Table Union Search on Open Data

Fatemeh Nargesian
University of Toronto
fnargesian@cs.toronto.edu

Erkang Zhu
University of Toronto
ekzhu@cs.toronto.edu

Ken Q. Pu
UOIT
ken.pu@uoit.ca

Renée J. Miller
University of Toronto
miller@cs.toronto.edu

ABSTRACT

We define the table union search problem and present a probabilistic solution for finding tables that are unionable with a query table within massive repositories. Two tables are *unionable* if they share attributes from the same domain. Our solution formalizes three statistical models that describe how unionable attributes are generated from set domains, semantic domains with values from an ontology, and natural language domains. We propose a data-driven approach that automatically determines the best model to use for each pair of attributes. Through a distribution-aware algorithm, we are able to find the optimal number of attributes in two tables that can be unioned. To evaluate accuracy, we created and open-sourced a benchmark of Open Data tables. We show that our table union search outperforms in speed and accuracy existing algorithms for finding related tables and scales to provide efficient search over Open Data repositories containing more than one million attributes.

PVLDB Reference Format:
Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, Renée J. Miller. Table Union Search on Open Data. *PVLDB*, 11(7): 813-825, 2018.
DOI: https://doi.org/10.14778/3192965.3192973

1. INTRODUCTION

There has been an unprecedented growth in the volume of publicly available data from governments, academic institutions, and companies alike in the form of *Open Data*. Table 1 lists just three among the massive number of publishers and the size of their rapidly growing repositories. Data publishers often do not provide search functionality beyond simple keyword search on the metadata. This metadata varies greatly in quality across different datasets and publishers. The lack of sophisticated search functionality creates barriers for data scientists who want to use Open Data for their research.

Example 1: A data scientist is analyzing the current state of public funding of scientific research in the US and Canada, by all levels of governments (federal, state, etc.) and other funding agencies. She needs a master list of all publicly funded research projects. After searching the Web, she realizes that no such list currently exists in the public domain. Thus, she needs to create one. Her objective is to find tables on research project funding and union them into a master list. She first tries the US Federal Open Data portal[1], and searches for "research project funding" using keyword search. An overwhelming 20,184 datasets are found. She could manually go through the results to find useful data, or try different keywords. No matter what she does, it is going to be laborious work, and she also has to do the same for 50 states, all provinces of Canada, and many funding agencies. □

Proceedings of the VLDB Endowment, Vol. 11, No. 7

DOI: https://doi.org/10.14778/3192965.3192973

Table 1: Number of Published Open Data Tables as of March 2017.

Publisher	Number of Tables
United States (data.gov)	165,236
United Kingdom (data.gov.uk)	38,348
Canada (open.canada.ca)	11,809

A 2017 survey by CrowdFlower reported that data scientists spend on average 51% of their working time collecting and preparing datasets, but 60% reported that data preparation was the least enjoyable part of their job [9]. The same survey also indicates that 41% of data scientists use datasets from publicly available sources. Thus, a solution for more effectively finding unionable tables from Open Data has a very wide range of potential users.

1.1 Attribute Unionability

To solve the table union search problem, we need to first understand what it means for two attributes to be unionable. Since Open Data Tables rarely have informative schemas, we rely on data values to uncover as much information as possible to evaluate the unionability of attributes.

Example 2: Consider the example tables (Figure 1), one from NSF, one from NSERC. We have included meaningful attribute names to make the example easy to follow, but often such meaningful and unambiguous names are missing in Open Data. Clearly, if two attributes (like `country` and `Country`) contain a large overlap in their values, it seems reasonable to consider them "unionable", but an important question is how much overlap is sufficient?

Open datasets are published and maintained by many different people and organizations, thus the format of data values can be very different. For example, the attributes `location` and `Geo` contain no value overlap, but do contain the same semantic type of entities (major regions of countries). In `location`, these are represented by well-known two-letter acronyms, in `Geo` by full names. An ontology that contains the different types of representations can be used to determine that these two attributes contain values from the same semantic class. As with value overlap, an important question is how many values need to map to the same semantic class (or set of classes) for two attributes to be declared unionable?

[1]https://www.data.gov

pdPIName	location	country	estAmount	perfLocation	title	...
Charles...	MA	US	471150675	AURA	Construction of the Large...	
Charles...	MA	US	433789931	Associate of Universities for...	Mangement ... of the Gemini...	
Daphne...	CA	US	347150000	California Institute of Tech...	The Operation ... of the Laser...	

Name-Nom	Geo	Country	Amt	Institution-Établissement	ApplicationTitle	...
Abraham, ...	Ontario	CANADA	128700	University of Toronto	Probing ... Brightness Universe...	
Emily, ...	Québec	CANADA	87000	McGill University	Exploring the origin ... universe...	
Dobbs, ...	California	US	45000	University of California - Berkeley	The Migration and Growth ... Planets...	

Figure 1: Unionable Tables from NSF (Top) and NSERC (Bottom) Datasets

Open ontologies such as YAGO [34] and DBpedia [18], or specialized ontologies for specific domains, may help to identify syntactically different values that represent entities from the same semantic domain. However, even general purpose ontologies may have low coverage of values in Open Data. Consider the attributes, `pdPIName` and `Name-Nom` or `title` and `ApplicationTitle` which intuitively are unionable. Neither value overlap, nor an ontology will be sufficient to automatically determine this. However, from our understanding of natural language, we can recognize that the values in `pdPIName` and `Name-Nom` appear frequently in the same context in natural language since they are person names. Similarly, the title of scientific grants may contain scientific jargon that appears more often in the same context together than say the words describing movies or other titles. Again, the challenge is to define a principled mathematical framework for justifying how close the words in two attributes need to be before the attributes are considered unionable. Finally, which type of unionability (syntactic, semantic, and natural language) is the most suitable for estimating the unionability of a specific pair of attributes? □

1.2 Table Unionability

Once we have a principled way of understanding how likely it is that two attributes contain values drawn from the same domain, we also need a way of combining attribute unionability scores for a pair of tables. Ideally, the most unionable tables share a large number of highly unionable attributes with a query table. However, a candidate table might contain strongly unionable as well as weakly unionable attributes. An important question is how to define the unionability of two tables given the unionability of their attributes?

If table unionability decreases as more weakly unionable attributes are included, we need to consider the trade-off between the number of unionable attributes and the strength of attributes' unionability. Morever, one candidate table might have very few strongly unionable attributes and another candidate might have numerous attributes that are only slightly unionable with the query. In this case, to perform top-k search, how can we compare the unionability of tables if we are unioning on different numbers of attributes with different attribute unionability?

Example 3: Consider the two example tables in Figure 1. Attribute pairs (`country` and `Country`), (`location`, `Geo`), (`pdPIName`, `Name-Nom`), and (`title`, `ApplicationTitle`) are clearly unionable. Although attributes `perfLocation` and `Institution- Établissement` are also intuitively unionable, they are less unionable than other pairs since `perfLocation` contains hospital names, branches of government, and research centers in addition to the educational institutions of `Institution-Établissement`. We need to decide how the unionability of each attribute pair contributes to the unionability of NSERC and NSF tables. Now, suppose NSERC table is the query. For top-k search, how do we rank the NSF table (which shares a large number of weakly and highly unionable attributes with NSERC) and another candidate table which has fewer, but more strongly unionable attributes with the NSERC table? In this comparison, is it better to compare the unionability score of NSF based on three attributes or based on four attributes (including `perfLocation`)? □

1.3 Contributions

We present a new framework for table union search that makes the following contributions.

• We present a mathematical framework for determining the unionability of attributes. Unlike other approaches that use value overlap or class overlap heuristically to find unionable attributes, we present principled statistical tests for determining how likely a hypothesis that two attributes are unionable is to be true. We define *attribute unionability* using three statistical tests (measures) based on: (1) value overlap (set-unionability); (2) annotation of attributes with classes from an ontology and class overlap (sem-unionability); and (3) natural language similarity (NL-unionability).

• We propose a data-driven way of determining the best measure for evaluating the unionability of a specific pair of attributes.

• We present indices that permit us to quickly find an approximate set of candidate unionable attributes for each measure.

• We define the *table union search problem* as finding the k tables that have the highest likelihood of being unionable with a search table T on some subset of attributes. We present a principled way of determining if a table S that can be unioned with T on c attributes is more or less unionable than a table R that can only be unioned on $d < c$ attributes.

• We define (and publicly share) an Open Data table union benchmark and empirically compare the three attribute unionability measures on this benchmark. We show that NL-unionability can be used to effectively find unionable attributes, even when there is little value overlap and poor ontology coverage. We empirically demonstrate that in addition to greater precision and recall, table union using NL-unionability is also faster than other measures. We show that our solution for choosing among unionability measures (ensemble unionability) improves the accuracy of table union search with only a small runtime cost.

• We empirically show that table union search outperforms existing approaches for table stitching and finding related tables by finding unionable tables more accurately and more efficiently.

2. BACKGROUND AND RELATED WORK

A common first step to performing Open Data science is to search for joinable and unionable tables with a source table. In this scenario, the query is a table provided by a user and the search space consists of tables with potentially unavailable or meaningless schemas. In this paper, we argue that even when there is a reliable schema, which is rare, it is crucial to uncover as much information (or signal) from data values as possible to perform table search.

Web tables are tables extracted from Web pages [4]. The problem of finding related tables has been mostly studied for Web tables using queries that are (1) keywords, (2) a set of attribute values, (3) table schemas, and (4) table instances.

