Web Scale Entity Resolution using Relational Evidence

Taesung Lee 1,‡, Zhongyuan Wang † Haixun Wang †, Seung-won Hwang †
1POSTECH, Korea, Republic of
{elca4u,swhwang}@postech.edu
‡Microsoft Research Asia
Beijing, China
{zhy.wang,haixunw}@microsoft.com

ABSTRACT

Entity resolution has been extensively studied. Many approaches have been proposed, including using machine learning techniques to derive domain-specific lexical similarity measures, or rank entities’ attributes by their discriminative power, etc. In this paper, we study the problem in the setting of matching two web scale taxonomies. Besides the scale, we address the challenge that the taxonomies may not contain enough context (such as attributes) for entity resolution, and traditional lexical similarity measures result in many false positive matches. To tackle this new task, we explore negative evidence in the structure of the taxonomy, as well as in external data sources such as the web. To integrate positive and negative evidence, we formulate the entity resolution problem as a problem of finding optimal multi-way cuts in a graph. We analyze the complexity of the problem, and propose a Monte Carlo algorithm for finding greedy cuts. We conduct extensive experiments that demonstrate the advantage of our approach.

1. INTRODUCTION

One of the tasks in data cleaning and data integration is entity resolution. Recently, entity resolution arises as one of the biggest challenges in the area of ontology/taxonomy managing, and in particular, in ontology/taxonomy mapping.

As understanding data semantics becomes more and more important in many applications, researchers and industrial practitioners create all kinds of ontologies and taxonomies to manage semantics. Each taxonomy has its own scope and characteristics, and many also have significant overlap. It is important to create mappings between multiple taxonomies for two reasons. First, mappings improve understanding, even if the two taxonomies are in the same domain. Second, taxonomy construction is a costly process, and it saves time and money if an existing taxonomy can be used to enrich a new taxonomy, and vice versa.

We focus on two web-scale taxonomies, namely Freebase [8] and Probase [35], to study this problem. From their major features listed in Table 1, we can see they are unique in their own ways.

<table>
<thead>
<tr>
<th></th>
<th>Freebase</th>
<th>Probase</th>
</tr>
</thead>
<tbody>
<tr>
<td>how is it built?</td>
<td>manual</td>
<td>automatic</td>
</tr>
<tr>
<td>data model</td>
<td>deterministic</td>
<td>probabilistic</td>
</tr>
<tr>
<td>taxonomy topology</td>
<td>mostly tree</td>
<td>DAG</td>
</tr>
<tr>
<td># of categories</td>
<td>12.7 thousand</td>
<td>2 million</td>
</tr>
<tr>
<td># of entities</td>
<td>13 million</td>
<td>16 million</td>
</tr>
<tr>
<td>information about entity</td>
<td>rich</td>
<td>sparse</td>
</tr>
<tr>
<td>adoption</td>
<td>widely used</td>
<td>new</td>
</tr>
</tbody>
</table>

Table 1: Two unique taxonomies.

On the other hand, Freebase has richer information about many entities. For example, for someone like Barack Obama, Freebase has the date of his birth, names of his spouse and kids, information about his religion, political party, etc. Although Probase contains a list of attributes for each category (i.e., Probase knows birthday is an attribute of a person), there are not many attribute values. Thus, a big motivation for matching the two taxonomies is to enrich the content of Probase at the entity level, and to enrich the content of Freebase at the category level.

However, matching two web-scale taxonomies such as Freebase and Probase pose at least two big challenges:

- Scalability. Probase has 16 million entities and Freebase has 13 million entities. It is infeasible to perform pairwise comparison for entity resolution.
- Insufficient context for entity resolution. Since Probase lacks attribute values for many entities, we cannot decide whether “J. Doe” is the same person as “John Doe” by comparing their birthdays.

Overcoming the above two challenges is essential to effectively creating a mapping between two web-scale taxonomies. To address the first problem, instead of comparing every pair of entities, we first look at the categories they belong to in each taxonomy. The intuition is that we probably do not need to compare entities in the category of “rare plants” with entities in the category of “an-
imals.” However, we might need to associate the category of “en-
dangered species” with the category of “animals.” The challenge is
that among the 12.7 thousand × 2 million pairs of categories
between Freebase and Probase, how do we figure out which pairs
need our attention?

To address the second problem of not having enough contextual
information for entity resolution, we collect relational evidence,
which is evidence derived from relationships among entities, rather
than from entities’ content. Specifically, we collect two types of
relational evidence. The first type is positive evidence. Lexical
similarity is a source of non-relational positive evidence. However,
lexical similarity is not sufficient, as for example, “New York City”
and “the Big Apple” have no lexical similarity, but they refer to
the same entity. In this paper, we collect positive evidence by fo-
cusing on entities’ relationships manifested in data sources such as
Wikipedia or the world wide web.

Yet a more important type of relational evidence is negative ev-
idence. If we have a sentence that says “... presidents such as
Wikipedia or the world wide web.

In this paper, we collect positive evidence by fo-
cusing on entities’ relationships manifested in data sources such as
Wikipedia or the world wide web.

Finally, given both positive and negative evidence, we are facing
the task of consolidating them to achieve entity resolution of high
accuracy. For instance, knowing that x is similar to y, and y is
similar to z (positive evidence), and also knowing that x is not z
(negative evidence), how do we perform entity resolution for x, y,
and z? In this paper, we formulate the problem of entity resolution
with positive/negative evidence as a multi-way graph cut problem,
and we propose a greedy approach to effectively solve the problem.

The rest of the paper is organized as follows. Section 2 describes
the taxonomies we are working with. In Section 3, we introduce
the types of evidence we use for entity resolution. Section 4 presents
our method of entity resolution and taxonomy matching. In Section
5, we report experimental results, and we conclude in Section 6.

