Web Scale Taxonomy Cleansing

Taesung Lee [†,‡][*] Zhongyuan Wang [‡] Haixun Wang [‡] Seung-won Hwang [†]

[†]POSTECH
Korea, Republic of
{elca4u,swhwang}@postech.edu

[‡]Microsoft Research Asia
Beijing, China
{zhy.wang,haixunw}@microsoft.com

ABSTRACT

Large ontologies and taxonomies are automatically harvested from web-scale data. These taxonomies tend to be huge, noisy, and contains little context. As a result, cleansing and enriching those large-scale taxonomies becomes a great challenge. A natural way to enrich a taxonomy is to map the taxonomy to existing datasets that contain rich information. In this paper, we study the problem of matching two web scale taxonomies. Besides the scale of the problem, we address the challenge that the taxonomies may not contain enough context (such as attribute values). As existing entity resolution techniques are based directly or indirectly on attribute values as context, we must explore external evidence for entity resolution. Specifically, we explore positive and negative evidence in external data sources such as the web and in other taxonomies. 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 and compare our approach with three existing methods to demonstrate the advantage of our approach.

1. INTRODUCTION

One of the computational grand challenges for the 21st century is integrating, representing, and reasoning over human knowledge. For centuries, innumerable efforts have been devoted to manage human knowledge, and a large variety of domain-specific or domain-general ontologies and taxonomies have been constructed for this purpose.

Large scale modern ontologies and taxonomies are automatically harvested from a large amount of data, for example, the web corpus [27, 36, 11]. An automatically harvested taxonomy typically has the following characteristics:

- The taxonomy has a large number of categories. For example, Probase [36, 26] has 2.7 millions categories. In addition to popular categories such as *country* and *artist*, the taxonomy contains long tail concepts that are very concrete and specific, for example, *18th century European painters*, *coastal Chinese provinces*, and *basic watercolor techniques*. Each category contains a number of sub-categories or instances. The entire taxonomy can be considered a database of millions of tables.

- Since the taxonomy is automatically harvested using information extraction and machine learning techniques, inevitably it contains noises and inconsistencies. For example, the category of *US presidents* may contain the following instances:

  { $\cdots$, George H. W. Bush, George W. Bush, Dubya, President Bush, G. W. Bush Jr., $\cdots$ }

  Here, "George H. W. Bush" and "Dubya" refer to the same person, while "George W. Bush" and "George H. W. Bush" refer to two different persons.

- The raw taxonomy contains little context. Typically, each entity is only represented by a name. For example, for "George H. W. Bush" in the *US presidents* category, we have no information such as birthday, political party, religious belief, etc.

Cleaning and enriching a large-scale, automatically harvested taxonomy poses great challenges. Clearly, most cleaning requires entity resolution. To enrich the taxonomy, a natural approach is to map the taxonomy to all sorts of existing datasets that contain rich information about the entities. However, in order to perform such mapping, we must address the problem of entity resolution first.

Entity resolution relies on the *context* of the entity. For a person, such a context may include attribute-based information, such as his name, birthday, affiliation, etc., or relationship-based information, such as as co-authorship, etc. In summary, state-of-the-art entity resolution approaches can be classified into the following 3 categories [7].

- **Attribute-based entity resolution**: This approach uses a similarity measure over attributes, and two instances that have attribute similarity above a certain threshold are considered to be the same entity.

  Example: A and B have the same (or a very similar) name, affiliation, birthday $\rightarrow$ A and B are the same entity.

- **Naïve relational entity resolution**: In addition to the attributes of the two instances themselves, this approach also considers the attribute similarity of their related instances.

  Example: A and B have similar attributes, and A and B have co-authors with very similar attributes $\rightarrow$ A and B are the same entity.

[*]This work is done at Microsoft Research Asia. The first author's internship was partially supported by NRF (National Research Foundation of Korea)

- **Collective entity resolution**: Whether two instances map to the same entity or not also depends on whether their related instances map to the same entities.

  Example: A and B have similar attributes, and A and B have co-authors with very similar attributes, and A and B's coauthors map to the same entity $\rightarrow$ A and B are the same entity.

Clearly, all of the above approaches, including collective entity resolution, are directly or indirectly based on attribute similarity. This creates a big challenge for the taxonomy mapping application. A raw, automatically harvested taxonomy does not have much context for its instances or concepts. Each instance only has a name, and entity resolution based on names only is clearly error-prone.

Existing large-scale ontologies [27, 11] address this problem, by using natural language text appearing around the entity name as contextual information for entity resolution. However, such context is inherently noisy, which requires subsequent automatic reasoning to eliminate inconsistency. Such reasoning is computationally intensive and also highly conservative, which limits the scalability and recall of these methods.

In contrast, our goal is to tackle the dual challenges of the lack of context, and the scalability to web-scale ontologies. The second challenge is especially important for Probase, containing 16 million instances of unique names. Suppose we want to map its instances to Freebase, which contains 13 million entities. It is infeasible to perform 16 million $\times$ 13 million pairwise entity resolution.

Overview of Our Approach

Overcoming the two challenges mentioned above is essential to effectively cleaning and enriching a web scale taxonomy. To address the challenge of not having enough contextual information for entity resolution, we collect probabilistic relational evidence from external sources, and use this external evidence for entity resolution.

