InfoGather+: Semantic Matching and Annotation of Numeric and Time-Varying Attributes in Web Tables

Meihui Zhang*
National University of Singapore
mhzhang@comp.nus.edu.sg

Kaushik Chakrabarti
Microsoft Research
kaushik@microsoft.com

ABSTRACT

Users often need to gather information about "entities" of interest. Recent efforts try to automate this task by leveraging the vast corpus of HTML tables; this is referred to as "entity augmentation". The accuracy of entity augmentation critically depends on semantic relationships between web tables as well as semantic labels of those tables. Current techniques work well for string-valued and static attributes but perform poorly for numeric and time-varying attributes.

In this paper, we first build a semantic graph that (i) labels columns with unit, scale and timestamp information and (ii) computes semantic matches between columns even when the same numeric attribute is expressed in different units or scales. Second, we develop a novel entity augmentation API suited for numeric and time-varying attributes that leverages the semantic graph. Building the graph is challenging as such label information is often missing from the column headers. Our key insight is to leverage the wealth of tables on the web and infer label information from semantically matching columns of other web tables; this complements "local" extraction from column headers. However, this creates an interdependence between labels and semantic matches; we address this challenge by representing the task as a probabilistic graphical model that jointly discovers labels and semantic matches over all columns. Our experiments on real-life datasets show that (i) our semantic graph contains higher quality labels and semantic matches and (ii) entity augmentation based on the above graph has significantly higher precision and recall compared with the state-of-the-art.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services, Web-based services

Keywords

Web Table, Semantic Matching

*Work done while visiting Microsoft Research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGMOD'13, June 22–27, 2013, New York, New York, USA.
Copyright 2013 ACM 978-1-4503-2037-5/13/06 ...$15.00.

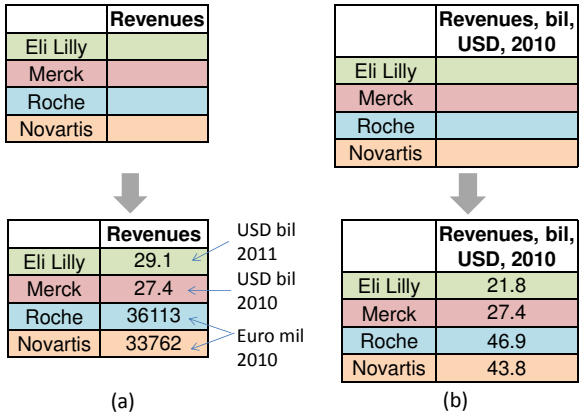


Figure 1: (a) Existing entity augmentation approach (b) Proposed entity augmentation operation for numeric and time-varying attributes

1. INTRODUCTION

Users often need to gather information about "entities" of interest [5, 21, 16]. For example, an analyst in an enterprise might want to gather revenue and market capitalization information of its customers in a particular sector (say, pharmaceutical sector) to predict its sales for that sector. Or a student might want to gather to information about colleges, such as their tuition fees, number of enrolled students, acceptance rates and average SAT scores, in order to choose the college to go to.

Such information gathering tasks are extremely labor-intensive today. The Web contains a vast corpus of HTML tables. In this paper, we focus on relational HTML tables where each row corresponds to an entity and each column corresponds to an attribute [5, 21]. In many cases, they contain the information sought by these tasks [5, 21, 19, 14, 12]. The desired information is typically scattered among various tables. To automate information gathering, researchers have recently proposed the *"entity augmentation"* operation (EA) [5, 21, 14](referred to as EXTEND in [5]). Given the names of a set of entities and a few keywords describing the name of an attribute, the entity augmentation operation leverages the vast corpus of HTML tables to automatically "fill in" the values of the specified entities on the specified attribute. Figure 1(a) shows an example of an EA query where the entities are 4 pharmaceutical companies and the keyword is 'revenues'.

To support EA, we first find tables that "match" with the query table, i.e., contains some of the entities and an attribute whose name matches the keywords. We then consolidate the matching tables to fill in the desired values.

For high precision and recall, it is critical that the match-

ing tables are *semantically consistent*. Matching the attribute names with the keywords using IR techniques does not ensure this. For example, the query shown in Figure 1(a) might match with tables containing 2010 revenues as well as those containing 2011 revenues, those containing revenues in billions of USD as well as those containing revenues in millions of Euros (because they all contain the keyword 'revenues'). Consolidating values from such different tables without understanding the semantic relationships between them leads to erroneous augmentation.