2. TAXONOMIES

A taxonomy or an ontology provides a shared conceptualization
of a domain. Recently, there is a lot of interest in using structured
data to empower search or other applications. A general purpose
taxonomy about worldly facts is indispensable in understanding the
user intent, and much efforts are being devoted to composing and
managing such taxonomies.

Freebase [8] is a taxonomy composed mainly by its commu-
nity members. It is an online collection of structured data har-
vested from many sources, including individual ‘wiki’ contribu-
tions. Freebase aims to create a global resource which allows peo-
ple (and machines) to access common information more effectively.

Compared with manually constructed taxonomies, taxonomies
automatically generated from data have advantages in scale and
costs. Probase [35] is a research prototype that aims at building
a unified taxonomy of worldly facts from web data and search log
data. Compared with Freebase, the Probase taxonomy is extremely
rich. The core taxonomy alone (which is learned from 1.68 billion
web pages and 2 years’ worth of Microsoft Bing’s search log) con-
tains more than 2 million categories, while Freebase contains about
12.7 thousand categories. As categories in Probase correspond to
concepts in our mental world, Probase is valuable to a wide range
of applications, such as search [32], where there is a need to inter-
pret users’ intent.

Probase contains many isA relationships that are harvested using
the so called Hearst linguistic patterns [24], that is, SUCH AS like
patterns. For example, a sentence that contains “... artists such as
Pablo Picasso ...” can be considered as an evidence for the claim
that Pablo Picasso is an instance in the artist category. For each cat-
ogy, Probase also collects a large set of attributes that can be used to
describe instances in the category. For instance, the artist cate-
gory may contain such attributes as name, age, nationality, genre,
specialization, etc. Furthermore, Probase contains many relation-
ships among instances of different categories. Fig. 3 (in Appendix)
shows an interface with which users can browse the Probase tax-
onomy, and we can see a category (politicians) has many super
categories, sub categories, instances, and similar categories.

It is not difficult to see that there is a strong need to integrate a
taxonomy like Freebase with a taxonomy like Probase. With the in-
tegration, Freebase will have more information about categories, al-
lowing Freebase to understand human concepts better, and Probase
will have more information for each instance, giving Probase more
knowledge in inference. In this paper, we study the challenges in
creating a mapping between such taxonomies under this setting.

3. RELATIONAL EVIDENCE

In this section, we discuss how to obtain and quantify relational
evidence, and in Section 4 we discuss how to use the evidence for
entity resolution. Previous work assumes that each entity comes in
a context, for example, a text window where an entity appears, or
a set of attributes (such as a person’s gender or birthday) that de-
scribe the entity. Such context information is then used as positive
or negative evidence. However, the context may be insufficient or
noisy (e.g., the text window around an entity may contain irrelevant
information that confuses entity resolution). In our work, we focus
on a large number of entities that come with little context. For in-
stance, assume a list contains nothing but three names George W.
Bush, George H. W. Bush, and Dubya, how to find out how many
distinct entities it contains? In this section, we explore evidence
beyond entities’ immediate context to solve this problem.

3.1 Negative Evidence

Let \( (x_i, y_i) \) denote the claim that \( x_i \) and \( y_i \) represent the same
entity. Any evidence that supports the claim \( (x_i, y_i) \) is called pos-
tive evidence, and any evidence that rejects the claim \( (x_i, y_i) \) is
called negative evidence.

Negative evidence is particularly important in entity resolution.
For instance, string similarity may provide strong evidence that
George W. Bush is likely President Bush, and President Bush is
likely George H. W. Bush. Intuitively, we may conclude that George
W. Bush is George H. W. Bush, unless there is negative evidence to
break the transitivity.

The challenge is then, how to find negative evidence? Previous
work is based on data content. For example, if two persons with
similar names have different birthdays, then we can conclude that
they cannot be the same individual, unless the data is wrong. How-
ever, in many cases (e.g., taxonomies such as Probase), we do not
have sufficient content information for all entities.

We argue that although individual data items may not contain
much content, the community formed by the related items may con-
tain valuable clue to entity resolution. In our work, we derive neg-
ative evidence based on how the data is inter-connected internally
(e.g., in the taxonomies we are studying) as well as externally (e.g.,
on the web).

The ‘Birds of a Feather’ Principle (BoF)

Here is our intuition: Michael Jordan the professor and Michael
Jordan the basketball player may not have too many friends in com-
mon. In other words, unless two persons with similar names have many common friends (i.e., their friends also have similar names), it is not likely that the two names refer to the same individual.

In a taxonomy (such as Probase), a data element may not contain much content, but the connections among the elements can be extremely rich. We use the connections, or the structure formed by the data, as a most important source of evidence. Specifically, we consider two types of connections: i) the connection between elements and the category they belong to (e.g., [Michael Jordan, Shaquille O’Neal] and basketball players); and ii) the connection between attributes and the category they describe (e.g., {genre, artist, producer} and album). In a modern taxonomy such as Probase, categories, elements, and attributes are all objects, and they are interconnected into a graph. Thus, a category can be regarded as a graph community formed by elements and attributes. We derive important negative evidence from such communities:

**Negative evidence from the community:** Let $t$ be a threshold, and $c_1, c_2$ be two categories. We consider $\text{sim}(c_1, c_2) \leq t$ as negative evidence for any claim $(x, y)$ where $x \in c_1$ and $y \in c_2$.

Here, $\text{sim}(c_1, c_2)$ measures the similarity between two categories. We define $\text{sim}(c_1, c_2)$ by linear combination as follows:

$$\text{sim}(c_1, c_2) = \lambda \cdot f(E_{c_1}, E_{c_2}) + (1 - \lambda) \cdot f(A_{c_1}, A_{c_2})$$

where $E_c$ and $A_c$ denote the set of elements and attributes in category $c$ respectively, $\lambda$ is a parameter that balances the importance between elements and attributes, and $f$ is a set similarity function based on Jaccard distance.