Specifically, we collect two types of evidence. The first type of evidence is positive evidence. For example, suppose in the category of *US cities* there are two instances, "New York City" and "the Big Apple." Since the two names have no lexical similarity, it is impossible to come to the conclusion that they actually refer to the same entity unless we find evidence elsewhere. There are many external sources that may provide such evidence. For example, we may find a sentence that says "... New York City, also known as the Big Apple ..." or we may find that "the Big Apple" is listed as one of the synonyms for "New York City" in Wikipedia. One question is, how do we know "the Big Apple" in the external source refers to "the Big Apple" in the taxonomy? The answer is that we do not know for sure. However, we are not relying on a single instance. The fact that the same set of instances co-occurs in the taxonomy as well as in the external sources reinforces the mappings between the corresponding pairs. In our approach, positive evidence is probabilistic, and the associated probability indicates the strength or the credibility of the evidence.

We rely on positive evidence from external sources because our data itself does not have much context information. However, using positive evidence only is very risky, particularly when the external sources (e.g., the world wide web) are not always reliable and information extraction methods have errors. Without counter-evidence, we may conclude that two instances refer to the same entities if they show some insignificant similarity.

To solve this problem, we focus on a more important type of external evidence, which is negative evidence. If we find a sentence that says "... presidents such as *George W. Bush*, Bill Clinton, and *George H. W. Bush*", then we may conclude that *George W. Bush* and *George H. W. Bush* refer to two different persons, because a well formed list probably contains no duplicates. The idea of using negative evidence is not new. For example, when resolving references of co-authors of a paper, we know that each name in the co-author list refers to a unique person. However, in our work, we go beyond the data we have to explore negative evidence in external sources. We show that such kind of negative evidence can be found in many external sources, for example, in tables or lists on the web, or in a very well organized taxonomy (such as Freebase).

Finally, given both positive and negative evidence, we face 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.

Besides the challenge of lacking context for entity resolution, we also have the scalability challenge. Consider mapping Probase, an automatically harvested taxonomy, to Freebase, a community built taxonomy. 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.

Since Probase contains 16 million entities in more than 2 million categories, and Freebase contains 13 million entities in 12.7 thousand categories, pairwise entity resolution is infeasible. 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 "animals." However, we might need to associate the category of "endangered species" with the category of "animals." The challenge is that among the $12.7\ thousand \times 2\ million$ pairs of categories between Freebase and Probase, how do we figure out which pairs need our attention?

Paper Organization

The rest of the paper is organized as follows. In Section 2, we discuss related work. Section 3 describes the taxonomies we are working with. In Section 4, we introduce the types of evidence we use for entity resolution. Section 5 presents our method of entity resolution and taxonomy matching. In Section 6, we report experimental results, and we conclude in Section 7.

2. RELATED WORK

Entity resolution, or reference reconciliation, is a challenging problem in data integration. Naive methods for entity resolution are based on string similarity [25, 13]. Machine learning approaches are introduced to combine domain knowledge with string similarity measures [8, 29]. String similarity based methods cannot handle the case where an entity can have totally different names, for example, 'The Governator' as a nickname for 'Arnold Schwarzenegger.' Furthermore, supervised methods are costly in our scenario, since the taxonomies we are dealing with contain millions of categories (domains), which means we need millions of training data sets.

Entity resolution is critical for integrating relational data [19]. A database table usually has multiple columns and contains many values. Attribute-based entity resolution is effective if the data is rich and noise free. But there are also challenges; for example, attributes in the two tables may require reconciliation as well; some attributes may be more important than others for measuring similarity for records [2]; and the data is never noise free. A possible approach is to regard attributes and their values as classes and entities on their own, and leverage relations among them to help solve the entity resolution problem. This leads to the relational entity resolution approach [1], which says two similar instances are likely to be the same entity if the instances that they are associated with are similar. Bhattacharya et al. [6] introduced the collective entity resolution approach, which strengthens this heuristic by requiring that the instances they are associated with must not only look similar, but need to be the same entities.

Recent approaches also leverage knowledge acquired from external sources for domain-independent entity resolution. For example, some approaches extract entities' surface forms (names or aliases) from Wikipedia and build a synonym dictionary for entities [14, 10, 21]. 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. These approaches assume 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 positive evidence (i.e., if x is a synonym of y, and y is a synonym of z, then x might be a synonym of z), although it may introduce some noise.

Some recent work employs *negative evidence* for entity resolution. Dong et al. [17] focus on the case where each entity has a set of attribute values. They use a propagation method to link related mentions, and use negative evidence to restrict the propagation. However, the constraint rules for negative evidence is handcrafted. Most recently, Whang et al. [34] use domain knowledge as negative rules. Entity resolution relies on attributes and values of the entities, and the approach does not scale well when the number of entities and rules become large. Basu et al. [4] introduced a probabilistic clustering framework using a Hidden Markov Random Field with must-links and cannot-links embedded in it. This clustering method is only applicable when the number of clusters is known a priori.

Ontology integration has been studied in the context of small-scale domain-specific ontologies. Specifically, ILIADS [30] employs a similarity measure, linearly combining lexical, structural and extensional similarities, between categories, entities, and properties. GLUE [16] trains a classifier using lexical similarity, and exploits the structure of ontology using relaxation labeling. It focuses on matching concepts between ontologies, assuming instances are rather clean. Meanwhile, [32] discusses mapping instances using an SVM classifier combining string and structural similarity. All these methods, however, assume integrating small-scale ontologies and cannot scale over a large number of entities and concepts. In addition, as web-scale ontologies cover a large number of domains, their assumptions of (a) high structural similarity between ontologies and (b) existence of sufficient human-labeled training data become unrealistic as well. For web-scale taxonomies [27, 11], cleansing is tightly coupled with population, using textual context for resolution. Our work complements these approaches by cleansing in a batch.