Keyword-based search. Most data portals provide keyword search functionality on metadata, if it is available for datasets. To find related tables to keyword queries, OCTOPUS performs Web document-style search on the content and context of Web tables [5]. These tables are then clustered into groups of unionable tables by means of syntactic measures such as attribute-to-attribute mean string length difference and attribute-to-attribute tf-idf Cosine similarity. Moreover, Pimplikar and Sarawagi present a search engine that finds tables similar to a query table that is represented by a set of keywords each describing a column of the query [27].

Attribute-based search. Existing work on the problem of finding linkage points (attributes) among data sources on the Web propose lexical analyzers and similarity functions accompanied by a set of effective and efficient search algorithms to discover joinable attributes [10]. Finding such linkages potentially leads to the alignment of data sources that have unavailable or non-overlapping schemas. Moreover, to find joinable tables, Zhu el al. [40] studied the domain search which is the problem of finding attributes that highly contain the values of a query attribute. The solution to the domain search problem uses an LSH index and a novel partitioning scheme to effectively find joinable tables over massive sets of attributes even with skewed cardinality distributions. Infogather studied *entity augmentation* which is the problem of extending a Web table containing a set of entities with new attributes. Query attributes are described by attribute names or attribute value examples [36]. Infogather applies schema matching techniques to precompute a correspondence graph between Web tables. The matching techniques rely on schema and instance features, page URLs, and the text around tables in HTML pages. This correspondence graph is then used at runtime to find relevant tables to query attributes and entities.

Schema and table instance-based search. Ling et al. defined table stitching as the task of unioning tables with identical schemas within a given site into a union table [22]. In order to provide a union table with semantically consistent values, Ling et al. augment the result with new descriptive attributes, extracted from the context of tables. For example, after stitching two tables which contain information about schools in NY and MA, this approach adds an extra attribute to the union table about the location of each school. Ling's work relies heavily on schema information and unions tables that have identical schemas.

Das Sarma et al. defines two HTML tables as *entity-complements* if they contain information about related sets of entities [31]. Entity-complement assumes that each table has a subject attribute which contains the entities that the table is about, and non-subject attributes which provide properties of the entities. It uses an aggregation of three measures: (1) entity consistency, (2) entity expansion, and (3) schema consistency. Entity consistency and expansion verify if subject attributes of two tables contain the same type of entities and if a candidate table adds new entities to the subject attribute of a query. To do so, entity-complement relies on the signals mined from high coverage ontologies curated from all data on the Web as well as publicly available ontologies (such as YAGO [34]). On the other hand, schema consistency determines if the non-subject attributes provide similar properties of entities by finding value-based and label-based matchings between non-subject attributes. The strength of entity-complement search is tied to the coverage of an ontology both for search space exploration and unionability evaluation. However, due to the breadth of tables in Open Data, ontology-based techniques are not always reliable. In this paper, we do not make any assumption on the existence of subject attributes for tables and propose a suite of syntactic and semantic measures that uncover unionable tables without requiring the presence of schemas or an ontology with full coverage.

A plethora of research has been done on scalable schema matching and ontology alignment, where the problem is to match large schemas (ontologies) encompassing thousands of elements [28, 26, 38] or to match more than two schemas [11, 33]. Although schema matching has been studied for tables with no or opaque attribute names [15], schema matching and ontology alignment have not been studied as search problems – find k best matches to a given query table except in the case that only data (and not the schema) is considered [11]. Recently, Lehmberg and Bizer have built upon the work by Ling et al. and created union tables of stitched tables by means of a set of schema-based and instance-based matching techniques [19]. Unlike all previous work on union which finds all unionable tables in a corpus (in one large batch computation), our table union search algorithm is designed to efficiently find the top-k unionable tables for a single query table over a large corpus.

3. ATTRIBUTE UNIONABILITY

In this section, we present a new principled framework for estimating the likelihood that two attributes contain values that are drawn from the same domain - a problem we call *attribute unionability*. We describe three types of domains for attributes and three statistical tests for testing the hypothesis that two attributes come from the same domain. Furthermore, we present a data-driven way to identify the type of domain two attributes are drawn from.

The test for the first domain type treats values as uninterpreted and uses the co-occurrence of the same value to estimate attribute unionability. The second test is useful for domains that have some values that can be mapped into an ontology (for example, attributes containing cities, countries, or diagnoses when a geography or medical ontology is available). This test can identify attributes from the same domain even if the values do not overlap (for example an attribute with Canadian Universities and one with US Universities). The final test does not require values to overlap either, but instead measures how semantically close the words in the attributes are. This test, which we call natural language unionability can be used for attributes like grant titles. Even though grant titles will not overlap, the semantic similarity of the words used in the titles can be used to estimate the likelihood that two attributes can be meaningfully unioned. Throughout this paper, a denotes the values of an attribute A.

3.1 Set Domains

In the simplest case, a domain D is a finite set of discrete values. In order to evaluate if attributes A and B are from the same domain D, we assume A (our query attribute) contains a random sample of D and we want to verify if B is also a sample of D. Since D is unknown and the only known values in D are the values of A, we use the size of intersection of the values in A and B as the test statistic for evaluating the probability that A and B are samples from the domain D. The size of intersection of A and B assuming they are drawn from the same domain follows a hypergeometric distribution [29]. Suppose D is a population that contains the values of A as success values and we draw $|a|$ samples from D, without replacement. If a draw is in the intersection of A and B, it is a success, otherwise it is a failure. In this scenario, the number of successful draws indicates how likely it is that A and B are from the same domain D. The maximum number of successful

draws from D is the size of the intersection of A and B. The hypergeometric test uses the hypergeometric distribution to calculate the statistical significance of the number of successful draws [29]. This provides a way to statistically test whether A and B are indeed from the same domain D by knowing the size of their intersection.

Let $C(m,n) = \begin{pmatrix} m \\ n \end{pmatrix}$ be the number of n items contained in m items. Let $n_a = |a|$, $n_b = |b|$, and $n_D = |D|$. If A is in domain D and B is drawn from D, then the distribution of s successful draws, $s \in \{1, \ldots, |a \cap b|\}$, is given by [29]:

$$p(s|\ n_a, n_b, n_D) = \frac{C(n_a, s)C(n_D - n_a, n_b - s)}{C(n_D, n_b)} \quad (1)$$

The probability $p(s)$ denotes the probability of achieving s successful draws (the size of intersection) when A is the set of successful draws in D and B is drawn from the same domain D. This allows the evaluation of the statistical significance of the intersection size s. That is, whether a give intersection size is likely to indicate that our hypothesis (that A and B come from the same domain) is correct.

Using the actual intersection, $t = |a \cap b|$, we can define the cumulative distribution of t:

$$\mathbf{F}(t|A,B) = \sum_{0 \le s \le t} p(s|n_a, n_b, n_D) \quad (2)$$

The cumulative distribution of a hypergeometric distribution, $\mathbf{F}$, is often used as a hypothesis test. Namely, we can *reject* our hypothesis that B comes from the same domain as A, if $\mathbf{F}(t|A,B) < \theta$, where θ is some confidence level (like .95). We will use $\mathbf{F}$ to define unionability of two domains.

Definition 1. *The* set unionability *of attributes A and B (where $t = |a \cap b|$) is defined as:*

$$\mathbf{U}_{\text{set}}(A,B) = \mathbf{F}(t|A,B) \quad (3)$$

Example 4: Consider the two tables from Section 1. Suppose D is the set of world country names, thus $N = 196$. Suppose attribute `country` of table NSF has values {UK, US, Canada, Mexico} and attribute `Country` of table NSERC has values {France, UK, Canada, Germany, Spain, Portugal, Italy}. Since these attributes have two overlapping values, the likelihood, of `country` and `Country` coming from the same domain D, $\mathbf{F}(t = 2|country, Country)$, is 0.9998. □

Our discussion so far requires knowledge of the cardinality of the domain D. For Open Data, it is impractical or impossible to know the true domain. Thus, we make a practical assumption that D is approximately the disjoint union of A and B, and therefore $n_D \simeq |a| + |b|$. Choosing a larger D only increases the likelihood, $\mathbf{F}(t|A,B)$, of the unionability of A and B, for a fixed intersection size. In Section 5, we describe how to efficiently search a large repository for attributes with high set unionablility with a query.

3.2 Semantic Domains

Under the set domain assumption, value overlap of two attributes provides a syntactic measure for evaluating the likelihood of their unionability. Notice that value overlap can easily be generalized to approximate value matching (or overlap of n-grams). However, some attributes may contain values that come from an ontology and this can provide more accurate information about the semantics of the attribute domains. In particular, two attributes can be semantically similar despite their syntactic dissimilarity. For example, consider attributes *Geo* and *location* of the two tables *NSERC* and *NSF* from Section 1. Since there is no overlap between the two attributes, using our hypothesis testing for set domains (Section 3.1), one would *reject* the hypothesis that these attributes have the same domain, and this would be a *false negative*.

Some domains may contain values that appear in an ontology. To keep our definitions simple, consider an ontology, with classes and entities that belong to classes: $\mathcal{O}(\mathcal{E}, \mathcal{C}, \mathcal{R})$, where $\mathcal{E}$ is a set of entities, $\mathcal{C}$ is a set of classes, and $\mathcal{R}$ is a set of pairs each containing an entity and a class indicating that the entity belongs to (*is-a*) that class. Notice that in this formulation an entity may belong to multiple classes. For brevity, we do not consider *is-a* relationships between classes, but our definitions can be easily extended to accommodate such relationships. Assuming *city* as a domain, two values that have *is-a* relationship with the class label *city* are semantically similar regardless of their syntax and regardless of whether the attributes they reside in overlap. We define a *semantic domain* as a subset of classes in $\mathcal{C}$ of ontology $\mathcal{O}$.

