 1)vol(b(i, r))$, at which point we stop, remove all vertices covered by $b(i, r)$ from U, and add $b(i, r)$ as a newly created synonym cluster to our result set B. We iterate with this ball-growing process until all vertices in U are exhausted. Note that in the output B, synonym clusters for all attributes have already been created naturally.

This approach has a $(\frac{1}{c}O(\log(n)), c)$ bi-criteria approximation guarantee. This means that if f^* is the optimal objective value of the loss-minimizing MIN-ECS given a budget t as in Equation (20), then Algorithm 1 can find a solution with an objective value no more than $\frac{1}{c}O(\log(n))f^*$ while violating the given budget t by at most a factor of c.

THEOREM 3. *Algorithm 1 is a* $(\frac{1}{c}O(\log(n)), c)$ *bicriteria approximation algorithm to MIN-ECS.*

We prove this result using techniques similar to ones first developed in [7, 25]. We omit details of the proof here due to space limitations. A proof of this theorem can be found in the full version of this paper.

4. ATTRIBUTE SYNONYM DISCOVERY WITH ANCHORS

An interesting variant of the problem is to discover synonyms for a few known attributes, with the knowledge that these are distinct attributes (i.e., they are not synonyms to each other). This corresponds to the natural scenario of discovering synonyms for a given web table, or a given knowledge base, where the user may only be interested in finding synonyms for the attribute names present in that table/knowledge-base. We call these given attributes `anchors`, and term this problem *Attribute Synonym Discovery with Anchors*.

Definition 3. (**Attribute Cluster Discovery with Anchors**). Given the attribute-similarity graph $G = (V, E)$, and anchor attributes $A_s = \{a_1, \ldots, a_m\} \in V$ for which synonyms need to be discovered, compute a disjoint set of attributes $\{S_1, \ldots, S_m\}$ such that cluster S_i contains all attribute synonyms of a_i, $\forall i \in [m]$.

For conciseness, we will simply refer to this problem as the anchored variant, and the problem discussed in Section 3 as the general attribute-synonym problem when the context is clear.

EXAMPLE 5. *Suppose there is a table with entities {Abraham Lincoln, Barack Obama, Bill Gates}, and the set of column names is {date of birth, place of birth}, which are input anchor attributes in our problem. The output only consists of synonyms for these two anchor attributes and disregards other attributes.*

While this anchored problem variant is very similar to our general attribute synonym problem in Section 3, there are subtle differences that make this anchored variant different.

First, in the anchored variant, all attributes in the anchor set A_s are known to be distinct attributes and cannot be synonyms. This effectively introduces constraints to our problem, by forcing anchors into different clusters. Compared to the general attribute-synonym problem, such constraints can potentially allow us to produce clusters of higher quality (e.g., in the general problem, an algorithm may confuse between "date of birth" and "place of birth" by thinking that they are synonyms. This will not happen if they or their synonyms are provided as anchors). The new constraints induced by anchors, however, do make the problem more difficult to solve in a technical sense.

Second, since the goal here is to only find synonyms for the anchor set A_s, as opposed to all attributes in the universe, the objective function will also change to reflect this focus, which also provides opportunities of finding better synonyms for a targeted attribute set.

4.1 Optimization formulation

We use the same attribute-similarity graph $G = (V, E)$ described before, and also use the *positive similarity score* $g^+(S)$ (Equation (9)), and *negative similarity score* $g^-(S)$ (Equation (10)) defined for a cluster as in the general attribute synonym problem.

Given an anchor set $A_s = \{a_1, \ldots, a_m\}$, we can define the anchored variant as the following optimization problem.

$$\text{(MAX-ACS)} \quad \max \sum_{i=1}^{m} g^+(S_i) \tag{23}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} g^-(S_i) \leq t \tag{24}$$

$$a_i \in S_i, \forall i \in [m] \tag{25}$$

$$S_i \cap S_j = \emptyset, \forall i \neq j \tag{26}$$

$$S_i \subset V, \forall i \tag{27}$$

In this anchored problem variant, we only care about synonym clusters for anchor attributes, as reflected in the objective function. Notice that a new subset constraint in Equation (27) replaces the partitioning constraint in Equation (14) used in the general problem MAX-CS.