Baseline Approach: The baseline approach to compute semantically consistent matches is to build a *semantic matching graph over web tables* [21]. Suppose each web table is a binary relation with the first column corresponding to the entity name and the second to an attribute of the entity (referred to as entity-attribute binary (EAB) relations). Each web table is a node in the graph. There is an edge between two nodes iff (i) the first columns of the two web tables contain the same type of entities and (ii) the second columns refer to the same attribute of those entities. The edges are computed using schema matching techniques [21].

The accuracy of entity augmentation critically depends on the accuracy of the semantic graph. While the baseline technique works well for string-valued and static attributes, it is inadequate for *numeric* and *time-varying* attributes.

EXAMPLE 1. *Consider the 5 web tables shown in Figure 2. The baseline technique identifies the 3 edges shown using solid, black lines (referred to as simple(S) edges as they capture simple, 1:1 mappings): between T_2 and T_3, T_1 and T_4 and T_4 and T_5. Consider the query table Q in Figure 1(a). The baseline technique matches with all 5 tables. The result of entity augmentation is shown in Figure 1(a). For Eli Lilly, T_4 and T_5 provides the value* 29.1 *while T_1 provides* 21.8*; hence, it selects* 29.1 *(based on aggregate score). For Merck, T_1 provides* 27.4*, T_5 provides* 45.9 *and T_2 provides* 21091*; say, it selects* 27.4 *based on matching score. For Roche and Novartis, T_3 provides the values* 36113 *and* 33762 *respectively. The result is undesirable: the value for Eli Lilly is in USD billion and from 2011, the one for Merck is also in USD billion but from 2010 and the ones for Roche and Novartis are in Euro million and from 2010.*

Why does the baseline approach produce semantically inconsistent results? First, we observe that *a large fraction attributes in web tables are numeric* and the same semantic attribute occurs in a *wide variety of units and scales across different web tables*. For example, in Figure 2, T_1 has 2010 revenue in billions of USD while T_2 and T_3 have it in millions of Euros. The baseline approach has no knowledge of unit and scale; without this knowledge, it is impossible to produce semantically consistent results. Second, many numeric attributes are time-varying in nature: different tables, possibly created at different times, contain values of the same attribute at different points of time. For example, T_1 has the revenue information for 2010 while T_4 and T_5 have the same information for 2011. The baseline approach has no knowledge of time; without this, it is again impossible to produce semantically consistent results. Finally, the baseline approach fails to detect relationships where the same semantic attribute is expressed in different units or scales. For example, it fails to detect that T_1, T_2 and T_3 contain the same semantic attribute (2010 revenue) and the values can be converted from one to another. Without this knowledge, it is impossible convert the values in the matching tables into the desired scale and unit.

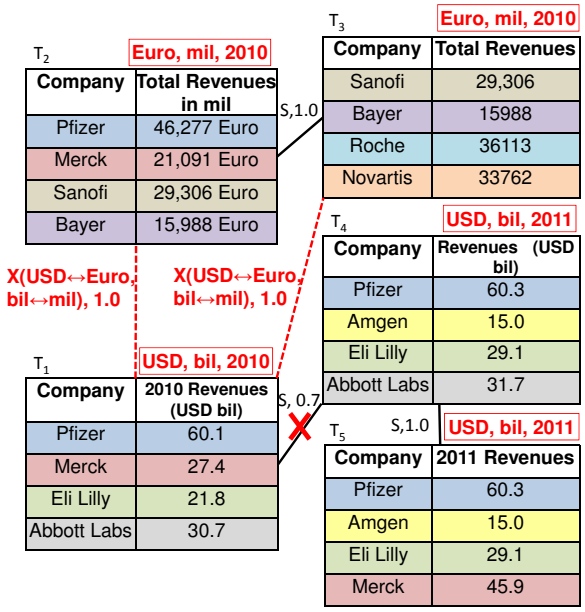


Figure 2: Example of Semantic Graph over Web Tables

Main insights and contributions: In this paper, we build a semantic graph over web tables suited for numeric and time-varying attributes. It consists of:

(i) *Semantic labels*: Annotates each attribute column with unit, scale and timestamp (referred to as *"semantic labels"*). In this paper, we focus on year as the timestamp.[1] The semantic labels are shown in red inside boxes in Figure 2).