The problem of annotating attributes with class labels from an ontology has been well studied in the literature [21, 35]. A value in attribute A can be mapped to zero or more entites in $\mathcal{E}$. For attribute A, let $\mathcal{E}(a)$ be the set of entities of $\mathcal{O}$, to which some value in a can be mapped. In attribute annotation [35], the goal is to recognize a subset of classes in $\mathcal{C}$ that are most relevant to $\mathcal{E}(a)$. We can do this in a variety of ways, for example, we can take the union of all classes to which any entity in $\mathcal{E}(a)$ belongs, or more commonly in the literature, we can take the class (or classes in the case of ties) that contain the most entities in $\mathcal{E}(a)$ or the top-k most represented classes [30, 35]. Regardless of the method, we call the class annotations for attribute A, its semantic representation denoted $\hat{a} \subseteq \mathcal{C}$.

Now we can perform the same statistical test on the semantic values $\hat{a}$ of an attribute that we performed on the raw values a of the attribute. Our goal is to test the hypothesis that A and B are drawn from the same semantic domain $\hat{D}$.

As we did with raw attribute values in Section 3.1, we consider the intersection of $\hat{x}$ and $\hat{y}$ as an indicator that attributes A and B may be drawn from the same semantic domain $\hat{D}$. Let $\hat{n_a} = |\hat{a}|$, $\hat{n_b} = |\hat{b}|$, and $\hat{n_D} = |\hat{D}|$. We define the semantic intersection of A and B as:

$$\hat{t} = |\hat{a} \cap \hat{b}| \quad (4)$$

Following Section 3.1, if we assume that both $\hat{a}$ and $\hat{b}$ are random samples of $\hat{D}$, the size of the intersection $\hat{t}$ follows the hypergeometric distribution, which gives a way to statistically test whether A and B are indeed from the same semantic domain [29]. Assuming that $\hat{D}$ is the disjoint union of $\hat{a}$ and $\hat{b}$, we perform the following statistical hypothesis test using the cumulative distribution.

$$\mathbf{F}(\hat{t}|\hat{A},\hat{B}) = \sum_{0 \le s \le \hat{t}} p(s|\hat{n_a}, \hat{n_b}, \hat{n_D}) \quad (5)$$

Definition 2. *The* semantic unionability *of attributes A and B with class annotations $\hat{a}$ and $\hat{b}$, respectively, where $\hat{t} = |\hat{a} \cap \hat{b}|$ is defined as:*

$$\mathbf{U}_{\text{sem}}(A,B) = \mathbf{F}(\hat{t}|\hat{A},\hat{B}) \quad (6)$$

Example 5: Consider the two tables from Section 1. Suppose attributes `location`={MA, CA, 10701} and `Geo`={Ontario, Quebec, California, Toronto, 19600, M5G1Z2} and their class labels $\widehat{location}$ = {north_american_cities, east_coast_cities, west_coast_cities, american_planned_cities} and $\widehat{Geo}$ = {central_canada_cities, west_coast_cities, north_american_ cities, east_coast_cities}. Suppose $\hat{D} = \widehat{location} \dot{\cup} \widehat{Geo}$, where $\dot{\cup}$

refers to disjoint union. For attributes Geo and location, $\mathbf{U}_{\text{sem}}(\hat{t} = 3|\widehat{location}, \widehat{Geo}, \hat{D})$ is 0.9857. □

In Open Data, it may not be the case that all (or even most) values can be mapped to entities in the ontology. Using semantic unionability, only mappable values contribute to the unionability score. Therefore, we require a unionability measure that provides more semantic coverage of attributes.

3.3 Natural Language Domains

The presence of an ontology with large coverage of attribute values can greatly improve the accuracy of our unionability estimates. However, in Open Data, even with some of the largest and most precise open ontologies available such as YAGO [34], the coverage may be quite poor. Using an entity mapping technique and attribute annotation algorithm that closely follows Venetis et al. [35], we were able to partially or fully map 91% of attributes in Canadian Open Data to classes in YAGO. However, we observed that on average only 13% of values in attributes can be mapped to YAGO entities. The partial coverage of an ontology may result in $\mathbf{U}_{\text{sem}}$ misidentifying unionable attributes. In this section, we present an alternative way of measuring the semantics in an attribute that is based on natural language rather than ontologies.

Example 6: Consider the two tables NSERC and NSF from Section 1. Attributes pdPIName and Name-Nom contain the name of researchers and recipients of grants. These attributes are clearly (intuitively) unionable. However, since pdPIName and Name-Nom have very small value overlap, the statistical test would fail to identify them as set unionable attributes. Furthermore, since the names in pdPIName and Name-Nom most probably do not have many corresponding entities in standard ontologies, the statistical test for evaluating their semantic unionability would fail as well. □

For many attributes, the values are part of a natural language. Values of attributes such as names of people, grant keywords and product descriptions can also be found as part of English sentences which may or may not be found in ontologies. In contrast, domains such as product ID, unique identifiers, and Web URLs may not have values that are frequently used as part of a natural language. We call domains with values from natural languages *natural language domains*, or NL-domains. The unionability of two attributes is determined by some relation that associates values from the respective domains. Equality and ontology relations yield the unionability measures: $\mathbf{U}_{\text{set}}$ and $\mathbf{U}_{\text{sem}}$. However, Example 6 shows that they are not sufficient for NL-domains. We will discover hidden associations between NL-domains using the powerful method of word embedding [2, 8, 25, 32].

Word embeddings have been successfully employed in various information retrieval tasks [16, 37]. In word embedding technique each word is mapped to a dense high dimensional unit vector. Words that are more likely to share the same context have embedding vectors that are closer in embedding space according to Euclidean or angular distance measures. We are interested in natural language associations between domain values. We assume the words (attribute values) that share the same context are drawn from the same NL-domain. Therefore, we choose to use the *fastText* [14] word embeddings which are trained on Wikipedia documents. This allows NL-unionability to take advantage of the knowledge of embeddings trained using external sources and as we will show in Section 6 results in high accuracy of search.

In this paper, we apply word embedding vectors to define natural language domains. Furthermore, we design statistical tests that evaluate the likelihood that two attributes are drawn from the same natural language domain. Each value $v \in a$ is represented by a p-dimensional embedding vector $\vec{v}$. Thus, the embedding representation of attribute A, is a set of p-dimensional embedding vectors each corresponding to a value $v \in a$. We assume that an NL-domain is about one topic. Unlike semantic domains, the topic of an NL-domain is not restricted to classes in an ontology. Suppose μ_D is a vector that represents the *topic* of the values in an NL-domain D. The vectors $\vec{v}$ in domain D are statistically close to their topic vector μ_D.

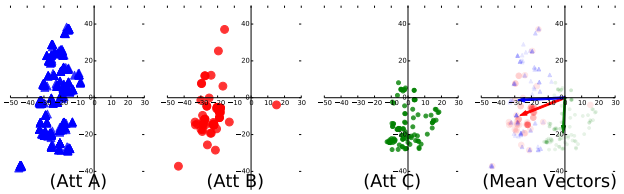


Figure 2: The Embedding Representation of three Attributes and their Topic (Mean) Vector Projected on 2-D Space.

Definition 3. *An* NL-domain *is a set D of values, each with p-dimensional embedding vector $\vec{v}$, such that they form a multivariate normal distribution centered around μ_D with some covariance matrix Σ_D:*

$$v \in D \implies \vec{v} \in \mathcal{N}(\mu_D, \Sigma_D) \tag{7}$$

For instance, consider the domain of researchers, where each instance of *researcher* domain is represented by a p-dimensional unit vector. The vectors are trained such that instances that have similar context are represented by vectors that have small angular distance (thus, small Euclidean distance, because embedding vectors are unit vectors) [14, 25]. Since researchers appear in similar contexts in the natural language, their corresponding embedding vectors are closely located in p-dimensional space. We assume that the embedding vectors of instances of an NL-domain (such as *researcher* domain) form a normal distribution with parameters μ_D and Σ_D.

Example 7: Figure 2 illustrates the embedding vector representation of sample values from three attributes: (A) sci-fi movie directors of year 2000, (B) sci-fi movie directors of year 2010, and (C) university names. The embedding vectors of attribute values are generated using pre-trained publicly available 300-dimensional word embedding vectors [14] and projected on two-dimensional space [23]. Each point in this figure represents an attribute value. It is obvious that the values of attributes *A* and *B* are closely located in the embedding space and are likely to be drawn from the same distribution, while values of attribute *C* are drawn from a different distribution. The vectors shown in Figure 2 demonstrate the topic vectors of these attributes. □

Our assumption on the distribution of embedding vectors in the same domain allows the application of a statistical test to determine whether two attributes share a common NL-domain. Attribute A is represented by set $\vec{a} = \{\vec{v_1}, \ldots, \vec{v_{n_a}}\}$, where $\vec{v_i}$ is the embedding vector of value v_i in a. Assuming $\vec{a}$ has a p-variant normal distribution, we can estimate the mean and covariance of $\vec{a}$. Similarly, we can define $\vec{b}$ and its mean and convriance. The distance between the estimated mean of two sets of embedding vectors is an indicator of whether they are drawn from the same domain with normal distribution. The Hotelling's two-sample statistics has been defined to undertake tests of the differences between the multivariate means of different sets with normal distribution [12].