3. 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 many efforts are being devoted to composing and managing such taxonomies.

We choose two web-scale taxonomies, namely Freebase [9] and Probase [36], to study the problem of ontology mapping and entity resolution. From their major features listed in Table 1, we can see the two taxonomies are unique in their own ways. Probase is automatically harvested from web data. Although both Probase and Freebase are big, Probase is unique in the sense that it has an extremely large number of categories (2 million in Probase vs. 12.7 thousand in Freebase). For example, in Probase, *Emerging Markets* is a category on its own, and it includes subcategories such as *BRIC*, and instances or entities such as *China, Mexico, Russia, India, Brazil*, etc. The goal of Probase is to cover as many concepts of worldly facts in the collective mind of human beings as possible, in the hope that it will enable better understanding of human communications (natural language).

Table 1: Two unique taxonomies.

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

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 a date of birth 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.

Compared with manually constructed taxonomies, taxonomies automatically generated from data have advantages in scale and costs. Probase [36] 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) contains 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 [33], where there is a need to interpret users' intent.

Probase contains many IS-A relationships that are harvested using so called Hearst linguistic patterns [23], that is, SUCH AS like patterns. For example, a sentence that contains "... artists such as Pablo Picasso ..." can be considered evidence for the claim that *Pablo Picasso* is an instance in the *artist* category. For each category, Probase also collects a large set of attributes that can be used to describe instances in the category. For instance, the *artist* category may contain such attributes as *name, age, nationality, genre, specialization*, etc. Furthermore, Probase contains many relationships among instances of different categories. Figure 1 shows an interface which users can use to browse the Probase taxonomy, 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 integration, Freebase will have more information about categories, allowing 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.

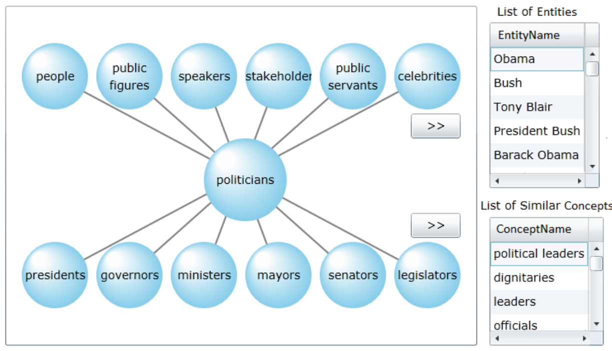

Figure 1: The Probase Taxonomy.

4. EXTERNAL EVIDENCE

In this section, we discuss how to obtain and quantify evidence from external sources, and in Section 5 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 describe 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 instance, assuming 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.

4.1 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 positive 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. However, 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 contain valuable clues for entity resolution. In our work, we derive negative 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 common. 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 in 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 $sim(c_1, c_2) \leq t$ as negative evidence for any claim (x, y) where $x \in c_1$, and $y \in c_2$.

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

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

where E_c, and A_c denote the set of elements and attributes in category c respectively, λ 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 IS-A 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.

4.2 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 2: String similarity as positive evidence.

id	claim	evidence
1	(*Bill Clinton*, *President Clinton*)	.33
2	(*George W. Bush*, *Dubya*)	0
3	(*George W. Bush*, *George H. W. Bush*)	.75

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 [14, 10]. 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 $x \sim y_i$, where x is the ambiguous phrase, and y_i 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 $x \sim y$.

4.3 Quantifying positive evidence

The evidence that supports 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 $x \cap y$ denotes the set of common words in x and y, and $x \cup y$ denotes the union of words in x and y. Alternatively, we can use Dice's coefficient, $w_{(x,y)} = \frac{2n_t}{n_x + n_y}$, where n_t is the number of character bigrams found in both strings, and 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 which contains the following instances {*riverdale high school, riverfall 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* $\sim$ *usa*, and 122 occurrences of *George W. Bush* $\sim$ *George Bush*. We can use the Sigmoid function to take into consideration the multiple occurrences.

$$w_{(x,y)} = \begin{cases} \frac{1}{1+e^{1-t}} & t > 0 \\ 0 & t \leq 0 \end{cases} \tag{2}$$

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 returns a vector of k scores, $(a_1, a_2, \cdots, a_k)$, 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.

$$1 - (1 - a_1) \cdot (1 - a_2) \cdots (1 - a_k) \tag{3}$$

Intuitively, the evidence for a claim has a high 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).

5. METHODS

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

5.1 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 $C \subseteq E$ 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:

- **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:

$$\sum_{e \in C} w(e)$$

- **Negative Evidence**: A valid solution G' should disconnect x and y rejected by some piece of negative evidence N_i.

This problem can be formalized as the *multi-multiway cut* problem [3]. 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 [3]:

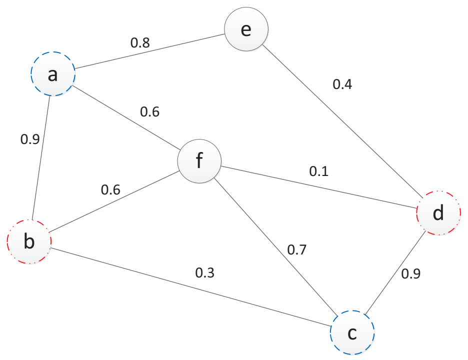

Figure 2: A graph with i) 6 entities connected by positive evidence of different weights, and ii) two pieces of negative evidence: $\{$a,c$\} \in N_1$ and $\{$b,d$\} \in N_2$.

$$\textbf{minimize} \sum_{e \in C} w(e)x(e)$$

$$\textbf{subject to}$$

$$\forall P \in \mathcal{P} : \sum_{e \in P} x(e) \geq 1$$

$$\forall e \in E : x(e) \in \{0, 1\}$$

where $\mathcal{P}$ 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, $O(10^{14})$ 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 [3] requires expensive LP optimization and $\mathcal{P}$ enumeration, and is still too expensive for our target problem. In addition, this solution has $O(logk)$ 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 4, 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 $x \in c_1, y \in c_2, sim(c_1, c_2) \leq t$.