The ‘Clean data has no duplicates’ Principle (CnD)

If a list is well formed, then it probably does not contain any duplicates. Freebase is manually created and maintained. Thus, we can be relatively certain that each element in a Freebase category is a unique element in that category. This gives us negative evidence for entity resolution. For example, given that George W. Bush and George H. W. Bush both appear in the category of US Presidents, we can conclude that the two very similar names cannot be referring to the same individual.

Probase, on the other hand, is automatically created and maintained. Thus, a category may contain duplicates. For example, Bill Clinton and William J. Clinton may both appear in the category of US Presidents. Still, we can derive negative evidence from Probase. As we mentioned, Probase derives the isA relationship from Hearst patterns. For example, from the sentence “US Presidents such as George W. Bush, Bill Clinton, George H. W. Bush,” Probase concludes that George W. Bush, Bill Clinton, George H. W. Bush are US Presidents. But given that the three names appear in the same sentence, we can conclude that the three names represent three individuals. In other words, there is negative evidence for the claim (George H. W. Bush, George W. Bush), but there is likely no negative evidence for (Bill Clinton, William J. Clinton).

Besides Freebase and Probase, we also derive negative evidence from the web using the same argument. One source of evidence comes from Wikipedia. Wikipedia contains many lists such as a list of mountains. Furthermore, the lists are well formed and easily identifiable: they all have a title in the form of “list of ***.” We extract entity names from the list, and assume mentions in the list will represent different entities. Furthermore, Wikipedia has structured tables. Usually, a table has an entity column and multiple attribute columns. In the entity column, we can get entity names and assume there are no duplicates. In theory, we can apply the same reasoning to data on the web. However, web data is often very noisy, which reduces the quality of the negative evidence.

### 3.2 Positive Evidence

Each piece of positive evidence is associated with a weight in the range of $[0, 1]$, which indicates how strong the evidence is. For instance, string similarity can be used as a source of evidence. Table 2 gives some examples, where we use Jaccard similarity to measure how strong the evidence is.

<table>
<thead>
<tr>
<th>id</th>
<th>claim</th>
<th>evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Bill Clinton, President Clinton)</td>
<td>.33</td>
</tr>
<tr>
<td>2</td>
<td>(George W. Bush, Dubya)</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>(George W. Bush, George H. W. Bush)</td>
<td>.75</td>
</tr>
</tbody>
</table>

**Table 2: String similarity as positive evidence**

Although string similarity seems to work well in some general cases, its limitations are obvious: i) For claim 2, the evidence based on string similarity has 0 weight, yet we know George W. Bush is also known as Dubya; ii) For claim 3, George H. W. Bush and George W. Bush have strong string similarity, yet they are father and son, not a single person.

In our work, we obtain positive evidence to support the claim that George W. Bush and Dubya refer to the same person. To do this, we need to go beyond string similarity, and explore the relationships among the two instances directly in external sources. Specifically, we explore Wikipedia and the web for positive evidence. Wikipedia has been used for disambiguation in several recent works [13, 9]. In our work, we consider some special constructs used in Wikipedia.

- **Wikipedia Redirects:** Some Wikipedia pages do not have their own content, and accesses to such pages are redirected to other pages. We use $x_i \sim y_i$ to denote the redirection.
- **Wikipedia Internal Links:** Links to internal pages are expressed in shorthand by [[Title | Surface Name]] in Wikipedia, where Surface Name is the anchor text, and the page it links to is titled Title. Again, we denote it as $x_i \sim y_i$, where $x_i$ is the anchor text, and $y_i$ is the title.
- **Wikipedia Disambiguation Pages:** An ambiguous phrase may correspond to multiple Wikipedia pages, each representing a specific interpretation of the phrase. Wikipedia puts such pages together for each ambiguous phrase. We denote this as $x \sim y$, where $x$ is the ambiguous phrase, and $y$ is the title of any of the Wikipedia pages.
- **Besides Wikipedia,** we also explore positive evidence on the web. We select several patterns, including ‘$x$ also known as $y$’, ‘$x$, whose nickname is $y$’, etc., and construct a large set of $x \sim y$.

### 3.3 Quantifying positive evidence

The evidence that support a claim $(x, y)$ may come from multiple sources, and each source gives a score in the range of $[0, 1]$ as an indicator of the strength of the evidence.

Each source employs its own mechanism to score the evidence. For example, for string similarity, a useful measure is the Jaccard coefficient:

$$w(x, y) = \frac{|x \cap y|}{|x \cup y|},$$

where $x \cap y$ denote the set of common words in $x$ and $y$, and $x \cup y$ denote the union of words in $x$ and $y$. Alternatively, we can use Dice’s coefficient:

$$w(x, y) = \frac{2n_x}{n_x + n_y},$$

where $n_x$ is the number of character bigrams found in both strings, $n_x$ and $n_y$ are the number of bigrams in string $x$ and $y$ respectively.
Both the Jaccard and the Dice's coefficients may encounter some problems. Consider the highschool category that contain the following instances \{riverdale high school, riverfull high school\}. The two have high similarity (0.5 based on Jaccard, 0.9 based on Dice's coefficient) because they share a substring "high school." Unfortunately, this is a very common substring in the highschool category. To correct this problem, we use weighted Jaccard similarity, where the weight is defined by the inverse of term frequency.

Other sources may collect multiple pieces of evidence for a claim. For example, in Wikipedia links, there are 32,467 occurrences of united states ~ usa, and 122 occurrences of George W. Bush ~ George Bush. We can use the Sigmoid function to take into consideration the multiple occurrences.

\[
w(x, y) = \begin{cases} 
1 & t > 0 \\
\frac{1}{1 + e^{-t}} & t \leq 0
\end{cases}
\]

where \(t\) is the number of occurrences. Hence, if there is only a single piece of evidence backing up a claim, then the evidence has a weight of 0.5. When \(t\) becomes larger, the weight becomes closer to 1.