Given two attributes A and B and their corresponding embedding representation $\vec{a}$ and $\vec{b}$, the Hotelling's statistics can be con-

structed as follows. Let the sample mean of the embeddings of attribute values be (recall $n_a = |\vec{a}|$):

$$\bar{a} = \frac{\sum_{v \in \vec{a}} \vec{v}}{n_a} \tag{8}$$

The sample covariance matrix is given by:

$$S_a = \frac{1}{n_a - 1} \sum_{v \in \vec{a}} (\vec{v} - \bar{a})(\vec{v} - \bar{a})^T \tag{9}$$

Similarly, we define $\bar{b}$ and S_b. The Hotelling's two sample statistics is:

$$T^2(A, B) = \frac{n_a n_b}{n_a + n_b} (\bar{a} - \bar{b})^T S^{-1} (\bar{a} - \bar{b}) \tag{10}$$

where S is an estimation of the covariance of $\bar{a}$ and $\bar{b}$, namely the pooled covariance:

$$S = \frac{(n_a - 1)S_a + (n_b - 1)S_b}{n_a + n_b - 2} \tag{11}$$

If A and B are samples of a p-variant normal distribution, then we know that the T^2 has a well defined probability distribution known as the F-distribution (parameterized by p and $n_a + n_b - 2$ degree of freedom) [12]. Hence, T^2 can be used as a way of converting the distance between topic vectors to the probability of belonging to the same NL-domain.

Definition 4. *The* natural-language unionability *of attributes A and B represented by $\vec{a}$ and $\vec{b}$ is defined as:*

$$\mathbf{U}_{\text{nl}}(A, B) = 1 - \mathbf{F}(T^2, p, n_a + n_b - p - 1) \tag{12}$$

where F is the cumulative distribution function of F-distribution.

F-distribution is inversely proportional to Hotelling's T-squared statistic. In other words, as the $T^2(A, B, p)$ increases, the probability of A and B being from the same NL-domain decreases. Therefore, we use the inverse of T-squared as a way of comparing the unionability of A and B. In Section 5, we explain how we estimate Hotelling's T-squared likelihood to efficiently search for unionable attributes drawn from the same NL-domain.

Using pre-trained word embedding models for NL-unionability is limited to the vocabulary of the training corpus. This results in NL-unionability not having full coverage on Open Data attribute values (such as codes). Next, we describe how we address the limitations of all unionability measures by picking the best-suited one during search.

3.4 Ensemble Unionability

We presented three tests for measuring the unionability of attributes based on the assumption of knowing the type of domain they are drawn from.

Example 8: Consider the two tables NSERC and NSF from Section 1 and their unionable attributes from Examples 4, 5, and 6. Attributes of NSERC and NSF tables are drawn from different types of domains. Attributes `country` and `Country` are clearly drawn from the set domain of countries and have $\mathbf{U}_{\text{set}}$=0.9998. Although attributes `Geo` and `location` have very small value overlap, they are drawn from a semantic domain with $\mathbf{U}_{\text{sem}}$=0.9857. Moreover, `pdPIName` and `Name-Nom` are clearly NL-unionable despite small raw value and ontology class overlap. Attributes of two tables can be samples drawn from different types of domains, thus we need to be able to pick the best measure, even though the measures are not directly comparable. □

In this section, we describe how the unionability of attributes is evaluated without knowing the domain type by using the ensemble of the three unionability measures. Suppose q_m is the distribution of unionability probabilities generated by measure U_m, $m \in \{Set, Sem, NL\}$, in corpus $\mathcal{C}$. The cumulative distribution function of $U_m(A, B)$, namely $\mathbf{G}_m(A, B)$, is defined as follows:

$$\mathbf{G}_m(A, B) = \sum_{s' \leq U_m(A,B)} q_m(s') \tag{13}$$

Intuitively, $\mathbf{G}_m(A, B)$ is the probability of A and B being the most unionable pair in $\mathcal{C}$ according to measure m. In other words, $1 - \mathbf{G}_m(A, B)$ is the probability of finding an attribute pair in $\mathcal{C}$ that has higher m-unionability score than $U_m(A, B)$. We call $\mathbf{G}_m(A, B)$ the goodness score of m-unionability of A and B. We will use $\mathbf{G}_m$ to compare the unionability of attributes across the three measures. In the ensemble of the three measures, we select the unionability measure that has the highest goodness.

Definition 5. *Given an attribute pair A and B, and the goodness scores $G_m(A, B)$, $m \in \{Set, Sem, NL\}$, the ensemble-unionability of A and B is:*

$$\mathbf{U}_{\text{ensemble}}(A, B) = \max_{m \in \{\text{Set,Sem,NL}\}} \mathbf{G}_m(A, B) \tag{14}$$

Ensemble-unionability provides a way of ranking attributes with respect to a query attribute even if different measures are found to be best-suited for unionability evaluation. Moreover, the goodness score of A and B is monotonically increasing with respect to the unionability probability of A and B. In Section 5, we show how this property allows us to perform efficient attribute search based on $\mathbf{U}_{\text{ensemble}}$.

Finding the best unionability measure to evaluate an attribute pair requires knowing the distribution of unionability probabilities for each measure in a corpus. Modelling the exact distributions for a corpus of n attributes involves computing the unionability probabilities for n^2 attribute pairs. Large corpuses, such as Canadian Open Data, contain too many attributes to permit exact computation of the distribution. To overcome this, in Section 5, we propose an efficient algorithm that accurately approximates the unionability distributions of the three measures for large corpuses.

4. TABLE UNIONABILITY

Given a source table *S*, we want to discover tables which are unionable with *S*. We will use *X* and *Y* to denote sets of attributes in tables *S* and *T*. We assume *S* and *T* are unionable if there is a one-to-one alignment $\mathcal{A}$ between subsets of *X* and *Y* such that the aligned attributes are highly unionable. To define table unionability, we need to define a unionability score for alignments and we need to define a meaningful way of comparing the unionability of alignments between sets of attributes of different sizes.

Definition 6. *A* c*-alignment is a one-to-one mapping $h : X' \to Y'$ such that $X' \subseteq X$, $Y' \subseteq Y$, and $|X'| = |Y'| = c$. We denote all c-alignments from X to Y as $\mathcal{A}^c(X, Y)$.*

Assuming independence among the unionability probability of attributes, we can compute an *alignment unionability* probability. Let h be a *c-alignment* over $X = \{A_1, \ldots, A_c\}$.

Definition 7. *The unionability score of a c-alignment $h : X \to Y$ is the joint probability of the unionability of attribute pairs, namely $U(A_i, h(A_i))$, in the alignment.*

$$U(h) = \Pi_{i \in \{1,\ldots,c\}} U(A_i, h(A_i)) \tag{15}$$

For a pair of relations S and T, we define their *c-unionability* as the highest unionability score of all possible c-alignments.

Definition 8. *The* c*-unionability of S and T, written as $\sigma^c(S,T)$, is defined as:*

$$\max\{U(h) : h \in \mathcal{A}^c(X,Y)\} \tag{16}$$

Any c*-alignment* h *that achieves this maximum is called a max-*c*-alignment.*

Example 9: Consider the two tables from Section 1 and the following ensemble attribute unionability scores.

$$\mathbf{U}(\texttt{NSF[pdPIName]},\texttt{NSERC[Name-Nom]}) = 0.7$$
$$\mathbf{U}(\texttt{NSF[location]},\texttt{NSERC[Geo]}) = 0.9$$
$$\mathbf{U}(\texttt{NSF[location]},\texttt{NSERC[Country]}) = 0.2$$
$$\mathbf{U}(\texttt{NSF[country]},\texttt{NSERC[Geo]}) = 0.4$$
$$\mathbf{U}(\texttt{NSF[country]},\texttt{NSERC[Country]}) = 0.9$$

Assume (to simplify the example) that all other ensemble attribute unionability scores are 0.1 or lower. The max 2-alignment h_2 is:

$$h_2(\texttt{NSF[location]}) = \texttt{NSERC[Geo]}$$
$$h_2(\texttt{NSF[country]}) = \texttt{NSERC[Country]}$$

The max 3-alignment h_3 is

$$h_2(\texttt{NSF[pdPIName]}) = \texttt{NSERC[Name-Nom]}$$
$$h_2(\texttt{NSF[location]}) = \texttt{NSERC[Geo]}$$
$$h_2(\texttt{NSF[country]}) = \texttt{NSERC[Country]}$$

The concept of max c-alignment allows us to compare alignments of the same size (same number of attributes). But it does not provide a way of comparing alignments of different sizes. Since our best max 2-alignment in this example is a subset of our best max 3-alignment, it is necessarily the case that $U(h_2) \geq U(h_3)$. But this does not imply that h_2 is a better alignment. □

To compare alignments of different scores, we define table unionability relative to all possible c-alignments. Suppose r_c is the distribution of unionability probabilities of max-c-alignments (σ^c) in corpus $\mathcal{C}$. In Section 5, we describe how this function is estimated for a corpus. The cumulative distribution function of $\sigma^c(S,T)$, namely $\mathbf{J}_c(S,T)$, is defined as follows:

$$\mathbf{J}_c(S,T) = \sum_{s' \leq \sigma^c(S,T)} r_c(s') \tag{17}$$

Intuitively, $1 - \mathbf{J_c}(S,T)$ is the probability of finding another pair of tables in $\mathcal{C}$ with higher σ^c than that of S and T. We call $\mathbf{J_c}(S,T)$ the c-goodness of S and T. This notion allows the comparison of c-alignments of two table across c values. It also provides a way of ranking tables with respect to a query table. Notice that although c-goodness of a table pair is monotonically increasing with respect to their c-unionability, it is not monotonically increasing with respect to c.

Definition 9. *Given a source table* S *and a candidate table* T*, and c-goodness scores $\mathbf{J}_c(S,T)$, for $c \in \{1,\ldots,|X|\}$, the table unionability score of* S *and* T *is defined as follows:*

$$U(S,T) = \max_{c \in \{1,\ldots,|X|\}} \mathbf{J}_c(S,T) \tag{18}$$

Any alignment h that achieves this maximum is called a max-alignment.