5.2 Using the CnD Principle

As Section 4 discussed, negative evidence from some sources, e.g., Freebase created and maintained manually, does not contain any duplicates. Such cleanness can be exploited for efficient graph cut computation. More formally, we define clean data as follows:

DEFINITION 5.1 (CLEAN EVIDENCE). *k sets of negative evidence $N_1, \ldots, N_k$ is clean, if and only if $x \in N_i$ and $y \in N_j$ with different names cannot refer to the same real-life entity.*

For example, N_1' =$\{$*Bill Gates, Bill Clinton*$\}$ and N_2' =$\{$*William Gates, Bill Clinton*$\}$ are not clean, as Bill and Willam Gates refer to the same entity. In clear contrast, N_1 =$\{$*Bill Gates, Bill Clinton*$\}$ and N_2 =$\{$*Barack Obama, Bill Clinton*$\}$ is a clean set.

For a clean set, our optimization problem can be reduced to a special case of multi-multiway cut with $k = 1$, known as the *multiway cut* problem [15]. That is, we can aggregate k sets of negative evidence into a single set $\cup_i N_i$. For example, in the above clean set example, we can aggregate N_1 and N_2 into a single set $\cup_i N_i$ =$\{$*Bill Gates, Bill Clinton, Barack Obama*$\}$, suggesting any claim (x, y) for $x, y \in \cup_i N_i$ should be rejected. The same aggregation would not work for the dirty set example, as Bill and William Gates in $\cup_i N_i'$ =$\{$'Bill Gates', 'Bill Clinton', 'William Gates'$\}$ refer to the same person and should not be rejected.

The multiway cut problem is proved to be NP-hard when $m = |\cup_i N_i| \geq 3$, and an efficient "isolation heuristic" gives an approximation ratio of $2 - \frac{2}{m}$ [15]. In this paper, we discuss this heuristic, though there is an approximation algorithm with possibly better ratios, e.g., 1.348 [24], as all other existing solutions do not scale for our graph, requiring LP optimization.

The isolation heuristic works as follows. For any "terminal" $t_j \in \cup_i N_i$, 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 to 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, the union of the sets is an approximate answer. A more detailed description is available in Algorithm 1.