(ii) *Semantic matches*: Computes semantic matches between columns representing the same semantic attribute, even if they are expressed in different units and scales. For example, in addition to the S edges, the graph should contain edges between T_1 and T_2 and between T_1 and T_3 (shown using dashed, red lines) in Figure 2. We refer to these as X edges (X referring to transformations/conversions). We assume a set of pre-defined *conversion rules*. An X edge between a pair of tables T and T' is associated with a set of conversion rules which together convert the values between $T.B$ and $T'.B$. For example, the X edge between T_1 and T_3 is associated with two conversion rules: $Euro = 1.3 \times USD$ and $bil = 1000 \times mil$.

There are several technical challenges in building such a graph. First, identifying the semantic labels as well as the X edges is hard as the unit and scale information is often missing in the column headers or values [19, 14]. For example, table T_3 does not specify that the information is in millions of Euros. *Our key insight is to leverage the wealth of tables on the web: even if the column header does not contain the label information, the semantically matching columns of other web tables might contain the label information (in their column headers). We can "propagate" those labels to this table.* For example, T_3 semantically matches with T_2; assuming we can extract T_2's labels *mil* and *Euro* "locally" from its column header and column values, we can propagate them to T_3. Although this shows propagation over an S edge, such propagation can occur even over X edges. However, this creates an interdependence between labels and semantic matches: we need semantic matches to compute the labels and we need the labels to compute the matches. We address this challenge by *representing the task as a*

[1]Our approach can handle other granularities of timestamp like quarter, month, day, etc. We focus on year for simplicity of explanation.

probabilistic graphical model (Markov random field) that simultaneously discovers labels and semantic matches over all columns. Second, making such inferences can lead to inconsistencies. For example, the same table might be assigned both *Euro* and *USD* as labels. We address this challenge by defining hard constraints to control such inconsistent labels and integrating them into the inference algorithm. Finally, schema matching techniques can compute spurious matches. For example, the values of an attribute (say, revenues) may not change significantly from one year to another; this can lead to spurious matches. In Figure 2, the revenues of Pfizer and Abbott Labs did not change significantly from 2010 to 2011; this leads to a spurious semantic match between T_1 and T_4. This leads to semantically inconsistent results as shown in Example 2. Our main insight is to leverage the discovered labels to eliminate such spurious edges. For example, we can infer that the edge between T_1 and T_4 is spurious based on the discovered labels 2010 and 2011 respectively (hence marked with a red cross). We encode this knowledge as hard constraints into our graphical model.

We have built the InfoGather+ system based on the above insights. Our contributions can be summarized as follows:

• We propose a novel technique based on probabilistic graphical models to discover the semantic labels of columns and semantic matches between columns over all web tables collectively instead of individually. Our model elegantly combines diverse signals such as "local" extraction of labels from the column headers and values, semantic matches computed using traditional schema matching techniques and label propagation. We develop efficient algorithms to solve the joint discovery task.

• We present a novel entity augmentation API suited for numeric and time-varying attributes. Specifically, we allow users to specify unit, scale and time information to unambiguously specify the augmentation task. An example of the API and the desired output (based on the graph shown in Figure 2) is shown in Figure 1(b). We develop novel query processing algorithms for the new entity augmentation operation.

• We perform extensive experiments on three real-life datasets of web tables. Our experiments show that the graph produced by our graphical model has significantly higher quality in terms of semantic labels and semantic matches compared with the baseline approach. Furthermore, entity augmentation based on the above graph has significantly higher precision and recall compared with the baseline approach.

The rest of the paper is organized as follows. We formulate the semantic graph building problem in Section 2. Section 3 describes the system architecture. We present our model and inference algorithms in Section 4. We describe the entity augmentation API and query processing algorithm in Section 5. Experiments, related work and conclusions appear in Section 6, 7 and 8.