From all tables in a repository, the table unionability search problem is to find the top-k tables based on table unionability scores.

Definition 10. Table union search problem. *Given a set of tables $\mathcal{T} = \{T_1,\ldots,T_z\}$ and a query table* S*, the top-*k *table union search problem is to find* k *tables in $\mathcal{T}$, whose table unionability with* S *is the highest.*

Example 10: Continuing Example 9, the c-unionability of h_2 is 0.81 and of h_3 is 0.567. To understand which is a better alignment, we compare each alignment with the distribution of all other 2-alignments (and 3-alignments) respectively. If we find that h_2 is in the 90th percentile of 2-alignment scores and h_3 is in the 80th percentile of 3-alignment scores, then we would return h_2 as it is more surprising (higher-rated) alignment. If however, h_2 is only in the 70th percentile, meaning there are lots of other max 2-alignments that are better than this, but h_3 is in the 85th percentile, then we would just return h_3 to be the better alignment. □

Given a repository $\mathcal{R}$ of n tables where each table has on average t attributes and an attribute unionability measure m, searching for top-k unionable tables with a query table $\mathcal{Q}$ with s attributes has a complexity of $O(n \times C(s \times t, min(s,t)) \times f_m \times min(s,t))$, where $C(s \times t, min(s,t))$ is the number of c-alignments between R and a candidate table, and f_m is the complexity of computing the attribute unionability score of an attribute pair using measure m. Because of this large complexity, we give an approximate solution that uses indices to efficiently retrieve a set of likely candidates.

5. ALGORITHM

Given a query table S, our goal is to find the k tables with the highest unionability with S. To do this interactively over massive repositories of data, we cannot afford to do a linear scan. To perform table union search for S, we first find most unionable attributes with attributes of S, which gives us a set of candidate unionable tables. Then, we compute exact unionability scores only for the candidates. The challenge we face is that to the best of our knowledge, there are no efficient indexing structures for any of our three unionability measures. To overcome this challenge, we observe empirically that each of our unionability measures is highly correlated with well-known metrics (specifically Jaccard and Cosine similarity) that can be efficiently indexed using Locality Sensitive Hashing (LSH) [20]. We rely on this observation to find a set of candidate unionable attributes and tables using the index.

5.1 Attribute Unionability Search

In attribute search, our goal is to find the most unionable attributes with a given attribute A considering $\mathbf{U}_{\text{ensemble}}$. According to Definition 5, the ensemble-unionability maximizes the goodness score of an attribute pair across all three measures. Finding attributes with high goodness scores for at least one measure gives a set of candidate attributes with high $\mathbf{U}_{\text{ensemble}}$ scores. By definition, the goodness score is monotonically increasing with respect to unionability probability. Thus, attributes with high unionability probability based on any of unionability measures are candidates for high $\mathbf{U}_{\text{ensemble}}$ scores. Attribute search using any unionability measure is a nearest neighbor search. Locality Sensitive Hashing (LSH) provides an efficient solution [20], which we use as an index structure for attribute unionability search. Next, we explain how we use index structures for different measures to find candidate attributes. Notice that we compute exact ensemble-unionability only on merged candidates returned by each of indices. In Section 5.3, we describe how we estimate the unionability distributions required for calculating goodness scores in practice.

5.1.1 Set Unionability

We observed that in Open Data, the cumulative hypergeometric distribution of the intersection of a pair of attributes (their $\mathbf{U}_{\text{set}}$) is positively correlated with their Jaccard similarity. Figure 3a shows this correlation for 40K attribute pairs from Canadian Open Data. Attributes with high Jaccard similarity have high $\mathbf{U}_{\text{set}}$ scores. Moreover, the goodness score is monotonically increasing with respect to unionability probability. Thus, attributes with high Jaccard similarity have high goodness score. To identify attributes with high goodness of $\mathbf{U}_{\text{set}}$, we use *minhash LSH* [3] to find attributes with high Jaccard similarity. We then compute the exact ensemble-unionability score of these pairs. Hence, we are using the Jaccard score as a filter to efficiently find attributes that have high ensemble-unionability.

5.1.2 Semantic Unionability

Similar to $\mathbf{U}_{\text{set}}$, attributes with high Jaccard similarity of annotation classes have high $\mathbf{U}_{\text{sem}}$, thus high goodness scores. For semantic unionability, we construct an LSH index over minhash of the class annotations of attributes, rather than the raw values. Building LSH indices for set and sem-unionability can be done efficiently, even for large repositories [40].

5.1.3 Natural Language Unionability

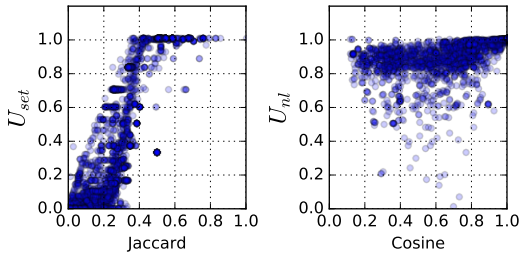
To efficiently identify natural language unionable attributes, we unfortunately cannot use the Jaccard similarity. Through empirical evaluation, however, we observe that the Cosine similarity of the mean vectors of attributes' embedding vectors is strongly negatively correlated with T-squared and positively correlated with $\mathbf{U}_{\text{nl}}$. Specifically, suppose that attribute A with values a is represented by a set of vectors $\vec{v_i}$, where $\vec{v_i}$ is the embedding vector of value v_i in a. The sample mean of the embeddings of attribute values of A is (recall $n_a = |\vec{a}|$):

$$\bar{A} = \frac{\sum_{v_i \in a} \vec{v_i}}{n_a} \tag{19}$$

We observe that there is a positive correlation between the $\mathbf{U}_{\text{nl}}$ of attributes A and B and the Cosine similarity of their $\bar{A}$ and $\bar{B}$. Figure 3b shows this correlation for 10K attribute pairs from Canadian Open Data. This empirical observation is aligned with the Hotelling test (the basis for $\mathbf{U}_{\text{nl}}$) which assigns higher probability to populations with close mean and small covariance [12]. According to Definition 4, we favor attribute pairs with low T-squared scores. To identify attributes with high natural language unionability, thus goodness score, we use *simhash LSH* [6] to find attributes with high Cosine similarity. We later compute the exact ensemble-unionability score of these pairs.

Example 11: Figure 2 illustrates the mean vectors of the embedding vectors of three attributes described in Example 7. Attributes A and B belong to the same domain (domain of *sci-fi movie directors*) and their representative mean vectors have high Cosine similarity, while the mean vector of attribute *C* which is drawn from the domain of *university names* has a small Cosine similarity with *A* and *B*. Thus, assuming *A* is a query attribute, it is more likely that a simhash LSH index built on attributes *B* and *C* returns *B* as a candidate unionable attribute. □

Notice that our indexing techniques let us retrieve a set of attributes that are likely to have high unionability. In Section 6, we show that these approximations (using Jaccard similarity as a surrogate for set and semantic unionability, and using Cosine similarity as a surrogate for natural language unionability) do not lead to missing many of the actual most unionable attributes.



(a) **(b)**

Figure 3: (a) Jaccard vs. $\mathbf{U}_{\text{set}}$ and (b) Cosine vs. $\mathbf{U}_{\text{nl}}$.

5.2 Table Search

Given a set of tables $\mathcal{T} = \{T_1, \ldots, T_z\}$ and a query table S, the table union search problem is to find k tables in $\mathcal{T}$, whose unionability with S is the highest. The unionability of S and T, with attributes X and Y, is the alignment h that maximizes goodness score. We consider all alignments of size up to $|X|$. To compute the unionability score of alignments ($U(h)$), we can use any measure or the ensemble-unionability of attributes. Recall attribute unionability score of A and B is $\text{U}(A, B)$. Since $U(h)$ is upper bounded with its highest attribute unionability goodness, $max_{i=1,\ldots,c}(U(x_i, h(x_i))$, in our search, we approximate the score of an alignment with the maximum attribute unionability score in the alignment.

Definition 11. *Given tables* S *and* T *(with attributes* X *and* Y*), and alignments $\mathcal{A}$, the maxattr of an alignment* h *is:*

$$maxattr(h) = max_{i=1,\ldots,c}\{U(A_i, h(A_i))\} \tag{20}$$

where $A_i \in X$ and c *is the size of* h.

In top-k table union search, we seek k tables with highest *maxattr*. The monotonicity property of *maxattr* with respect to attribute unionability probability allows early pruning of certain tables without exactly computing their scores. Specifically, if T has a *maxattr* score smaller than the k other tables, we prune T and no longer consider it. Obviously, T could have an exact unionability score that is better than some of the k tables with higher *maxattr*. The performance experiments of Section 6 shows that this approximation does not lead to missing many of the actual top-k most unionable tables.

The table union search procedure is described in Algorithm 1. For a given set of tables $\mathcal{T}$ and for each unionability measure f, we build an index, $\mathbf{I}_f(\mathcal{T})$, on attributes in $\mathcal{T}$ as described in Section 5.1. Given a query table S, `Att-Search` probes the indices with all attributes in S, in parallel, and gets candidate attributes with high goodness scores in a streaming fashion. Candidates are processed in batches. For each batch, `Att-Search` computes the exact ensemble-unionability of candidate attributes and ranks them from highest-to-lowest ensemble-unionability. Since *maxattr* is an upper bound for table unionability score, tables of attributes in $\mathcal{A}_{\text{union}}$ are candidate unionable tables. The goal is to find top-k unionable tables, hence, we only need to rank candidates based on their table unionability.