Finally, assume evidence from \(k\) sources return a vector of \(k\) scores, \((a_1, a_2, \ldots, a_k)\), where each score \(a_i\) is in the range of \([0, 1]\). Our goal is to map the \(k\) scores into one score in the range \([0, 1]\). We apply the noisy-or model.

\[
1 - (1 - a_1) \cdot (1 - a_2) \cdots (1 - a_k)
\]

Intuitively, the evidence for a claim has a score as long as one evidence source gives it a high score. For instance, though George W. Bush and Dubya share no lexical similarity, Wikipedia frequently suggests Dubya is his nickname (strong evidential similarity).

4. METHODS

This section discusses how we use the evidence for entity resolution.

4.1 Problem Definition

We formally abstract claim \((x, y)\) as graph connectivity, using graph \(G = (V, E)\) with a set of vertices \(V\) representing the union of entities from two taxonomies, and a set of edges \(E\) with weight quantifying positive evidence between two entities (Eq 3).

Given \(G\), our aim is to group entities into clusters, so that entities in the same cluster refer to the same real-life entity. Specifically, we want to find a cut \(C \subseteq V\) to specify which edges we want to cut from \(G\), such that each connected component in \(G'(V, E - C)\) corresponds to the same real-life entity. Among all possible cuts, our goal is to find \(C\) that are (a) best supported by positive evidence and (b) not rejected by negative evidence. More formally, our goal is to find \(G'\) satisfying the following criteria:

**DEFINITION 4.1** (POSITIVE EVIDENCE). \(G'(V, E - C)\) should maximize the positive evidence supporting the claim \(e \in E - C\) quantified as \(w(e)\). In other words, \(C\) should minimize the following objective function:

\[
\sum_{e \in C} w(e)
\]

**DEFINITION 4.2** (NEGATIVE EVIDENCE). A valid solution \(G'\) should disconnect \(x\) and \(y\) rejected by some piece of negative evidence \(N_c\).

This problem can be formalized as the multi-multiway cut problem [2]. In this problem, given \(G\) and \(k\) sets \(N_1, \ldots, N_k\) of vertices, the goal is to find a subset \(C\) of edges whose removal disconnects every \(x, y \in N_i\) for some \(i\) with minimal cost. This problem can be formulated as the following integer programming problem [2]:

\[
\begin{align*}
\text{minimize} & \quad \sum_{e \in E} w(e) x(e) \\
\text{subject to} & \quad \forall P, c, t \in \mathbb{P} : \sum_{e \in P} x(e) \geq 1 \\
& \quad \forall e \in E : x(e) \in [0, 1]
\end{align*}
\]

where \(\mathbb{P}\) denotes the set of all paths between \(u, v \in N_i\) for some \(i\). However, finding an exact solution requires ILP (integer linear programming) optimization populating binary decision variables \(x(e)\) for every entity pair, \(O(10^M)\) in our problem, with the constraints enumerating all possible paths between every entity pair mentioned in negative evidence and requiring their disconnection. Even the best known approximation algorithm [2] requires expensive LP optimization and \(P\) enumeration, and is still too expensive for our target problem. In addition, this solution has \(O(\log k)\) approximation ratio, which is also not desirable for our target problem of large \(k\).

To build a scalable approximation algorithm, we revisit the two principles discussed in Section 3, which we will describe in detail in the following two sections.

- Using the ‘Clean data has no duplicates (CnD)’ principle: We use negative evidence from clean data to develop an approximate graph cut that is highly efficient and accurate.
- Using the ‘Birds of a Feather (BoF)’ principle: We localize our graph to compare only the entities in matching categories \(c_1\) and \(c_2\) with high \(sim(c_1, c_2) > t\), as the BoF principle rejects \((x, y)\), when \(x \in c_1, y \in c_2, sim(c_1, c_2) \leq t\).

4.2 Using the CnD Principle

As Section 3 discussed, negative evidence from some sources, e.g., Freebase created and maintained manually, does not contain any duplicates. Such cleanliness can be exploited for efficient graph cut computation. More formally, we define clean data as follows:
DEFINITION 4.3 (CLEAN EVIDENCE). k sets of negative evidence \( N_1, \ldots, N_k \) is clean, if and only if for \( x \in N_i \) and \( y \in N_j \), with different names cannot refer to the same real-life entity.

For example, \( N'_1 = \{\text{Bill Gates}, \text{Bill Clinton}\} \) and \( N'_2 = \{\text{William Gates}, \text{Bill Clinton}\} \) are not clean, as Bill and William Gates refer to the same entity. In clear contrast, \( N_1 = \{\text{Bill Gates}, \text{Bill Clinton}\} \) and \( N_2 = \{\text{Barack Obama}, \text{Bill Clinton}\} \) is a clean set.

For a clean set, our optimization problem can be reduced to a special case of multi-way cut with \( k = 1 \), known as the multi-way cut problem [14]. That is, we can aggregate \( k \) sets of negative evidence into a single set \( U, N_i \). For example, in the above clean set example, we can aggregate \( N_1 \) and \( N_2 \) into a single set \( U, N'_1 = \{\text{Bill Gates}, \text{Bill Clinton}, \text{Barack Obama}\} \), suggesting any claim \( \{x, y\} \) for \( x, y \in U, N'_1 \) should be rejected. The same aggregation would not work for the dirty set example, as Bill and William Gates in \( U, N'_2 = \{\text{Bill Gates}, \text{Bill Clinton}, \text{William Gates}\} \) refer to the same person and should not be rejected.

The multiway cut problem is proved to be NP-hard when \( m = |U_i, N_i| \geq 3 \), and an efficient “isolation heuristic” gives an approximation ratio of \( 2 - \frac{1}{m} \) [14]. In this paper, we discuss this heuristic, though there is an approximation algorithm with possibly better ratios, e.g., 1.348 [25], as all other existing solutions do not scale for our graph, requiring LP optimization.