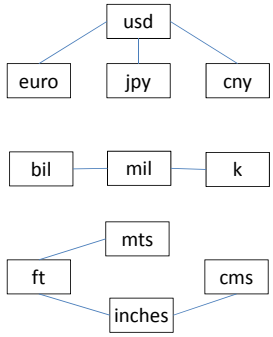
2. PROBLEM DEFINITION

2.1 Data Model

We assume that each web table is an entity-attribute binary (EAB) relation, i.e., a web table $T \in \mathcal{T}$ is of the form $T(K, B)$ where K denotes the entity name attribute and B is an attribute of the entity. Figure 2 shows five web tables (T_1,T_2,T_3,T_4,T_5) satisfying the EAB property. We assume that each web table T has (i) a column header H_T of the attribute column $T.B$ (e.g., H_{T_1} for table T_1 in Figure 2 is "2010 Revenues (USD bil)"). H_T can be empty if the table has no column headers. (ii) context information C_T including header rows that describes the table (that spans across all the columns), caption of the table, text surrounding the table and the url and title of the web page from which it was extracted. In our system, we distinguish between the different context fields. For simplicity, we ignore these distinctions in this paper.

Rule Id	LHS	Conversion Factor (θ)	RHS
r1	euro	1.3	usd
r2	bil	1000	mil
r3	usd	6.2	cny
r4	mil	1000	k
r5	usd	80.2	jpy
r6	mts	3.28	ft
r7	ft	12	inches
r8	inches	2.54	cms

(a)



(b)

Figure 3: (a)Conversion rules (b) Mutex groups

In practice, not all web tables are EAB relations. Most such tables have a "subject column"; this column contains the names of the entities. All other columns are attributes of these entities [19]. Furthermore, there are effective heuristics to detect the subject column [19]. We can therefore split the $n-$ary table into $(n-1)$ EAB relations: the subject column with each of the other $(n-1)$ columns.

2.2 Conversion Rules

We assume that the set of conversion rules are known upfront. They are specified by the system administrator (or domain experts) using a simple, rule specification language. Each rule has 3 components: a left-hand side (LHS), a conversion factor (θ) and a right-hand side (RHS). The LHS and RHS are strings describing units and scales. For example, for the rule $Euro = 1.3 \times USD$, the LHS is *Euro*, θ is 1.3 and the RHS is USD. Figure 3(a) shows more examples of rules.

The same unit and scale can be referred to in several ways. For example, USD is also referred to as \$ and *US Dollar*; *mil* is also referred to as *mln*, *million* and *millions*. We assume there is a canonical string for every unit and scale; the rules are specified using the canonical strings. Furthermore, we assume that all the synonyms are known so that their occurrences can be detected in column headers and column values.

For simplicity, we assume if a rule $a = \theta \times b$ is present, the reverse rule $b = \frac{1}{\theta} \times a$ is also present in the rule database (not shown in Figure 3(a) to avoid clutter). Finally, our system can handle ranges as the conversion factor in order to capture fluctuating conversion factors. For example, one might specify the conversion factor between Euro and USD to be anywhere between 1.2 and 1.3. For simplicity, we focus on a single number as the conversion factor. Note that the rules can change with time: the system administrator can insert new rules, delete existing rules or modify the LHS, conversion factor or RHS of existing rules.

2.3 Problem Statement

Given the corpus of web tables and a set of conversion rules, the goal is to build a semantic graph G over web tables that enables us to perform accurate entity augmentation. Each web table $T \in \mathcal{T}$ is a node in G. The ideal semantic

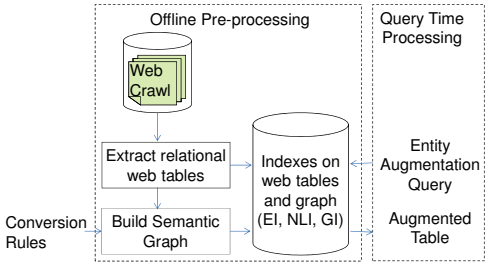


Figure 4: InfoGather+ system architecture

graph G should contain:
(i) **S edges**: between two nodes $T(K, B)$ and $T(K', B')$ iff $T.K$ and $T.K'$ refer to the same type of entities and $T.B$ and $T.B'$ refer to the same attribute of those entities, expressed in the same unit and scale and reflect information of the same points of time (i.e., same year).
(ii) **X edges**: between two nodes $T(K, B)$ and $T(K', B')$ iff $T.K$ and $T.K'$ refer to the same type of entities and $T.B$ and $T.B'$ refer to the same attribute of those entities (at the same points of time) but expressed in different units and scales. Each X edge is associated with a set of conversion rules which converts the values from $T.B$ to $T'.B'$. Since the reverse rule is present for each forward rule, for every X edge from T to T', there is an X edge with an equivalent set of rules from T' to T. Hence, we ignore the directionality of X edges in the discussions except when necessary.
(iii) **Semantic labels**: for unit, scale and year at each node. We distinguish between the unit and scale labels (*SU labels* in short) and *year labels* as a node can be assigned multiple SU labels but only one year label.