To find max-alignments of a candidate T and S, `Align` finds the max-c-alignments of S and T for all possible c's. To achieve that, `Align` only needs to build the max-c-alignment of tables for the largest possible c. Given tables S and T, we build a bipartite graph, $\mathcal{G}$, where nodes are the attributes of S and T and edges are weighted by the unionability score of attributes. A matching of c edges in $\mathcal{G}$ (called c-matching) is equivalent to a c-alignment. Thus, finding a

Algorithm 1: Table Union Search

Table-Union-Search($\mathbf{I}_{\text{set}}(\mathcal{T})$, $\mathbf{I}_{\text{sem}}(\mathcal{T})$, $\mathbf{I}_{nl}(\mathcal{T})$, S, k)
Input: $\mathbf{I}_{\text{set}}(\mathcal{T})$, $\mathbf{I}_{\text{sem}}(\mathcal{T})$, $\mathbf{I}_{nl}(\mathcal{T})$: indices on attributes in $\mathcal{T}$, S: a query table, k: search parameter

let $\mathcal{T}_{\text{union}} \leftarrow \emptyset$, $\mathcal{A}_{\text{union}} \leftarrow \emptyset$
while $|\mathcal{T}_{\text{union}}| \neq k$ **do**
  $\mathcal{A}_{\text{union}}$ = `Att-Search`($\mathbf{I}_{\text{set}}(\mathcal{T})$, $\mathbf{I}_{\text{sem}}(\mathcal{T})$, $\mathbf{I}_{nl}(\mathcal{T})$, S)
  for $T \in$ `Get-Tables`($\mathcal{A}_{\text{union}}$) **do**
    $\mathcal{A} \leftarrow$ `Align`(S, T, $\{1, \ldots, \min(|S|, |T|)\}$)
  $\mathcal{T}_{\text{union}} \leftarrow$ `Merge`($\mathcal{T}_{\text{union}}$, `Rank`($\mathcal{A}$));
return $\mathcal{T}_{\text{union}}$

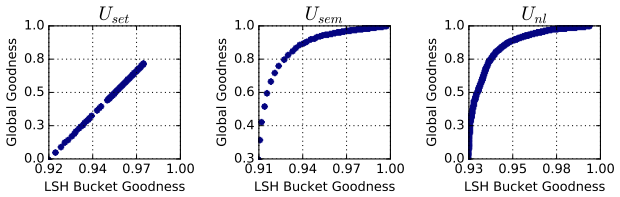


Figure 4: The estimation of unionability distribution using all pairs (3 millions) versus pairs in LSH buckets.

max-c-alignment of S and T becomes a bipartite graph c-matching problem. The max-c-alignment of S and T is the c-matching, with the maximum unionability score. `Align` implements the greedy solution for graph matching and builds all alignments incrementally. The process of searching unionable attributes and aligning tables is done iteratively until k tables with max-alignments are found.

5.3 Estimating Unionability Distributions

Computing goodness and c-goodness scores requires the distributions of attribute unionability probability and table unionability probability. We assume that unionability scores between a query and tables in a sufficiently large corpus follow a similar distribution to that of all attribute and table pairs in the corpus. Therefore, the distributions can be pre-computed. Doing this exactly would require a quadratic number of computations in the number of attributes and tables in a corpus.

To do efficient search, LSH indices bucketize attributes into groups of highly unionable pairs. Therefore, LSH buckets contain attribute pairs that cover the head (highest values) of probability distribution. The attributes that do not appear in the same bucket are likely to be pairs from the tail of the distribution and their unionability can be estimated as zero. Therefore, estimating unionability distribution requires computing scores only for the pairs in the same index buckets.

Figure 4 shows the goodness scores in a corpus of three million attribute pairs when cumulative distribution functions are computed using all pairs versus using only pairs of the same buckets. The goodness rankings are strongly correlated, especially for pairs with high goodness scores, which are the ones returned by indices. In fact, the calculated Spearman's rank correlation coefficient [17] of the rankings is 1.0 for all measures. Out of three million pairs, the number of pairs in the same buckets are 40,967 for set-unionability, 41,886 for sem-unionability, and 72,608 for NL-unionability. Thus, in practice, estimating unionability distributions has linear computational complexity in the number of attributes in a corpus. Following the same paradigm, to estimate table unionability distributions for various c we only consider table pairs whose attributes are bucketized together.

To make sure that goodness evaluation is not affected by distribution estimation, we perform a robustness test. We modify the calculated unionability probability scores by one standard deviation σ and calculate an interval of goodness scores. During search, whenever we need to rank attribute pairs and table pairs by goodness, we rank them by their goodness intervals.

6. EXPERIMENTAL EVALUATION

We evaluated the effectiveness and efficiency of table union search on a repository of Canadian, UK and US Open Data tables. We also compared table union search with approaches that solve relevant problems for Web tables in terms of accuracy and efficiency.

6.1 Data Preparation

In this section, we describe the datasets and pre-processing steps we took to prepare for our experimental evaluation.

6.1.1 Datasets

We have downloaded CSV-formatted tables from the Canadian, UK and US Open Data portals. In our experiments, we consider attributes with text values. Canadian Open Data consists of 11,809 tables with 40,308 text attributes, UK Open Data contains 38,348 tables with 217,445 text attributes, and US Open Data contains 165,236 tables with 25,127,735 text attributes.

6.1.2 Entity Recognition and Attribute Annotation

To generate the semantic representation of attributes, $\hat{a}$, we use YAGO [34], which is one of the largest and most current publicly available ontologies. In order to use an entity mapping technique that closely follows Zhou et al. [39], we downloaded the YAGO database (2013 version) and built a full-text search index on the entity names. To map an attribute value to entities in YAGO, we first lowercase and remove all punctuation characters, then query the full-text search index with AND query of all tokens in the attribute value. Each entity name in the search result should contain all tokens of the query. We map the data value to the top-3 entities in the search result ranked by BM25 score [13]. Based on the mapped entities, we annotate each attribute with classes from YAGO, using the MAJORITY algorithm proposed by Venetis et al. [35]. We observed that on average only 13% of values in attributes can be mapped to entities in YAGO. Using the mapped entities, we were able to map 36,826 (91%) of attributes in Canadian Open Data to classes in YAGO.

6.1.3 Word Embedding

We downloaded the word embedding database for English published by Facebook AI Research from the fastText[2] project [14]. To build embedding vectors for each attribute value, we first lowercase and remove all punctuation characters, then find the embedding vector of each token (tokens with no embedding vector in the fastText database are skipped). Following Mikolov et al., the embedding vector of an attribute value is constructed by summing the individual embedding vectors of its tokens [25]. The generated vectors form the embedding set of the attribute. We were able to generate embedding representations for 196,666 (90%) attributes of UK Open Data, 40,268 (99%) attributes of Canadian Open Data, and 17,516 (0.06%) attributes of US Open Data. The coverage of word embedding on US Open Data is surprisingly low because the majority of attributes in this corpus contain encodings that do not exist in natural language.

[2]https://github.com/facebookresearch/fastText

6.2 Implementation

Estimation of Unionability Distribution. To estimate the distribution of unionability probabilities, we applied the technique of Section 5.3 to 150,000 random Open Data table pairs with seven million attribute pairs and calculated unionability distributions for all measures. We used the 150,000 table pairs and calculated their alignments using greedy graph matching for various values of c. Our parallel implementation of this task took under three hours. Computing the goodness of a table or attribute unionability score requires evaluating Cumulative Distribution Functions (CDFs) for the score during search. To do this efficiently, during pre-processing we rank and group scores into 500 equi-depth partitions between zero and one, and estimate CDFs using the partitions. Theses CDFs and partitions are used in evaluating goodness during search. Equi-depth partitioning provides a more accurate approximation of goodness scores compared to equi-width partitioning because the unionability distributions are not uniform. In order to make sure that goodness evaluation is not affected by distribution estimation, we perform the robustness test explained in Section 5.3 with standard-deviation σ=1%.

Calculation of NL-unionability. Evaluating NL-unionability requires the calculation of mean vector and covariance matrix of the embedding vectors of each attribute. In order to improve the interactivity of search, like Venetis et al., we assume that attribute values are generated independently [35]. Therefore, we can use the variance vector of attributes' embedding vectors instead of covariance, which saves us time during data preparation and NL-unionability calculation during search.

Unionability Indices. We build all unionability indices using self-tuning LSH-forest [1] with optimal parameters calculated for threshold 0.7.

6.3 Effectiveness

We report the precision, recall and mean average precision (MAP) of table union search devising single unionability measure and the ensemble measure on a benchmark created from Open Data tables.

6.3.1 Benchmark

Since there is no available ground truth for table union search, we synthesized a benchmark[3] using tables from Canadian and UK Open Data. The ground truth provided by the benchmark allows us to evaluate the precision, recall and MAP of table union search.

Das Sarma et al. [35] consider unionable tables as results of projections, selections, or a sequence of selections and projections on a base table. Following this intuition, we start by finding base tables from Open Data. Then, we perform selection and projections to divide them into unionable tables. Various projection sizes and projected attributes give us a set of tables that are unionable on different c's.

We choose the base tables by looking at all Open Data tables in descending order of row counts and pick tables that have at least 5 text columns. This is to ensure both natural language and ontology measures are treated equally. In order to avoid using near-duplicate tables, every pair of selected tables must have at least 5 different column names. Octopus performs keyword search to find related tables, then clusters the results into groups of unionable tables [5]. Following the Octopus intuition, to add diversity and find vaguely unionable tables with base tables, we use the meta-data (table name and publisher) of each base table to perform keyword search on the meta-data of Open Data tables. For each base table, we select one of the highly ranked and non-duplicate result tables. This gives us an extended set of base tables. The added tables are about similar topics as the original base tables but are not necessarily unionable on all attributes. We manually align each search result table with the corresponding query table.