The isolation heuristic works as follows. For any “terminal” \( t_j \in U_i, N_i \), we will find its minimum weight “isolating cut”, which is a subset of edges, whose removal disconnects \( t_j \) from all other terminals. Such a cut can be found by connecting all other terminals in a shared new vertex \( v \) with infinite weight and running the maximum flow algorithm from \( t_j \) to \( v \). Once we find the isolating cut for each terminal node, we can union the sets to get an approximate answer. A more detailed description is available in Algorithm 1.

Algorithm 1 Isolation heuristics

Require: \( G = (V, E) \), terminals \( U_i, N_i = \{t_1, \ldots, t_m\} \)
Output \( E_1 \cup \ldots \cup E_{m-1} \), assuming \( w(E_1) \geq \ldots \geq w(E_{m-1}) \) (without loss of generality)

In our problem, as we collect negative evidence that are both clean and dirty, we divide them into two groups of sets \( N \) and \( N' \), i.e., \( N \cup N' = U_i, N_i \) and take a two-phase approach: First, we run the isolation heuristics for clean set \( N \). Once we get connected components, we investigate each \( N_i \in N' \), to identify violated entity pairs \( s, t \in N_i \), which belong to the same connected component. Once we collect all such pairs, we can find a minimum weight cut using \( s \) as a source node and \( t \) as a sink, known as \( s-t \) cut [18], separating two pairs and thus eliminating a violation. \( s-t \) cut can be computed in polynomial time for every violated pair.

For faster computation, we can consider a greedy approximation of isolation heuristics. Specifically, we start from \( G \) with singleton clusters, i.e., \( G' = (V, \phi) \), where all terminals are trivially isolated. We can then greedily add an isolating cut with the highest weight first.

As the problem is NP-hard, this greedy heuristic finds a suboptimal solution, as illustrated in Fig 1 with two pieces of negative evidence \( N_1 = \{a, c\} \) and \( N_2 = \{b, d\} \). Starting from singleton clusters, we iteratively add edges in decreasing order of weights. We will choose to insert edges \( (a, b), (c, d), (a, e), (c, f) \). After these four inserts, the next candidate would be \( (a, f) \), which is not an isolating cut by violating \( N_1 \) and generating a path from \( a \) to \( c \). Similarly, \( (b, f) \) is not an isolating cut by generating a path from \( b \) to \( d \). We thus skip these two candidates and continue to add \( (d, f) \). This insert terminates the search, as we cannot add any more edges without violating negative evidence. This solution with a cost of \( 0.4 + 0.3 + 0.6 + 0.6 = 1.9 \) is sub-optimal, as adding \( (a, f) \) and \( (b, f) \) instead of \( (c, f) \) and \( (d, f) \) would lower the cost to \( 0.4 + 0.3 + 0.1 + 0.7 = 1.4 \).

We thus implement a Monte-carlo approach in Algorithm 4.2 to randomize edge insertion with probability proportional to the cost from the “randomized candidate list” \( RCL \). Furthermore, this procedure can be repeated several times and the lowest cost cut can be identified as a solution.

Algorithm 2 Monte-carlo heuristics

Require: \( G = (V, E), N = N_1, \ldots, N_M \)
Output \( E' = \phi, RCL = E \)
while \( RCL \) is not empty do
Randomly select \( e \) from \( RCL \) with probability proportional to its weight
\( RCL = RCL \setminus \{e\} \)
if \( e \) is an isolating cut then
\( E' = E' \cup \{e\} \)
end if
end while
return \( (V, E') \)

After a graph cut, each connected component corresponds to one real-life entity. As each Freebase entity corresponds to a unique real-life entity, each component has at most one Freebase entity, which we label with belief 1.0 and then propagate to unlabeled Probase entities using Random Walk with Restart to calculate the label probability of each node in the subgraph. Once the propagation converges, the scores can be used to prune out Probase entities with low scores.

4.3 Using the BoF Principle

As computing cuts is expensive, we reduce the given graph such that entities in \( E \) belong to the matching categories. That is, for our aim of entity resolution, a book entity cannot refer to a person. We can thus use ontological information, representing which category the entity belongs to, in order to significantly reduce the graph size.

To decide whether two categories \( c_1, c_2 \) are a match, we discussed \( \text{sim}(c_1, c_2) \) in Eq. 1, combining the similarity between element sets \( E_c \) and attribute set \( A_c \) of the two categories. More specifically:

- \( f(E_{c_1}, E_{c_2}) \): Most existing ontology integration work quantifies a set similarity between \( E_{c_1} \) and \( E_{c_2} \), e.g., using Jaccard similarity or its variants as used in [31].
- \( f(A_{c_1}, A_{c_2}) \): Alternatively, one can compare \( A_{c_1} \) and \( A_{c_2} \). From Freebase, each category is associated with a single relational table, from which we can obtain a set of attributes. However, for Probase, we can collect many table instances describing entities of the given category. We thus use vector space model used for representing text documents, to represent each category as a frequency vector of a universe of attribute names used, normalized by the number of table instances. For instance, a Probase class \( \text{album} \) is frequently represented by attributes \{\text{genre, artist, producer}\}, represented by attribute frequency vector \( \{0.9235, 0.8431, 0.8301\} \). Meanwhile, attribute frequency vector for the Freebase category will have binary values. The similarity \( f(A_{c_1}, A_{c_2}) \) is generally computed using cosine similarity or KL-divergence.
Using $A_x$ complements $E_x$ in the following two aspects: First, when using $E_x$ similarity alone, two entities with the same name cannot be distinguished, such as *electronics*, which is both an industry and a genre. When comparing two categories of small size, such a false match may lead to the overestimation of the concept similarity—In our dataset, two unrelated categories /broadcast/genre and /manufacturing/industry have sizes 98 and 108 and have a few false matches, such as *electronics* and *automotive*, which gives a high Jaccard similarity score 0.0147. Meanwhile, if considering $A_x$ similarity as well, we can distinguish *electronics* the industry and the genre, as attributes describing the two would be different. Second, the size of $E_x$ varies by orders of magnitude over categories and sources, e.g., two identical categories albums (from Probase) /music/album (from Freebase) have 1900 and 526,038 entities respectively. A severe imbalance in the size of $E_x$, though we attempt normalization, negatively affects the reliability of metrics, while $A_x$ sizes are more balanced.