Algorithm 1 Isolation heuristics

```
Require: $G = (V, E)$, terminals $\cup_i N_i = \{t_1, \ldots, t_m\}$
  For each $t_i$, compute a minimum weight isolating cut $E_i$
  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 is both clean and dirty, we divide it into two sets N and N', i.e., $N \cup N' = \cup_i N_i$ 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 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 the isolating cut with the highest weight first.

As the problem is NP-hard, this greedy heuristic finds a suboptimal solution, as illustrated in Figure 2 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 because it violates N_1 and generates a path from a to c. Similarly, (b, f) is not an isolating cut because it generates 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 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: $G = (V, E)$, $N = N_1, ..., N_M$
  $E' = \phi$, 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.

5.3 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 $sim(c_1, c_2)$ in Eq. 1, combining the similarity between element sets E_c and attribute sets 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 [30].

- $f(A_{c_1}, A_{c_2})$**:** 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 *the 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 *album* is frequently represented by attributes $\{$*genre, artist, producer*$\}$, represented by an attribute frequency vector $\{0.9253, 0.8431, 0.8301\}$. Meanwhile, an 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_c complements E_c in the following two ways: *First,* when using E_c similarity alone, two entities with the same name cannot be distinguished. For example, *electronics* 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 companies* of size 306 and 108 have a few false matches, such as *electronics* and *automotive*, which generates a high Jaccard similarity score 0.0147. (Check Table 7 to see this is significant). Meanwhile, if considering A_c 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_c 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_c, though we attempt normalization, negatively affects the reliability of metrics, while A_c sizes are more balanced.

Accurate computation of both metrics requires us 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* $\sim$ *William Gates*, to recompute $f(E_{c_1}, E_{c_2})$ 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.

5.4 Class Matching Algorithm

For deciding threshold t defining matching E_c or A_c, we model both E_c and A_c as frequency vectors and estimate the distribution of *dot product similarity*, as it is the common numerator in both Jaccard similarity and cosine similarity. For each Freebase c_1, the distribution of its dot product similarity to all Probase classes, according to the observation in [28], can be modeled as inverse normal distribution and its mean and variance can be estimated from the mean and variance obtained from Probase distribution. This distribution is characterized by a mass of probability close to zero, corresponding to the pairs unmatched, with the minority long tail with high score, corresponding to matching pairs. We can identify t with drastic probability change as threshold.

Once t is identified, a naive solution for finding a matching class with the highest cosine similarity score would be all-pair computation, for 12.7 thousand $\times$ 2 million class pairs. Though a naive all-pair class comparison may be feasible for this specific case with a small number of categories in Freebase, it cannot scale for joining two large-scale ontologies, each with millions of categories (or, for self-joining Probase).

For more scalable computation of $f(A_{c_1}, A_{c_2})$, we can build inverted indices that map each attribute to a list of Probase classes (sorted in the order of probability), to efficiently locate the matching class and minimize computation of the similarity score. Such an index enables us to locate the matching Probase class with sublinear scan for each Freebase class and thus significantly reduce pairwise computation.

For $f(E_{c_1}, E_{c_2})$, we can view the problem as a *set similarity join* by the overlap of two sets X and Y. Many efficient algorithms have been proposed [37, 12, 5], which order all sets based on the same ordering. Once ordered, two sets with high enough similarity would share a prefix, and the length of the shared prefix can be used to safely prune out class pairs with short length. Using this principle, these algorithms drastically reduce the candidate pairs and are readily applicable to our problem to achieve scalability.

6. EXPERIMENTS

In this section, we evaluate our greedy Monte Carlo approach for entity resolution. We compare our method with three baseline algorithms in precision and recall. We also show the scalability of our method in comparison with the reasonably fast baseline.

6.1 Experiment Setting

We conduct comprehensive experiments in integrating Probase and Freebase. We build a distributed entity resolution system (as Figure 3 shows) that contains three servers running 64-bit Microsoft Windows Server 2003 Enterpriser SP2 OS, with 16 core 2.53 GHz Intel Xeon E5540 processors and 32 GB of memory. Each of the servers runs 10 instance mapping clients. All together, we run 30 instance mapping clients in parallel.