Next, we create groups of unionable tables by first issuing a projection on a random c-subset of columns of a base table, and then a selection with some limit and offset on the projected table. We make sure selection offsets on the same base table do not overlap. For every original table with n columns, we generate up to two random c-subset projections for $c = 1, 2, \ldots, n$. For each projection we generate five unionable tables by performing selection.

For every pair of benchmark tables originated from the same base table or unionable base tables found by keyword search, the ground truth alignment can be derived from matching or aligned columns, respectively. A correct alignment only happens when all and only matching or aligned columns are aligned, which means tables are aligned with the correct number of attributes (optimal c).

6.3.2 Effectiveness Measures

We report the precision and recall of top-k table union search at various k. For multiple queries, we average the precision and recall scores of all queries, respectively. We also use mean average precision (MAP) as a single-figure measure to evaluate the quality of top-k search on results ordered by exact table unionability scores. Given a set of query tables Q, suppose A_q is an ordered list of result table alignments for query q in Q. Suppose G_q is the set of unionable table alignments with q based on the ground truth. The mean average precision of Q is defined as follows [24].

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{1}{|G_{q_i}|} \sum_{g_l \in G_{q_i}} \text{Precision}(T_{g_l}, q_i) \quad (21)$$

where T_{g_l} is the set of alignments in A_{q_i} starting from the top-1 until we find table g_l and *Precision* is the precision of T_{g_l} given the ground truth of q_i. If g_l does not appear in the returned results, the *Precision* is considered as 0.

Moreover, for each measure we report the relative recall [7] of top-k results and the number of queries for which they can successfully find some unionable tables. For the unionability measures $U = \{\mathbf{U}_{\text{set}}, \mathbf{U}_{\text{sem}}, \mathbf{U}_{\text{nl}}, \mathbf{U}_{\text{ensemble}}\}$, suppose A_{U_i} is the set of search results (tables) returned by measure U_i. The relative recall of U_i with respect to all measures is defined as follows.

$$\text{Recall}_{Relative}(U_i) = \frac{|A_{U_i}|}{|\cup_{U_j \in U} A_{U_j}|} \quad (22)$$

6.3.3 Attribute and Table Unionability

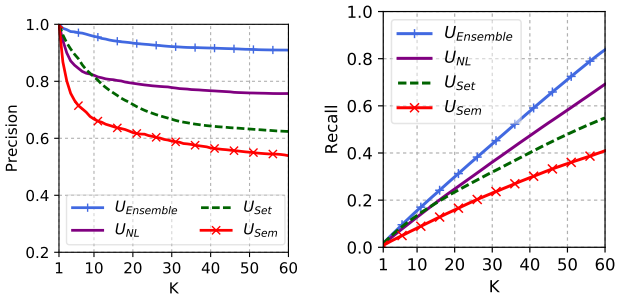
We used the benchmark of Section 6.3.1 to generate 5,000 tables, starting with 32 base tables and their unionable tables found by keyword search. All benchmark tables are indexed as described in Section 5. We randomly select 1,000 benchmark tables as queries. These queries are consistently used to evaluate the quality of search in four experiment sets. In three experiments, we only apply a single unionability measure and its corresponding index during search. The fourth set of experiments uses ensemble-unionability for search. Table 2 and Figure 5 report the precision, recall, MAP, and the result counts of top-60 table union search.

NL-unionability outperforms all single measures in precision, recall, MAP, and relative recall. This shows that NL-unionability is capturing the similarity shared by real unionable attributes. NL-unionability has slightly smaller query count than set-unionability due to the limited coverage of word embeddings on some unionable benchmark tables. On the other hand, although set-unionability is

[3] https://github.com/RJMillerLab/table-union-search-benchmark

Table 2: Accuracy and Result Counts of Search of 1,000 Queries.

Measure	Set	Sem	NL	Ensemble
MAP@60	0.7202	0.5269	0.7236	0.9249
Relative Recall	0.4867	0.4737	0.4896	0.4560
Precision	0.6239	0.5389	0.7570	0.9095
Recall	0.5489	0.4095	0.6918	0.8377
#Queries w/ Answers	995	966	986	995
#Returned Results	63,681	61,968	64,050	59,653



(a) Average Precision. **(b)** Average Recall.

Figure 5: Precision and Recall at different k over 1,000 Queries.

the most successful of all three measures in finding at least one unionable table for each query, it can misguide search when it finds value overlap in attributes of non-unionable tables. As a result, set-unionability has lower precision and recall than NL-unionability. Despite the partial coverage of the ontology, sem-unionability is sufficient to find some unionable tables for most queries. Recall that we strictly apply a 0/1 evaluation for the correctness of alignments, thus, if a measure fails to detect all unionable attributes of two tables, the result is evaluated as incorrect. Due to partial coverage, some unionable attributes cannot be detected by only considering sem-unionability which results in a drop in precision and recall. We observed that set and sem-unionability measures find some unionable tables, but NL-unionability is better at high k.

Ensemble-unionability chooses the best measure to evaluate attribute unionability. In particular for the same table, it can use different measures for different attributes, choosing the measure with best goodness (meaning the least likelihood of there being a more unionable pair based on that measure in the corpus). This results in the ensemble-unionability achieving the highest precision, recall, MAP and relative recall compared to table union search using only a single measure.

Figure 5 shows the precision and recall curves for $k = 1$ to $k = 60$. The precision of set and sem-unionability decreases as recall increases and as k increases. NL-unionability sustains precision as recall increases and can achieve the highest recall among three measures. As the high MAP of the ensemble-unionability also suggests, ensemble-unionability demonstrates the lowest decrease in precision and highest increase in recall across all k's.

6.4 Efficiency

To evaluate the efficiency of search devised by different unionability measures, we indexed 5,000 benchmark tables and used the 1,000 queries of Section 6.3.3. The index creation time for set, sem, NL, and ensemble-unionability are 12.90, 8.86, 10.76, and 10.78 seconds, respectively. Sem-unionability has the lowest indexing time because the number of attributes that have semantic domain representation is smaller than that of other types of domains due to partial coverage of ontology. Figure 6 reports the 10th and 90th percentile response times[4]. NL-unionability and ensemble-unionability achieve interactive response times, while sem-unionability has a significantly higer response time. There are two factors to the overall performance of search using a measure: (1) the access speed of the corresponding index, and (2) the pruning power of the unionability measure. We used LSH forest for set, sem and NL-unionability. Therefore, the index access speed is consistent across all measures. Figure 6 shows that NL-unionability has the most pruning power, of all single unionability measures. Due to the low coverage of the ontology on Open Data attributes, the sem-unionability of most attribute pairs are low. Recall that LSH forest starts with an optimal set of parameters based on a threshold and searches for unionable attributes. If the parameters do not lead to any results, LSH forest changes the parameters to search for attributes with lower unionability, until it finds some. Sem-unionability distribution has a long tail, this results in the corresponding index returning a lot of candidate attributes when no highly unionable attribute exists. We observed that only 3% of attribute pairs in Open Data have sem-unionability greater than 0.5.

Ensemble-unionability probes all indices in parallel. It involves computing all three measures for candidate attributes returned by any of the indices. In spite of this overhead, the combined pruning power of the three unionability measures results in the ensemble measure being only slightly more expensive than NL-unionability (in particular, the large number of candidates for sem-unionability can often quickly be pruned). As shown in Figure 6, all unionability measures scale smoothly with respect to k.

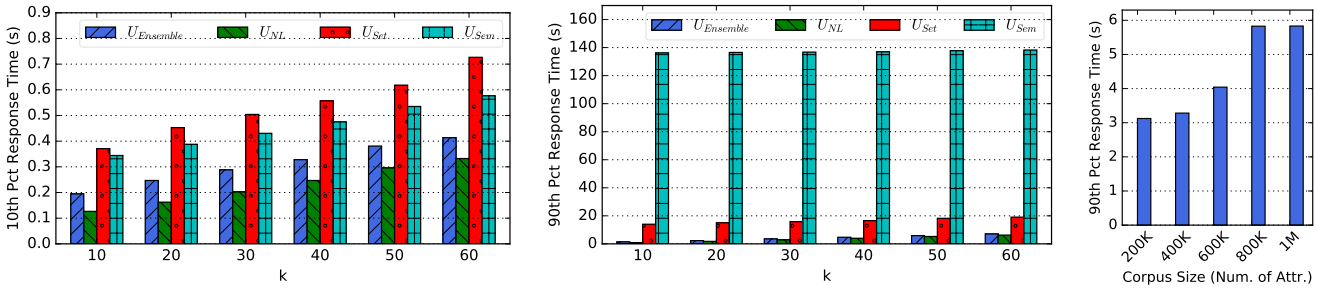
We further evaluated the response time of ensemble-unionability with respect to different Open Data corpus sizes. The index creation time for corpora with 200K, 400K, 600K, 800K and 1M attributes of Open Data are 142.25, 250.08, 333.97, 346.98, and 412.25 seconds respectively. Figure 6c reports the 90th percentile of the response time of top-10 table union search of 1,000 queries using ensemble-unionability on repositories containing 200K to over 1M attributes. Ensemble-unionability consistently achieves interactive speeds.

6.5 Comparison with Existing Approaches

We compared table union search using ensemble-unionability with approaches that address similar problems for Web tables. We were unable to run the open source implementations of some approaches on our full benchmark, so we generated a smaller benchmark consisting of 1,300 tables originating from 10 base tables. Figure 7 reports the precision and recall of top-60 results of these frameworks and table union search. The overall run time of top-60 table union search on the benchmark is less than four minutes.