Accurate computation of both metrics requires to resolve entities and attributes, as treating the entities Bill Gates and William Gates, or the attributes author and writer, as unmatched would underestimate the score. However, as we are computing these metrics for the goal of entity resolution, it is unrealistic to assume entities and attributes are resolved *a priori*. We thus consider only exact matches for both entity and attribute similarity, as a lower bounding estimate, to reduce the graph size. Once we compute graph cuts, we can apply our finding, e.g., Bill Gates ~ William Gates, to recompute $f(E_{x1}, E_{x2})$ so as to refine the score into a tighter lower bound. From this refinement, we can identify some unrelated categories, which were identified as matching, and we can drop entity resolution results obtained from such category pairs. We leave detailed discussions on the details of how we identify similarity threshold $t$ and compute matching pairs efficiently in Appendix C.

5. EXPERIMENTS

In this section, we evaluate our approach for entity resolution. Please refer to Appendix D for a description of the system, including its flowchart, and the setting of the experiments.

5.1 Evidence

We extract positive and negative evidence from Wikipedia and Freebase. Table 3 and Table 4 show the numbers of different types of evidence and their sources. Note that we also use Freebase as a source of negative evidence, as Freebase is handcrafted, so it satisfies the ‘Clean data has no duplicates’ principle.

<table>
<thead>
<tr>
<th>Wiki Source</th>
<th># of Pairs</th>
<th>Source</th>
<th># of Bags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Links</td>
<td>12,662,226</td>
<td>Wikipedia List</td>
<td>122,615</td>
</tr>
<tr>
<td>Redirect</td>
<td>4,260,412</td>
<td>Wikipedia Table</td>
<td>102,731</td>
</tr>
<tr>
<td>Disambiguation</td>
<td>223,171</td>
<td>Freebase</td>
<td>12,719</td>
</tr>
</tbody>
</table>

Table 3: Positive Evidence

<table>
<thead>
<tr>
<th>Source</th>
<th># of Bags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freebase</td>
<td>12,719</td>
</tr>
</tbody>
</table>

Table 4: Negative Evidence

5.2 Instance Mapping

We match Freebase and Probase categories based on similarity in their attributes and elements. We find 763,000 matchings. Here, we evaluate some selected pairs, including: i) Probase politician and Freebase /government/politician; ii) Probase format and Freebase /computer/file/format; iii) Probase system and Freebase /computer/operating_system; iv) Probase airline and Freebase /aviation/airline. More results can be found in the Appendix (Fig. 5).

We measure precision, recall and output size for each case. Precision and recall are well defined. However, in large-scale instance mapping, sometimes it is not easy to obtain the exact recall. We use output size [1] as an additional factor in our evaluation. When exact recall is not available, the output size can also give us a feeling of how effective the method is in finding all qualified matches.

We manually label instance pairs as ‘Match’ or ‘Non-match.’ We cannot simply use extracted positive evidence for automatic precision evaluation, because, first, it may contain noise; second, it may not contain all variations of possible mappings. Specifically, for a matching category pair, we randomly sample some Freebase entities, then label all Probase entities that map to the Freebase entities as ‘Match,’ and the rest as ‘Non-match.’ Currently, our algorithm maps each Probase entity to one Freebase entity. In our evaluation, for ambiguous instances such as ‘Bush,’ we consider both mappings to ‘George W. Bush’ and ‘George H. W. Bush’ correct.

Table 5 shows the result for the 4 selected category pairs. We labeled 214, 134, 76, 201 Freebase entities for each case, and obtained 1,687 positive mapping pairs (408, 406, 384 and 489 for each category pair, respectively). For our method, we used thresholds 0, 0.05 and 0.10. We compare our method with two baseline algorithms. Let $(X, Y)$ be a category pair. Baseline #1 maps $x \in X$ to $y \in Y$ if $(x, y)$ has the strongest positive evidence (e.g., it has the largest number of $x \sim y$ instances). Baseline #2 maps $x \in X$ to $y \in Y$ if $(x, y)$ has the biggest string similarity as measured by Jaro-Winkler [34], and the similarity is larger than a threshold of .9, as smaller thresholds produce too many false positives.

For popular and stable categories such as *politician* or *airline company*, we have more high quality positive evidence. Therefore, Baseline #1 shows high precision for the pair ‘politician’ and ‘government/politician,’ because it is based on strong positive evidence. However, Baseline #1 does not take string similarity into consideration. As there are mispelled instances, and variations in surface forms, the recall and the output size of Baseline #1 are smaller than ours. Although Baseline #2 is good for matching names with misspellings, it gives relatively low score to pairs such as (‘Barack Obama’, ‘Obama’), and produces many false positives such as (‘George H. W. Bush’, ‘George W. Bush’), (‘Hillary Clinton’, ‘Bill Clinton’). As a high threshold of .9 is used in Baseline #2 to boost precision, both the recall and the output size of Baseline #2 are smaller than ours, or even Baseline #1’s. On the other hand, the recall of our methods is higher than the baseline methods while the precision is still competitive.

For relatively new and less well-defined categories such as *file format* or *operating system*, using only positive evidence extracted from Wikipedia is not enough for entity resolution. Therefore, both the precision and recall of Baseline #1 is low. Meanwhile, Baseline #2 produced large amount of false positives because the string length is short (for *file format*), or the discriminative part in the string is small (e.g., ‘Windows 95’ and ‘Windows 7’). Our method outperforms both Baseline #1 and #2 in precision, recall, and output size.