The problem is unfortunately intractable and inapproximable under reasonable complexity assumptions. Furthermore, we could not apply the rounding technique used in MAX-CS and MAX-ECS for the general problem to the MAX-ACS problem in a similar manner, because of the constraints induced by anchors require no two anchors be assigned to the same cluster, which makes the optimization problem more difficult to solve.

In light of these, we propose the following method to optimize MAX-ACS. First, we assign each anchor to a cluster, so that each cluster initially has only one vertex in it. Then for each unassigned attributes, we check for each attribute-cluster combination, to find the pair that provides the most score gain in $g^+(S)$, without violating the budget t in $g^-(S)$. After enumerating all such pairs, we assign the attribute with the best score gain to the corresponding cluster. We repeat this process until we cannot find an attribute to assign without violating budget t. The resulting clusters are output as synonyms. Due to space constraints, more discussion of this algorithm and its pseudo-code can be found in the full version of this paper.

Class	Example Entities
building	sears tower, space needle, ...
disease	bronchitis, diabetes, flu, ...
organisation	microsoft, oracle, google, ...
person	tom hanks, bill gates, ...
country	india, canada, mexico, ...
celestial object	mars, moon, jupiter, ...
education institution	stanford university, ...
chemical compound	glucose, gypsum, galena, ...

Table 2: Input Classes and Example Entities

Class	Cluster	Example Output Attribute Synonyms
person	1	salary, annual income, pay, ...
	2	height, how tall, how tall is, ...
	3	contact, email address, how to contact, ...
	4	race, nationality, what ethnicity is, ...
	5	how rich is, worth, networth, ...
disease	1	what are the symptoms, symptoms, sign, ...
	2	how to cure, how do you treat, ...
	3	cause, how do you get, what causes, ...
	4	prevention, how to prevent, ...
	5	definition, what is, ...
organisation	1	phone no, what is the phone number, ...
	2	ticker, symbol, stock symbol, ...
	3	jobs, employment, ...
	4	headquarters, hq, ...
	5	contact, contact info, ...

Table 3: Example Output Synonym Clusters

5. EXPERIMENTS

5.1 Experimental Setup

We evaluate our system using 8 different entity classes, representing a diverse range of entities. These classes and their sample entities are listed in Table 2. The sample entities are obtained from Bing's knowledge base, Satori [4], which is an in-house knowledge base developed in Microsoft that is conceptually similar to knowledge bases like Freebase [8] or YAGO [22].

We discover attribute synonyms mainly leveraging query logs and web tables. Specifically, we use two-years' worth of query logs from Bing, and 50 million web tables extracted from a recent snapshot of Bing's index [29] to compute statistical similarity scores.

5.2 Attributes Synonym Discovery Evaluation

In order to give readers some concrete ideas of synonyms produced by our approach, in Table 3 we list top 5 clusters produced by our Algorithm 1 for the attribute synonym dis-

covery problem. As can be seen from the table, there are many interesting synonyms discovered for a wide variety of attributes, and many of them are in fact non-trivial (e.g., "salary", "pay", "annual income", etc.).

To quantitatively evaluate the quality of result clusters produced, we need to manually label synonyms produced as either true synonyms or false positives. Since some entity classes have hundreds of attribute clusters, manually labeling them exhaustively are very expensive. For this experiment we manually label the top 5 output clusters that have the highest number of attributes for each class tested. This results in a total of 1356 synonym pairs labeled across all entity classes tested.

For each synonym cluster, we compute the precision p, as $\frac{\#\text{ true synonyms in cluster}}{\#\text{ total attributes in cluster}}$, and report an average precision over that of all produced clusters. We also report recall using the average number of true synonyms in these clusters, instead of the standard relative recall, since in some cases we could not be sure that all possible synonyms of an attribute have been exhaustively enumerated.

We compare the following methods for attribute synonyms.

• **HolisticOpt (Holistic optimization):** This method uses our holistic optimization formulation MAX-CS, and executes our linear-program based algorithm (Algorithm 1), which has bi-criteria approximation guarantees for loss minimization. We use the Enterprise version of Microsoft Solver Foundation 3.1 [3] to solve the associated LP.