3. SYSTEM ARCHITECTURE

Figure 4 shows the architecture of the INFOGATHER+ system. It has two components: offline preprocessing and query time processing.

Offline preprocessing: There are 3 main steps in this component:
• *Extract HTML tables* from the web crawl and distinguish the relational tables from other types of tables, such as layout tables and attribute-value tables. Our approach is similar to the one in [7]; we do not discuss this further as it is not the focus of this paper.
• *Build the semantic graph.* This is the focus of the paper and is discussed in detail in Section 4.
• *Build indexes on the web tables and the graph* for efficient query time processing. We build 3 indexes: (i) An inverted index on entities (EI). Given a query table Q, EI(Q) returns the set of web tables (along with scores) that contains at least one of the query entities. (ii) An inverted index on column names and semantic labels (NLI): Given a query table Q, NLI(Q) returns the set of tables (along with scores) whose column headers contain the query keywords and/or the set whose semantic labels match with the query labels. (iii) An index on graph edges (GI): Given a web table T, $GI(T)$ returns the set of tables that are connected to T in the semantic graph along with a score indicating the strength of the matching relationship.

Our offline processing needs to scale to hundred of millions of tables. Steps 1 and 3 are easily parallelizable (e.g., using the MapReduce framework). We discuss scalable implementation of Step 2 in Section 4.

Query time processing: We focus on entity augmentation queries in this paper. We can support other classes of queries like augmentation by example [21], attribute discovery [21] and search by column keywords [14] using this architecture. There are two main steps:
• *Identify matching tables and edges*: along with their scores for the query. We leverage the EI, NLI and GI indexes for this purpose.
• *Fill-in values*: For each query entity, we then collect the corresponding values in these matching tables along with the scores, convert to the desired unit and scale, aggregate the scores for each value and select the one with the highest aggregate score.

4. SEMANTIC GRAPH CONSTRUCTION

One option is to follow a *"staged" approach*: start with the semantic graph proposed in the baseline approach [21] (which contains only S edges), then add semantic labels and finally add X edges as follows:
• *Semantic labels*: Let $\mathcal{L}$ denote the set of scale and unit descriptor strings that appear in the LHSs and RHSs of the conversion rules. Formally,

$$\mathcal{L} = \bigcup_{r \in \mathcal{R}} r.LHS \cup \bigcup_{r \in \mathcal{R}} r.RHS$$

where $\mathcal{R}$ denotes the set of conversion rules and $r.LHS$ ($r.RHS$) the scale or unit descriptor constituting LHS (RHS) of rule r. We assume that we know the synonyms for each scale or unit descriptor $l \in \mathcal{L}$. We annotate a web table T with a label $l \in \mathcal{L}$ if either l or one of its synonyms occur in either the column header H_T of $T.B$ or column values of $T.B$. For example, we annotate T_1 in Figure 2 with $\{USD, bil\}$ and T_2 with $\{Euro, mil\}$.

Let $\mathcal{Y}$ denote the set of year strings; we use all year strings from "1900" to "2050". We annotate a web table T with $y \in \mathcal{Y}$ if the y occurs in either in the column header H_T of $T.B$ or in the context C_T of T. For example, we annotate T_1 in Figure 2 with 2010. We refer to this approach as *"local extractions"* as the labels are extracted locally from this table.
• *X edges*: After we have added the semantic annotations as described above, we add an X edge between T and T' associated with the set $R \subseteq \mathcal{R}$ of conversion rules iff (i) the set L_T of labels of T contains LHS of each rule $r \in R$, i.e., $\bigcup_{r \in R} r.LHS \subseteq L_T$ (ii) the set L'_T of labels of T' contains RHS of each rule $r \in R$, i.e., $\bigcup_{r \in R} r.RHS \subseteq L'_T$ and (iii) the values of the common entities in the two tables can be converted from one to the other by multiplying with the product of the conversion factors, i.e., $Sim_X(T, T', R) > \delta$ where $Sim_X(T, T', R)$ denotes the fraction of common entities that can be converted using the product of the conversion factors and δ is a threshold. Formally,

$$Sim_X(T, T, R) = \frac{|(t, t') \in T \times T' \text{ s.t. } t.A = t'.A \wedge \frac{t'.B}{t.B} \sim \Pi_{r \in R} r.\theta|}{|(t, t') \in T \times T' \text{ s.t. } t.A = t'.A|}$$