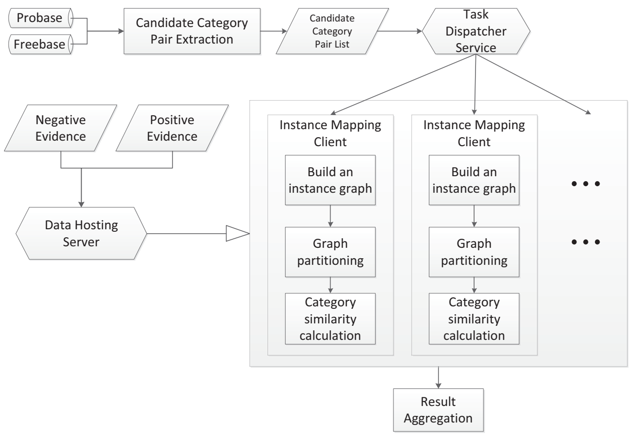

Figure 3: The Flowchart of the System.

We use Freebase data dump for 2010-10-14 [20]. 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 use 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.

6.2 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 3: Positive Evidence.

Wiki Source	# of Pairs
Links	12,662,226
Redirect	4,260,412
Disambiguation	223,171

Table 4: Negative Evidence.

Source	# of Bags
Wikipedia List	122,615
Wikipedia Table	102,731
Freebase	12,719

6.3 Baseline Algorithms

We compare our method with three other entity resolution techniques. Assume (X, Y) is a pair of categories.

- **Baseline #1 (Positive Evidence Based Method)**: This method maps $x \in X$ to $y \in Y$ if (x, y) has the strongest positive evidence (e.g., it has the largest number of $x \sim y$ evidence pieces). This method takes advantage of positive evidence, which is used in several recent works [14, 10].

- **Baseline #2 (String Similarity Based Method)**: This method maps $x \in X$ to $y \in Y$ if the string similarity between (x, y) measured by the Jaro-Winkler distance [35] is above a threshold δ. This method is a traditional approach to solve the entity resolution problem, and is usually used for name matching tasks [38]. In our experiment, we set a high threshold δ, as smaller thresholds produce too many false positives.

- **Baseline #3 (Markov Clustering Based Method)**: Among 12 graph clustering based algorithms discussed for entity resolution [22], we find five algorithms (marked as 'highly scalable' in the paper) to be applicable to web-scale taxonomy. Among these, we use Markov clustering [31] as a baseline, as it has shown the highest accuracy. Specifically, we employ the latest C implementation of MCL[1].

6.4 Instance Mapping

We match Freebase and Probase categories based on similarity in their attributes and elements. We find 763,000 matchings. Here, we look into 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*.

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* [2] 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. With regard to Baseline #3, as our final goal is instance mapping, we need to transform the resultant clusters to mappings from a Probase entity to a Freebase entity. Within a resultant cluster C of Baseline #3, if there is only one Freebase entity y in C, the rest of the entities in C, which are all from Probase, are mapped to y, and if there is more than one Freebase entity in C, we randomly choose one as y and the rest of Probase entities in C are mapped to y.

Table 5 shows the result for the 4 selected category pairs. We label 214, 134, 76, 201 Freebase entities for each case, and obtain 1,687 positive mapping pairs (408, 406, 384 and 489 for each category pair, respectively). For our method, we set different values for the threshold θ: 0, 0.05 and 0.10.

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 misspelled 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 Clin-

[1]http://micans.org/mcl/

Table 5: Precision and Recall for Selected Category Pairs. (P.: Precision, R.: Recall, S.: Output Size).

Probase Class Name	politicians			formats			systems			airline		
Freebase Type ID	/government/politician			/computer/file_format			/computer/operating_system			/aviation/airline		
Method	P.	R.	S.	P.	R.	S.	P.	R.	S.	P.	R.	S.
Baseline #1	0.9952	0.6603	603	0.9024	0.2517	1160	0.9040	0.3767	1110	0.9231	0.6326	712
Baseline #2 (δ=0.9)	0.9792	0.3110	451	0.7922	0.2743	1209	0.5965	0.2696	1420	0.9623	0.4788	743
Baseline #3	0.8860	0.6342	678	0.8228	0.4871	1561	0.8839	0.5210	1236	0.9696	0.5484	829
Our method (θ=0.00)	0.9815	0.8413	685	0.9370	0.8605	2222	0.9180	0.5967	5913	1.0000	0.7087	787
Our method (θ=0.05)	0.9851	0.8413	684	0.9654	0.7585	1766	0.9415	0.5367	1981	1.0000	0.7087	782
Our method (θ=0.10)	0.9924	0.8254	677	0.9951	0.6837	1581	0.9568	0.4433	1607	1.0000	0.7087	770

Table 6: Probase instances mapped to Freebase entities. 1) barack obama, barrack obama, senator barack obama, president barack obama, us president barack obama, and mr obama mapped to *Ricardo Mangue Obama Nfubea*. 2) windows vista, windows vista sp2, windows vista service pack 2 mapped to *Windows Vista Enterpriser.*

Freebase Entity	Baseline #1	Baseline #2 (δ=0.90)	Baseline #3	Our Method (θ=0.10)
Barack Obama	barack obama, barrack obama, senator barack obama, president barack obama	barack obama, barrack obama	$None^{1)}$	barack obama, barrack obama, senator barack obama, president barack obama, **us president barack obama**, mr obama
George W. Bush	george w. bush, president george w. bush, dubya, mr. bush	george w. bush	george w. bush, president george w. bush, **george h. w. bush**, george bush sr., dubya	george w. bush, president george w. bush, dubya, mr. bush, former president george w. bush
John Kerry	john kerry, senator john kerry, sen. john kerry, senator kerry	john kerry	john kerry, senator john kerry, senator kerry	john kerry, senator john kerry, sen. john kerry, senator kerry, massachusetts sen. john kerry, sen. john kerry of massachusetts
MP3	mp3, mp3s, mp3 files, mp3 format	mp3, mp3s	mp3, mp3 files, mp3 format	mp3, mp3s, mp3 files, mp3 format, high-quality mp3, mp3 songs
XLS	xls, microsoft excel	xls, xlsx	xls, excel file, excel documents, excel text	xls, microsoft excel, excel file, excel documents