In Table 9 (see Appendix), we show a list of Probase entities that are mapped to Freebase instances such as ‘Barack Obama,’ ‘American Airlines,’ and ‘XLS.’ Take the mapping between ‘us president barack obama’ and ‘barack obama’ for example. Baseline #1 failed because it lacks positives evidence from Wikipedia saying ‘us president barack obama’ is ‘barack obama,’ and Baseline #2 failed because the similarity between the two is less than the threshold of .9. Our method worked because the positive evidence from string similarity was not disrupted by any negative evidence. As another example, for ‘American Airlines,’ Wikipedia Disambiguation page provides a wrong piece of positive evidence: ‘Hawaiian airlines’ ~ ‘American Airlines.’ Baseline #1 failed because of this, but our method was able to overcome the noisy evidence with negative ev-
Table 5: Precision and Recall for Selected Category Pairs. (P.: Precision, R.: Recall, S.: Output Size)

<table>
<thead>
<tr>
<th>Probase Class Name</th>
<th>politicians/formats/systems/airline</th>
<th>Probase Type ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>politicians/politician/computer/file_format/computer/operating_system/aviation/airline</td>
<td>Freebase Type ID</td>
</tr>
<tr>
<td>Method</td>
<td>P.</td>
<td>R.</td>
</tr>
<tr>
<td>Baseline #1</td>
<td>0.9952</td>
<td>0.6603</td>
</tr>
<tr>
<td>Baseline #2 (0.90)</td>
<td>0.9792</td>
<td>0.3110</td>
</tr>
<tr>
<td>Our method (0.00)</td>
<td>0.9815</td>
<td>0.8413</td>
</tr>
<tr>
<td>Our method (0.05)</td>
<td>0.9851</td>
<td>0.8413</td>
</tr>
<tr>
<td>Our method (0.10)</td>
<td>0.9924</td>
<td>0.8254</td>
</tr>
</tbody>
</table>

5.3 Finding Candidate Category Pairs

We demonstrate how Eq 1 implements the BoF principle in finding candidate category pairs. First, Table 6 shows that Jaccard similarity alone does not work well for two reasons: i) The two sets have a big difference in size; and ii) Instances have ambiguous names. More specifically,

- A large Jaccard similarity threshold will exclude related pairs such as (Written Work, Novels) because although the categories are heavily related, they may not have enough similarity due to a large size difference.
- A small Jaccard similarity threshold will fail to exclude false positives such as (Places, Written Work). This is because there are instances of Written Work, Musical Album titled China, Canada, etc.

Second, as the first 4 rows in Table 6 show, attribute similarity fares better than entity similarity in distinguishing ambiguous category pairs, allowing us to obtain (Novels, Written Work) and (Albums, Musical Albums), and reject (Places, Written Work) and (Words, Musical Album). However, since Freebase has only a few attributes, using attributes similarity alone may also cause problems.

Overall, a linear combination of entity similarity and attribute similarity works well. Besides using similarity measures only, we can also use the hierarchy of Probase and a few handcrafted rules to improve the precision and recall in matching.

5.4 Scalability

In our work, we integrate positive and negative evidence by finding a multiway cut in a connected graph. A common approach to find a multiway cut is to use integer linear programming (ILP). To estimate the cost of doing this for integrating web scale taxonomies, we select three categories (of typical size) from Probase, connect entities in each category by positive evidence, and find the largest connected subgraphs. Table 8 shows that typically the largest connected subgraph contains almost half of the nodes in the original graph. Since we need to apply ILP on each connected subgraph, the sheer size of the subgraph makes the ILP approach computationally infeasible [2]. This justifies the use of a Monte Carlo approach to solve our problem.

Table 8: Size/ratio of the largest connected subgraph. $V_1$ is the set of nodes in the largest connected subgraph of $G = (V, E)$

| Probase category | $|V_1|$ | $|V|$ | $|V_1|/|V|$ |
|------------------|--------|-------|--------------|
| song             | 9932   | 21963 | 0.4522       |
| artist           | 18839  | 39648 | 0.4788       |
| disease          | 1435   | 4480  | 0.3203       |

6. CONCLUSION

Entity resolution for data integration is a challenging task. In this paper, we study the problem of matching millions of entities in two web scale taxonomies. Unlike integrating two relational tables, taxonomies may not contain much information about each entity. But it is exactly this reason that makes the task of integrating two taxonomies important, as integration serves as an indispensable mechanism for taxonomies to enrich themselves by “borrow-
ing” content from other taxonomies. We develop a framework that relies on the interconnections of the data in the taxonomies as well as in external data sources for entity resolution. We collect a large number of positive and negative evidence from the interconnections, and formulate the task of entity resolution as a multi-way graph cut problem. Our experiments show that our method scales up to millions of categories and entities, and produces very high quality resolutions.

7. REFERENCES


| Freebase Type | Probase Class | \(|E_{c_1} \cap E_{c_2}|\) | \(f(E_{c_1} \cap E_{c_2})\) | \(f(A_{c_1}, A_{c_2})\) | \(\text{sim} (c_1, c_2)\) |
|---------------|---------------|-----------------|-----------------|-----------------|-----------------|
| Written Work  | Novels        | 755             | 0.0004          | 0.0275          | 0.0220          |
|               | Places        | 4902            | 0.0023          | 0.0002          | 0.0006          |
| Musical Album | Albums        | 1095            | 0.0026          | 0.0638          | 0.0516          |
|               | Words         | 1550            | 0.0036          | 0.0007          | 0.0013          |
| Breed Origin  | Country       | 82              | 0.0826          | 0.0000          | 0.0165          |
| Musical Instrument | Percussion | 95              | 0.0565          | 0.0035          | 0.0141          |
|               |               | \(\sum |C_1 \times C_2|\) | 24,377,292,299  | 23,323,213,795  | 24,923,006,886  |
|               |               |                 | Task Set 1      | Task Set 2      | Task Set 3      | Task Set 4      |
APPENDIX

A. RELATED WORK

Entity resolution, also known as record linkage or reference reconciliation, is an important and difficult problem. Some researchers try to solve this problem by finding the best string similarity measures [26, 12]. But string similarity measures have limitations: They are not able to identify nicknames, or distinguish entities whose names are very similar, or handle noises in names. Recently, researchers focused on adaptive algorithms that learn similarity measures automatically [7, 30]. However, the difficulty of using these methods cannot be overlooked: It requires training data for each domain. To address this problem, some approaches use active learning so that only a little user input is needed [27]. But, still no methods can effectively handle the case where an entity can have totally different names, for example, ‘The Governor’ as a nickname for ‘Arnold Schwarzenegger.’ Also, labeling a small number of examples for each domain is still costly in our scenario, since the taxonomies we are dealing with contain millions of categories (domains).