• **T-Link (QL+WT) (Thresholding with link-based clustering, using query logs and web tables):** This corresponds to an approach that uses query logs and web tables to build attribute-similarity graph in the exact same way as HolisticOpt. However, instead of using our optimization formulation, this approach simply uses thresholds to determine what attribute pairs (edges) are synonyms, and then apply transitivity using link-based clustering (single-link in this case) to determine synonym clusters. Since it operates on the exact same graph as HolisticOpt, the comparison with HolisticOpt will reveal the usefulness of Algorithm 1.

• **T-Link (QL only) (Thresholding with link-based clustering, using query logs only):** This is the same as T-Link (QL+WT), except that instead of using both query logs and web tables to build attribute-similarity graph, this approach uses only the query logs. So comparing this with T-Link (QL only) will shed some light on the usefulness of the negative signals derived from web tables.

• **Thesaurus (Thesaurus-based Lookup):** Given a ground truth cluster of attribute synonyms, we look up synonyms in Wiktionary [5] for each attribute in the cluster. We then apply transitivity (single-link) to all pairs so discovered to produce result clusters.

• **ACSDb [9] (WebTable-based Synonym Finder):** In the context of the pioneering work on harvesting web tables [9], the authors discussed an interesting approach that uses context attributes for synonym finding. While attribute synonyms is not the focus of [9], this approach is nevertheless relevant and we implemented the algorithm using 50 million web tables extracted from a part of Bing's index.

In order to study the precision and recall trade-off, we vary different parameters used by these algorithms and evaluate cluster quality (e.g., we vary precision threshold t for HolisticOpt and edge-score threshold for T-Link).

The precision-recall results of all methods are shown in Figure 5. As we can see, the proposed HolisticOpt approach

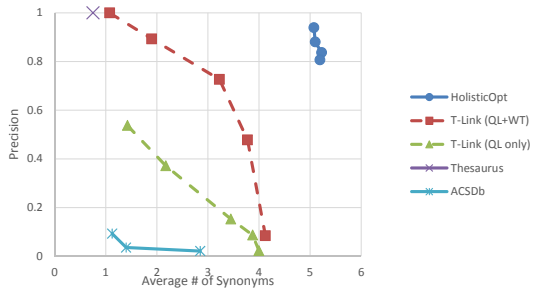

Figure 5: Results for general attribute synonyms

clearly outperforms all alternatives. On average, it discovers over 5 synonyms per cluster for top clusters with high precisions.

T-Link (QL+WT) has the second best performance, but clearly lags behind HolisticOpt. Recall that it works on the exact same graph as HolisticOpt, this demonstrates the effectiveness of our proposed algorithm, that not only has approximation guarantess, but also good empirical performance. T-Link (QL only) performs significantly worse compared to T-Link (QL+WT), showing that the negative signals from web tables are actually very useful since the query click logs are often quite noisy.

The Thesaurus approach is shown as a dot on the top-left corner of Figure 5. On average it only recovers 0.75 synonym per cluster but does have very high precision. This is expected, as it is a manually curated approach that is unlikely to have false-positives. However, because Thesaurus uses general dictionary without considering the specific context of the class of entities, it fails to return synonyms specific to the entity class. For instance, it does not output "etiology" for attribute "cause" in the "disease" class. This shows the recall limitation of a static dictionary-based approach.

The performance of ACSDb is not very satisfactory for the cases we tested. Our observation is that its main positive signals derived from table context attributes (if both attributes a and b co-occur with c often then they are likely synonyms) are often not a positive evidence reliable enough for synonymity. The query co-clicks signals we use appear to be more robust, and our global optimization formulation also helps to improve performance.

Class	Precision	Average # Synonyms
building	0.967	3.4
disease	1	4.2
organisation	0.942	6
person	0.88	5.6
country	0.93	7.56
celestial object	0.9	4.2
education institution	0.835	4.8
chemical compound	0.96	6.2

Table 4: Quality of Top 5 Clusters

We now drill down to results for each class tested. Table 4 shows average precision and recall across the top 5 clusters for each class. As we can see, precision is consistently high across all cases, and a good number of synonyms are generated per cluster in all the cases (typically 4-6). This shows that our proposed approach is effective for different classes of entities.

5.3 Attribute Synonym with Anchors

We also experimentally evaluate our approach discussed in Section 4 for the anchored problem variant.