Windows VISTA	windows vista, windows vista sp2, windows vista service pack 2	windows vista, windows vista sp2	$None^{2)}$	windows vista, windows vista sp2, windows vista service pack 2, microsofts windows vista
American Airlines	american airlines, american airline, aa, **hawaiian airlines**	american airlines, american airline	american airlines, american airline, aa, north american airlines	american airlines, american airline, aa

ton', '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. Baseline #3 shows higher recall than other baselines in general as it allows transitivity partially by clustering. However, it gives low precision because the clustering algorithm is based only on the similarity and connectivity of nodes (entities), so it is prone to produce error, due to false transitivity, especially for entities with similar names. 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 a large number 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'). Baseline #3 shows lower precision than Baseline #1, as in the case of 'politicians'. Our method outperforms the baselines in precision, recall, and output size.

In Table 6, 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 'barack obama' and 'us president barack obama' for example. Baseline #1 failed because it lacks positive evidence from Wikipedia saying 'barack obama' is 'us president barack obama,' and Baseline #2 failed because the similarity between the two is less than the threshold of .9. Baseline #3 failed because there is no general way to find the best cluster size so as not to merge two clusters representing two different entities. Moreover, this baseline also maps 'George W. Bush' to 'george h. w. bush' because of false transitivity. Our method worked because the positive evidence from string similarity was not disrupted by any negative evidence. As another example, for 'American Airlines,' a Wikipedia Disambiguation page provides a wrong piece of positive evidence: 'Hawaiian airlines' $\sim$ 'American Airlines.' Baseline #1 failed because of this, but our method was able to overcome the noisy evidence with negative evidence found in Wikipedia List (using the 'Clean data has no duplicates' principle). As yet another example, we have strong positive evidence that 'Microsoft Excel File' is 'XLS,' weak positive evidence that 'Microsoft .wav

Table 7: Similarities of Class Pairs. $\lambda = 0.2$.

Freebase Type	Probase Class	$\|E_{c_1} \cap E_{c_2}\|$	$f(E_{c_1}, E_{c_2})$	$f(A_{c_1}, A_{c_2})$	$sim(c_1, c_2)$
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

Table 8: Size of Each Task Set.

	$\sum \|C_1 \times C_2\|$
Task Set 1	24,377,292,299
Task Set 2	23,323,215,795
Task Set 3	24,923,006,886
Task Set 4	22,313,005,928

File' is 'Microsoft Excel File,' and through transitivity even weaker positive evidence that 'Microsoft .wav File' is 'XLS.' Without negative evidence, we might draw the wrong conclusion. However, since the positive evidence is very weak (because the matches are on words Microsoft and File, which have high frequency and hence a low score), such mistakes can be filtered out by a threshold as low as 0.005.

In Table 10, we show another five selected category pairs. Against these pairs, we just evaluate their precision, recall and output size in the same way, which are shown in Figure 6.

6.5 Finding Candidate Category Pairs

We demonstrate how Eq 1 implements the BoF principle in finding candidate category pairs. First, Table 7 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, or Musical Album that are titled China, Canada, etc.

Second, as the first 4 rows in Table 7 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 attribute 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.

6.6 Scalability

We now compare the scalability of our method with the winner baseline, Baseline #3 (MCL) [22]. In this comparison, we do not consider graph construction time because both algorithms require the same graph construction. We also note MCL is implemented in C, while ours in C#, which is a favorable setting for MCL.

In Figure 4, we can see, even in an unfavorable setting, our method significantly outperforms MCL in all categories. The gap is clearer in larger categories, e.g., ours 12 times faster than MCL in books, as MCL is sensitive to the size of largest component and ours is not. (Table 9).

We now evaluate the scalability of our method on a larger scale. For this experiment, we selected four sets of category pairs of roughly equal size (shown in Table 8), i.e., $\sum |C_1 \times C_2| \approx 23$ billion, where (C_1, C_2) is a pair of categories .

Table 9: The number of nodes in the largest connected subgraph. V_1 is the set of nodes in the largest connected subgraph of a graph $G = (V, E)$.

Category	$\|V_1\|$	$\|V\|$	$\|E\|$
companies	39,820	662,029	110,313
locations	53,694	964,665	118,405
persons	60,979	1,847,907	81,464
books	94,414	2,443,573	205,513

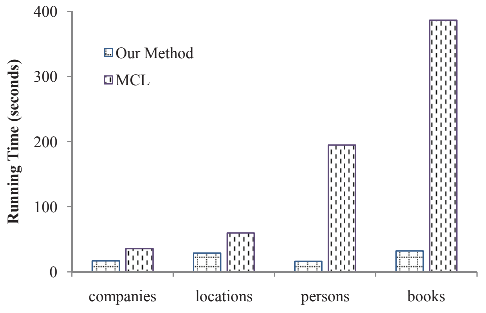

Figure 4: Running Time Comparison.

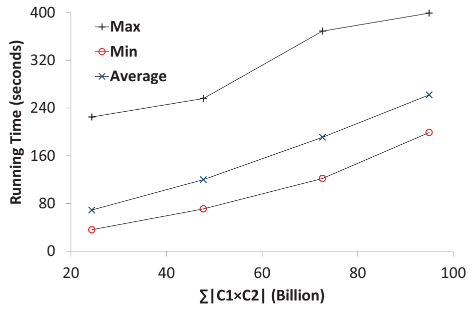

Figure 5: Running Time.