Entity resolution in specific domains is challenging as well. To match person names [20, 11, 21], a variety of rules, such as abbreviation, omission, transposition (or sequence changing), are developed to transform a person’s name before comparing it with other names [21]. Profilers that have domain knowledge (for domains such as movies, reviews, and people) are used for various object matching tasks [16, 15]. More recently, Markov logic is used [28] to soften the hard constraints of handcrafted rules in first-order logic. Because these methods depend on much domain knowledge, they work well for specific domains only.

Entity resolution is also studied extensively in the field of data integration [19]. A database table usually has multiple columns and contains many values. In this setting, some attributes may be more important than others for measuring similarity for records. Arasu et al [1] proposed a method that learns threshold-based boolean functions or linear classifiers for entity resolution. Given two tables and a precision threshold, their method provides the largest output (a concept similar to recall defined in their paper) satisfying the precision threshold. There are many challenges. First, it is difficult to construct a good training set that includes both positive and negative training data. Second, the training sets are usually domain-dependent. Third, attributes in the two tables may require reconciliation as well. A possible approach is to regard attributes and their values as another type of classes and entities, and leverage relations among them to help solve the entity resolution problem [5, 6, 4].

Recent approaches also leverage knowledge acquired from external sources for domain-independent entity resolution. For example, some extracts entities’ surface forms (names or aliases) from Wikipedia and builds a synonym dictionary for entities [13, 9, 23]. Given a name, we first find Wikipedia articles corresponding to the name or to its synonyms, and then create a bag-of-words vector for the name, using the words from the Wikipedia articles. Finally, we compare two names using metrics such as the cosine similarity for entity resolution. This approach assumes entities have corresponding Wikipedia articles, but that covers a very small percentage of entities. One way to extend it is to consider transitive relation of entities. One way to extend it is to consider transitive relation of entities involving Wikipedia articles, but that covers a very small percentage of entity resolution. This approach assumes entities have corresponding Wikipedia articles, but that covers a very small percentage of entities. One way to extend it is to consider transitive relation of entities involving Wikipedia articles, but that covers a very small percentage of entity resolution. This approach assumes entities have corresponding Wikipedia articles, but that covers a very small percentage of entities. One way to extend it is to consider transitive relation of entities involving Wikipedia articles, but that covers a very small percentage of entity resolution. This approach assumes entities have corresponding Wikipedia articles, but that covers a very small percentage of entities.
### Table 9: Probase instances mapped to the Freebase entities.

<table>
<thead>
<tr>
<th>Freebase Entity</th>
<th>Baseline #1</th>
<th>Baseline #2 (0.90)</th>
<th>Our Method (0.10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barack Obama</td>
<td>barack obama, barrack obama, senator barack obama, president barack obama</td>
<td>barack obama, barrack obama</td>
<td>barack obama, barrack obama, senator barack obama, president barack obama, us president barack obama, mr obama</td>
</tr>
<tr>
<td>John Kerry</td>
<td>john kerry, senator john kerry, sen. john kerry, senator john kerry</td>
<td>john kerry</td>
<td>john kerry, senator john kerry, sen. john kerry, senator john kerry, massachusetts sen. john kerry, sen. john kerry of massachusetts</td>
</tr>
<tr>
<td>MP3</td>
<td>mp3, mp3s, mp3 files, mp3 format</td>
<td>mp3, mp3s</td>
<td>mp3, mp3s, mp3 files, mp3 format, high-quality mp3, mp3 songs</td>
</tr>
<tr>
<td>XLS</td>
<td>xlsx, microsoft excel</td>
<td>xlsx, xlsx</td>
<td>xlsx, microsoft excel, excel file, excel documents</td>
</tr>
<tr>
<td>Windows VISTA</td>
<td>windows vista, windows vista sp2, windows vista service pack 2</td>
<td>windows vista, windows vista sp2</td>
<td>windows vista, windows vista sp2, windows vista service pack 2, microsft windows vista</td>
</tr>
<tr>
<td>American Airlines</td>
<td>american airlines, american airline, aa, hawaiian airlines</td>
<td>american airlines, american airline</td>
<td>american airlines, american airline, aa</td>
</tr>
</tbody>
</table>

### D. EXPERIMENT SETTING

We conduct comprehensive experiments in integrating Probase and Freebase. We build a distributed entity resolution system (as Fig. 4 shows) that contains three servers running 64-bit Microsoft Windows Server 2003 Enterprise SP2 OS, with 16 core 2.53 GHz Intel Xeon E5540 processors and 32 GB of memory. Each of the servers run 10 instance mapping clients. All together, we run 30 instance mapping clients in parallel.

We used Freebase data dump for 2010-10-14 [22]. The total number of topical instances in Freebase is 13,491,742. Non-topical instances are instances that do not represent real world entities, for example, data types that are used internally in Freebase, and we do not include them in our experiments. The number of non-empty topical categories in Freebase is 12,719.

In terms of Probase, instances are classified into a very diverse set of categories. In the version of Probase we used for the experiments, the total number of instances is 16,423,710, the total number of categories is 2,026,806, and the total number of attributes is 1,727,580.

#### Figure 4: The Flowchart of the System
Table 10: Category Pair Name, and Its Probase Class Name and Freebase Type ID