Class	Anchors	Example Output Attribute Synonyms
person	date of birth income profession	birth date, birthdate, dob, birthday earnings, annual salary, pay job, career
disease	cause symptoms treatment	how do you get, etiology, what causes sign, what are the signs how to treat, how to cure, therapy
organization	ticker ceo headquarters	stock symbol, what is the symbol leadership, president hq, location, head quarters

Table 5: Example Anchors and Synonym Output

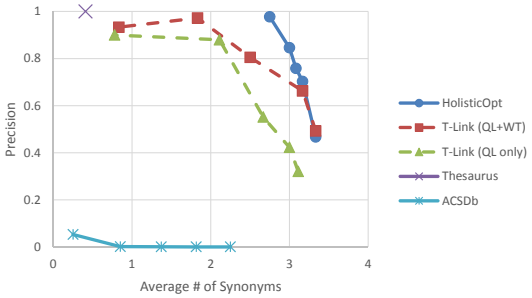

Figure 6: Results for attribute synonyms with anchors

For this anchored variant, we randomly select 3 common attributes from each entity class as given anchor attributes, and label other predicted synonyms as true synonyms or not. Sample input anchors and output synonyms are shown in column 2 and column 3 of Table 5, respectively.

We also compare our approach **HolisticOpt**, with alternatives including **T-Link (QL+WT)**, **T-Link (QL only)**, **Thesaurus**, and **ACSDb**.

Figure 6 shows the precision/recall results when varying parameters. HolisticOpt again has the best performance, and the general trend observed in this graph is consistent with that observed in Figure 5. We note that the performance gap between HolisticOpt and T-Link based approach narrows. This is partly because when evaluating precision/recall relative to a given attribute, the evaluation becomes local, such that using local information is almost as good as global optimization, partially erasing the benefit of our holistic optimization formulation.

6. RELATED WORKS

While there are no research efforts exclusively focusing on the problem of attribute synonyms, a number of prior works discuss intuitive methods for finding attribute synonym in the context of other problems. For example, Cafarella et al. propose the novel idea of using web table schemas to compute attribute synonyms [9]. Their approach is based on the observation that (i) synonymous attributes are unlikely to appear together in the same schema (ii) synonyms are likely to co-occur with the same context attributes with similar frequencies. Our experiments show that in some cases this technique has low precision. We believe this is mainly because the context-based positive signal (i.e., (ii) above) is often inadequate, because many non-synonyms also co-occur with same attributes. For example, we find "gender" and "date of birth" co-occur with a similar set of attributes (e.g., "name", "country", "zodiac sign"), yet the two attributes almost never co-occurs in the same schema; hence, this technique outputs "gender" and "date of birth" as synonyms. In comparison, our query-click-based approach uses more reliable positive signals and a global optimization formulation that produces high-quality synonyms.

Biperpedia [17] studies the important problem of large-scale attribute name extraction. The authors touch upon the related issue of attribute synonyms and discuss a simple supervised approach, but state that the topic warrants in-depth studies. Their supervised approach uses as the main ingredient a query-expansion-related feature (the set of related queries suggested). The main differences of their method and our method are two-fold. First, their approach is a supervised ML approach that requires expensive manual labeling, whereas our approach is unsupervised that does not require labels. Second, the query-expansion feature used is in fact often derived from query co-clicks [13], thus similar to our query log based positive signals. Our experiments show that query-log alone is often inadequate, combining query-logs, web tables and transitivity in a principled global optimization achieves the best performance.

The related problem of discovering *entity* synonyms has been extensively studied, e.g., [10, 11, 12, 19, 24], where techniques such as documents co-occurrence [24], document contexts similarity [19], and query co-click [10, 12] are used. It is not straightforward to apply these techniques to attribute synonyms. For example, users typically do not issue attribute-only queries, and techniques in [10, 12] are thus not directly applicable.

Dicitionary look-up (e.g., Wiktionary [5] or Merriam-Webster [2]) is a valid approach for attribute synonyms. For example, Wiktionary lists "dob" and "birth date" as synonyms of "data of birth". However, attribute synonyms are often specific to an entity class. For example, "etiology", "cause" and "what triggers" are synonyms only for the class medical conditions. A thesaurus lookup is static and does not adapt to the context, and is thus insufficient.

There is also a long line of work on the important problem of attributes name extraction [6, 14, 17, 20, 21, 27], where techniques used include linguistic pattern ("what is the a of e"?) and query patterns ("the a of e"), etc. The attribute names generated by these approaches can be used as input by our attribute synonym discovery (we in fact use a variant of these techniques), and is thus complementary to the problem studied here.