Octopus. Octopus clusters the results of keyword search on a repository of Web tables using two text-based attribute similarity measures: (1) column-text-cluster, and (2) size-cluster [5]. Column-text-cluster uses the tf-idf Cosine similarity of attributes while size-cluster uses the difference between the mean string length of attributes as a signal for structural similarity. The overall similarity of two tables using each measure is the score of the matching that maximizes the sum of attribute similarity scores. We consider an Octopus alignment to be correct if it matches at least all attributes in a ground truth alignment. Although column-text-cluster outperforms size-cluster, it can only detect similarity when value-overlap exists. Our table union search algorithm which uses semantic and natural language unionability in addition to text overlap-based unionability greatly outperforms both Octopus measures (Figure 7). The overall run time of Octopus on the benchmark

[4]The 10th and 90th percentile of x means 10% and 90% of queries have response times faster than x, respectively.



(a) **(b)** **(c)**

Figure 6: (a), (b) The 10th and 90th percentile response times, respectively, of table union search for various k on 5,000 Benchmark Tables. (c) The 90th percentile response times of top-10 table union search using $\mathbf{U}_{\text{ensemble}}$ for various Open Data corpus sizes.

is over one hour. This is due to the batch processing of tables by Octopus which has complexity in the number of tables in the corpus multiplied by the number of queries.

Web Table Stitching. Lehmberg and Bizer applied schema matching techniques to the problem of stitching Web tables [19]. Given two tables, this framework generates matching correspondences between their attributes. The label-based matching, which matches attributes with identical names, is the most effective table stitching technique. However, since most Open Data tables have missing or meaningless attribute names, label-based matching does not apply to Open Data. We considered their second best matcher, namely the value-based matcher. This matcher measures the overlap of the domains of two attributes followed by a matching refinement stage, during which inconsistent matchings are removed and new matchings are inferred via transitivity. To compare stitching with table union search, we apply maximum graph matching to generate the best alignment for two tables using the generated correspondences. The matching of two tables is considered to be correct if the correspondences in the matching include at least all attribute pairs in the ground truth alignment. Figure 7 shows that table union greatly outperforms Web Table Stitching, since it applies an ensemble of semantic and natural language unionability in addition to a text overlap-based unionability (set-unionability). The overall runtime of Web Table Stitching using the open source implementation of the framework on the benchmark is 19 minutes.

Entity complement. Das Sarma et al. defines two Web tables as entity complements if they contain information about related sets of entities (entity consistency and schema consistency) and if a candidate table expands a query with new entities (entity expansion) [31]. To evaluate these measures, entity complement uses the signals mined from high coverage ontologies curated from all data on the Web. In our experiments, we consider a publicly available ontology (YAGO [34]). Since the entity complement does not provide a way of evaluating the unionability of attributes or of aligning attributes, we only evaluate its precision and recall on the problem of finding entity-complement tables. Entity complement achieves a precision of 0.6252 and recall of 0.6378 on the benchmark. This result cannot be directly compared to the accuracy results of alignments produced by the other frameworks (and reported in Figure 7). Since the entity complement relies heavily on a high coverage ontology for evaluating the entity and schema consistency, and for evaluating entity expansion, the low coverage of publicly available ontologies on Open Data negatively affects its accuracy (we observed that only 54.6% of values in subject attributes of the benchmark tables are mapped to an entity in YAGO). Table union search has much better accuracy than the entity-complement framework even in the harder task of finding and aligning unionable tables. The overall runtime of entity complement was 15 hours. Although entity complement enforces a set of heuristics for filtering candidates during batch processing, we observed that the bottleneck is in the value-based schema consistency evaluation.

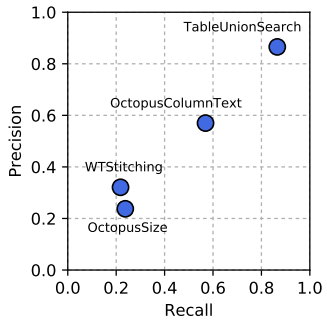


Figure 7: Comparison of Table Union Search with Octopus (Column-Text-Cluster and Size-Cluster) and Web Table Stitching.

7. CONCLUSION AND FUTURE WORK

We defined *table union search* problem and presented an efficient approximate solution. We defined three probabilistic measures for evaluating the unionability of attributes – the hypothesis that two attributes are drawn from the same domain. We have used these measures to develop a scalable table search framework. We proposed a novel distribution-aware way of deciding which measure to use for an attribute pair and what the optimal number of unionable attributes in two tables are. Our table union search achieves response times typically much less than five seconds on a real corpus of over a million Open Data attributes. We showed that our table search achieves greater accuracy, at much lower response time, than any of the existing related table union and stitching approaches.

Our search algorithm assumes independence of the unionability of attribute pairs when aligning two tables. In future work, it would be interesting to take into account the correlations between attributes in searching for unionable tables. While our empirical evaluation uses Open Data (and a new Open Data benchmark that we have made public), our techniques show great promise for helping to manage both public and private data lakes. Our framework uses publicly available word embedding vectors to evaluate natural language unionability. We plan to investigate the use of domain-specific word embeddings trained on Open Data or other data lakes.

8. ACKNOWLEDGMENT

This work was partially supported by NSERC.

9. REFERENCES

[1] M. Bawa, T. Condie, and P. Ganesan. LSH forest: self-tuning indexes for similarity search. In *Proceedings of the 14th international conference on World Wide Web*, pages 651–660, 2005.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[3] A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pages 21–30, 1997.

[4] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: Exploring the power of tables on the web. *PVLDB*, 1(1):538–549, 2008.

[5] M. J. Cafarella, A. Y. Halevy, and N. Khoussainova. Data integration for the relational web. *PVLDB*, 2(1):1090–1101, 2009.

[6] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing*, pages 380–388, 2002.

[7] S. J. Clarke and P. Willett. Estimating the recall performance of web search engines. *Aslib Proceedings*, 49(7):184–189, 1997.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[9] CrowdFlower. 2017 data scientist report. `https://visit.crowdflower.com/WC-2017-Data-Science-Report_LP.html`, 2017.

[10] O. Hassanzadeh, K. Q. Pu, S. H. Yeganeh, R. J. Miller, L. Popa, M. A. Hernández, and H. Ho. Discovering linkage points over web data. *PVLDB*, 6(6):444–456, 2013.

[11] B. He and K. C. Chang. Statistical schema matching across web query interfaces. In *SIGMOD*, pages 217–228, 2003.

[12] H. Hotelling. The generalization of student's ratio. *The Annals of Mathematical Statistics*, 2(3):360–378, 08 1931.

[13] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments - part 1. *Information Processing and Management*, 36(6):779–808, 2000.

[14] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *ACL*, 2017.

[15] J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *SIGMOD*, pages 205–216, 2003.

[16] S. Kuzi, A. Shtok, and O. Kurland. Query expansion using word embeddings. In *CIKM*, pages 1929–1932, 2016.

[17] A. Lehman. *JMP for Basic Univariate and Multivariate Statistics: A Step-by-step Guide*. SAS Institute, 2005.

[18] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[19] O. Lehmberg and C. Bizer. Stitching web tables for improving matching quality. *PVLDB*, 10(11):1502–1513, 2017.

[20] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets, 2nd Ed*. Cambridge University Press, 2014.

[21] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *PVLDB*, 3(1):1338–1347, 2010.

[22] X. Ling, A. Y. Halevy, F. Wu, and C. Yu. Synthesizing union tables from the web. In *IJCAI*, pages 2677–2683, 2013.

[23] L. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.

[24] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NISP*, pages 3111–3119, 2013.

[26] A. Nandi and P. A. Bernstein. HAMSTER: using search clicklogs for schema and taxonomy matching. *PVLDB*, 2(1):181–192, 2009.

[27] R. Pimplikar and S. Sarawagi. Answering table queries on the web using column keywords. *PVLDB*, 5(10):908–919, 2012.

[28] E. Rahm. Towards large-scale schema and ontology matching. In *Schema Matching and Mapping*, pages 3–27. 2011.

[29] J. A. Rice. *Mathematical Statistics and Data Analysis*. 2006.

[30] D. Ritze, O. Lehmberg, and C. Bizer. Matching HTML tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence*, pages 10:1–10:6, 2015.

[31] A. D. Sarma, L. Fang, N. Gupta, A. Y. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu. Finding related tables. In *SIGMOD*, pages 817–828, 2012.

[32] R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, pages 129–136, 2011.

[33] W. Su, J. Wang, and F. H. Lochovsky. Holistic schema matching for web query interfaces. In *EDBT*, pages 77–94, 2006.

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

[35] P. Venetis, A. Y. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. *PVLDB*, 4(9):528–538, 2011.

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

[37] H. Zamani and W. B. Croft. Estimating embedding vectors for queries. In *Proceedings of the International Conference on the Theory of Information Retrieval*, pages 123–132, 2016.

[38] S. Zhang, P. Mork, O. Bodenreider, and P. A. Bernstein. Comparing two approaches for aligning representations of anatomy. *Artificial Intelligence in Medicine*, 39(3):227–236, 2007.

[39] X. Zhou, X. Zhang, and X. Hu. Maxmatcher: Biological concept extraction using approximate dictionary lookup. In *PRICAI: Trends in Artificial Intelligence*, pages 1145–1149, 2006.

[40] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller. LSH ensemble: Internet-scale domain search. *PVLDB*, 9(12):1185–1196, 2016.