We now scale the pair size up to 100 billion, using Task Set 1, Task Set 1+2, 1+2+3, and 1+2+3+4 as input. As we process instance mapping in parallel, we measure the running time of each process and then compute their maximum, minimum, and average. Figure 5 shows the result. We see that the average running time is about 2 minutes to process 48 billion instance pairs, and almost linearly scales over the input size. The reason why the maximum running time of an instance mapping client is much longer than the average is that the size of category pairs has a very biased distribution: The Book (books, /book/book), Person (persons, /people/person), and Location (locations, /location/location) category pairs contain 2.4 million, 1.8 million, and 0.9 million instances respectively whereas other pairs may contain far fewer instances.

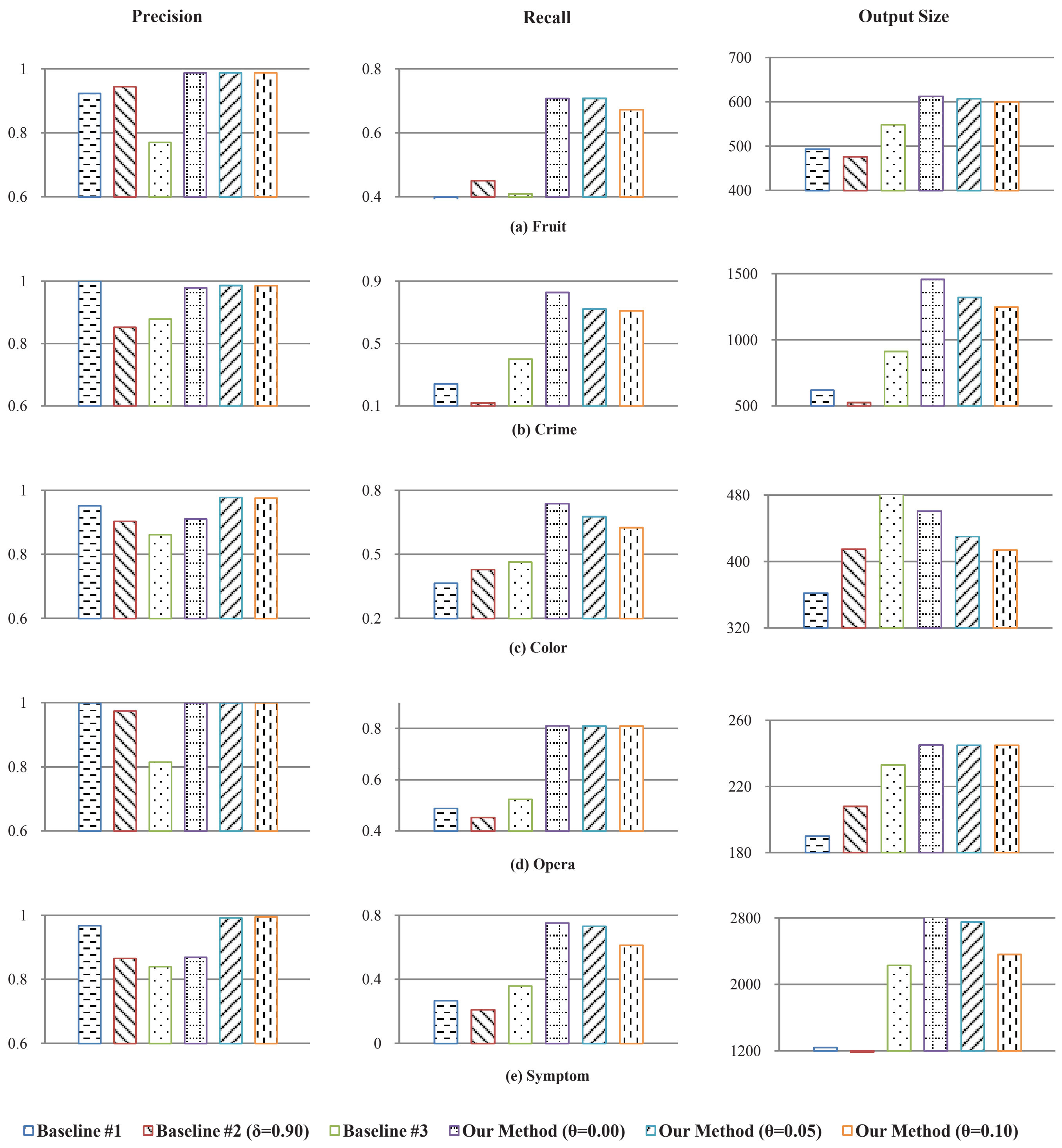

Figure 6: Precision, Recall and Output Size.

Table 10: Category Pair Name and Its Probase Class Name and Freebase Type ID.

Category Pair Name	Probase Class Name	Freebase Type ID
Fruit	fruits	/food/ingredient
Crime	crimes	/base/fight/crime_type
Color	colors	/visual_art/color
Opera	operas	/opera/opera
Symptom	symptoms	/medicine/symptom

7. CONCLUSION

Entity resolution for data integration is a challenging task. In this paper, we have studied 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 mechansim for taxonomies to enrich themselves by "borrowing" 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, 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.

8. ACKNOWLEDGMENTS

This research was supported by Microsoft Research and the MKE (The Ministry of Knowledge Economy), Korea, under IT/SW Creative research program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-C1810-1002-0003).

9. REFERENCES

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, pages 586–597, 2002.

[2] A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *SIGMOD*, pages 783–794, 2010.

[3] A. Avidor and M. Langberg. The multi-multiway cut problem. In *SWAT*, pages 273–284, 2004.

[4] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *SIGKDD*, pages 59–68, 2004.

[5] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *WWW*, pages 131–140, 2007.

[6] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *TKDD*, 1(1):5:1–5:36, 2007.

[7] I. Bhattacharya, L. Getoor, and L. Licamele. Query-time entity resolution. In *SIGKDD*, pages 529–534. ACM, 2006.

[8] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD*, pages 39–48, 2003.

[9] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[10] R. Bunescu. Using encyclopedic knowledge for named entity disambiguation. In *EACL*, pages 9–16, 2006.

[11] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, pages 1306–1313, 2010.

[12] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE*, page 5, 2006.

[13] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IJCAI*, pages 73–78, 2003.

[14] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, pages 708–716, 2007.

[15] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23:864–894, 1994.

[16] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *WWW*, pages 662–673, 2002.

[17] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, pages 85–96, 2005.

[18] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19:248–264, April 1972.

[19] A. Elmagarmid, P. Ipeirotis, and V. Verykios. Duplicate record detection: A survey. *TKDE*, pages 1–16, 2007.

[20] Google. Freebase data dumps. http://download.freebase.com/datadumps/, 2010.

[21] X. Han and J. Zhao. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *CIKM*, pages 215–224, 2009.

[22] O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller. Framework for evaluating clustering algorithms in duplicate detection. *PVLDB*, pages 1282–1293, 2009.

[23] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545, 1992.

[24] D. R. Karger, P. Klein, C. Stein, M. Thorup, and N. E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. In *STOC*, pages 668–678, 1999.

[25] A. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *SIGKDD*, pages 267–270, 1996.

[26] Y. Song, H. Wang, Z. Wang, and H. Li. Short text conceptualization using a probabilistic knowledgebase. In *IJCAI*, 2011.

[27] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.

[28] S. Tata and J. M. Patel. Estimating the selectivity of tf-idf based cosine similarity predicates. *SIGMOD Rec.*, 36:75–80, December 2007.

[29] S. Tejada and C. A. Knoblock. Learning domain-independent string transformation weights for high accuracy object identification. In *SIGKDD*, pages 350–359, 2002.

[30] O. Udrea. Leveraging data and structure in ontology integration. In *SIGMOD*, pages 449–460, 2007.

[31] S. van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, May 2000.

[32] C. Wang, J. Lu, and G. Zhang. Integration of ontology data through learning instance matching. In *WI*, pages 536–539, 2006.

[33] Y. Wang, H. Li, H. Wang, and K. Q. Zhu. Toward topic search on the web. Technical Report MSR-TR-2011-28, Microsoft Research, 2010.

[34] S. Whang, O. Benjelloun, and H. Garcia-Molina. Generic entity resolution with negative rules. *The VLDB Journal*, 18(6):1261–1277, 2009.

[35] W. E. Winkler. The state of record linkage and current research problems. Technical Report RR99/04, Statistical Research Division, U.S. Census Bureau, 1999.

[36] W. Wu, H. Li, H. Wang, and K. Zhu. Towards a probabilistic taxonomy of many concepts. Technical Report MSR-TR-2011-25, Microsoft Research, 2011.

[37] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *WWW*, pages 131–140, 2008.

[38] W. E. Yancey and W. E. Yancey. Evaluating string comparator performance for record linkage. Technical Report 2005-05, Bureau of the Census, 2005.