Although in our problem of synonym discovery, we need to produce clusters as output, standard clustering techniques such as single-link, average-link, correlation clustering, etc., are not suitable. For example, single-link enforces transitivity blindly without carefully optimizing for pair compatibility, which results in poor performance as shown in our experiments. Correlation clustering is also not suitable for this specific application, because its formulation does not offer flexible tradeoff of precision and recall, and its objective function mixes precision and recall qualities, making holistic optimization difficult.

7. CONCLUSION

In this paper, we study the problem of automatic discovery of attribute synonyms. We present a novel solution that leverages the power of query click logs and millions of web tables, and optimizes synonym decisions globally, with approximation guarantees. Our experiments show that our approach is significantly better than alternative methods.

Our work can be extended in multiple directions. We only leverage web table schema information in this work; utilizing column values may present intersting opportunites. Evaluating the impact of attribute synonyms on search quality is another useful item for future work.

8. REFERENCES

[1] Bing Synonyms API. https://datamarket.azure.com/dataset/bing/synonyms.

[2] Dictionary and Thesaurus - Merriam-Webster Online. http://www.merriam-webster.com/.

[3] Microsoft solver foundation. http://msdn.microsoft.com/en-us/library/ff524509.aspx.

[4] Satori. http://news.cnet.com/8301-10805_3-57596042-75/microsofts-bing-seeks-enlightenment-with-satori/.

[5] Wiktionary. http://www.wiktionary.org/.

[6] A. Bakalov, A. Fuxman, P. P. Talukdar, and S. Chakrabarti. Scad: collective discovery of attribute values. In *Proceedings of WWW*, 2011.

[7] Y. Bejerano, M. A. Smith, J. Naor, and N. Immorlica. Efficient location area planning for personal communication systems. In *Transaction of Networking*, 2006.

[8] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, 2008.

[9] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *PVLDB*, 2008.

[10] K. Chakrabarti, S. Chaudhuri, T. Cheng, and D. Xin. A framework for robust discovery of entity synonyms. In *Proceedings of the 18th ACM SIGKDD conference*, 2012.

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

[12] T. Cheng, H. W. Lauw, and S. Paparizos. Entity synonyms for structured web search. *TKDE*, 2011.

[13] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *Proceedings of WWW*, 2002.

[14] O. Etzioni et al. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of WWW*, 2004.

[15] N. Garg, V. V. Vazirani, and M. Yannakakis. Multiway cuts in directed and node weighted graphs. In *Automata, Languages and Programming*, pages 487–498. Springer, 1994.

[16] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi) cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.

[17] R. Gupta, A. Halevy, X. Wang, S. Whang, and F. Wu. Biperpedia: An ontology for search applications. In *Proc. 40th Int'l Conf. on Very Large Data Bases (PVLDB)*, 2014.

[18] T. Lee, Z. Wang, H. Wang, and S. won Hwang. Attribute extraction and scoring: A probabilistic approach. In *International Conference on Data Engineering (ICDE)*, 2013.

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

[20] M. Pasca. Organizing and searching the world wide web of facts step two: Harnessing the wisdom of the crowds. In *Proceedings of WWW*, 2007.

[21] M. Pasca and B. V. Durme. Weakly-supervised acquisition of open-domain classes and class attributes fromweb documents and query logs. In *Proceedings of ACL*, 2008.

[22] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *Proceedings of WWW*, 2007.

[23] I. Trummer, A. Halevy, H. Lee, S. Sarawagi, and R. Gupta. Mining subjective properties on the web. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1745–1760, New York, NY, USA, 2015. ACM.

[24] P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. *CoRR*, cs.LG/0212033, 2002.

[25] V. Varizani. *Approximation algorithms*. springer verlag, 2001.

[26] P. Venetis et al. Recovering semantics of tables on the web. *Proc. VLDB Endow.*, pages 528–538, 2011.

[27] J. Wang, H. Wang, Z. Wang, , and K. Zhu. Understanding tables on the web. In *International Conference on Conceptual Modeling*, 2012.

[28] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012.

[29] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. *SIGMOD*, pages 97–108, 2012.

[30] X. Yin, W. Tan, and C. Liu. Facto: a fact lookup engine based on web tables. In *WWW*, 2011.