For example, we add an X edge between T_1 and T_2 corresponding to conversion rules $R = \{USD = 0.77 \times Euro, bil = 1000 \times mil\}$ because (i) T_1 has labels $\{USD, bil\}$ (ii) T_2 has labels $\{Euro, mil\}$ and (iii) the values of the common entities (i.e., Pfizer and Merck) satisfy the product of the conversion factors, i.e., $\frac{46277}{60.1} \sim (0.77 \times 1000)$ for Pfizer and $\frac{21091}{27.4} \sim (0.77 \times 1000)$ for Merck. So, $Sim_X(T_1, T_2, R) = 1.0$. Note that we allow slight variations in values while checking the convertibilities.

This approach suffers from two problems:
• *Low precision*: Text in the context fields as well as column

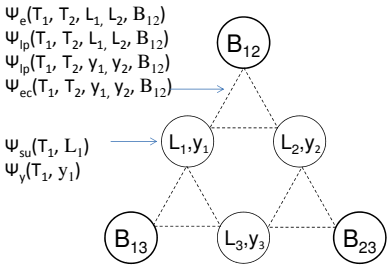


Figure 5: Graphical model for matching and annotating 3 tables T_1, T_2 and T_3

headers can be noisy and ambiguous. Annotations based on only local extraction might lead to incorrect labels.

• *Low coverage*: Often, unit, scale and year information is missing in the column headers, values and context. For example, T_3 does not contain this information. So, the staged approach fails to annotate T_3 with the unit, scale and year information; consequently, it fails to detect the two X edges (corresponding to conversion rules R_1 and R_2 listed in Figure 3) between T_1 and T_3. This is exacerbated by the fact that many tables do not have column headers and/or context information.

Our main insight is that we can improve the precision and coverage if we compute the labels not just using local extractions but also using labels of semantically matching columns. This creates an interdependence between labels and semantic matches: we need semantic matches to compute the labels and we need the labels to compute the matches. We propose a global approach that *collectively computes all the labels and matches and combines the diverse signals.* We present an elegant problem formulation based on probabilistic graph models (Markov random field). We first provide a brief overview of graphical models and then present our formulation.

4.1 Probabilistic Graphical Models

We use undirected graphical models in this paper, referred to as Markov networks or Markov random fields. A graphical model represents the joint distribution over a set of random variables $\mathcal{X} = \{X_1, \cdots, X_n\}$ where each X_i can take values from the space of labels [11]. The model represents each element of $\mathcal{X}$ as a node in a graph G and captures the dependencies between them with a set of cliques of G.

We identify the cliques and define a potential function $\psi(C, X_C) \rightarrow \mathbb{R}$ for each clique. $\psi(C, X_C)$ captures the compatibility of the labels X_C assigned to the variable subset C. We use two kinds of potentials: node potentials $\psi(i, X_i)$ defined on the label X_i of a single node i and edge potentials $\psi(i, j, X_i, X_j)$ defined over edge (i, j) in G and labels (X_i, X_j) assigned to the two nodes it connects. The overall probability distribution is the normalized product of all of the potentials. In logarithmic representation,

$$P(X_1, \cdots, X_n) \propto \exp(\sum_i \psi(i, X_i) + \sum_i \sum_j \psi(i, j, X_i, X_j))$$

The inference problem is to find $\arg\max_{\mathcal{X}} P(X_1, \cdots, X_n)$, the most likely joint assignment of labels to variables.

We describe how we model semantic annotation and matching task as a graphical model by defining the random variables, node and edge potentials, the overall objective and finally the inference algorithm.

4.2 Variables

For every table T, we associate two random variables: L_T to denote the set of SU labels and y_T to denote the year label. There can be multiple SU labels assigned to T; hence, L_T can take a value from the set $\mathcal{P}(\mathcal{L})$ where $\mathcal{P}(\mathcal{S})$ denotes the powerset of any set $\mathcal{S}$. Recall $\mathcal{L}$ denotes the set of scale and unit descriptor strings that appear in the LHSs and RHSs of the conversion rules. On the other hand, there can be at most one year label assigned to T; hence, y_T can take a value from the set $\mathcal{Y} \cup \{NA\}$ (NA denotes no year). For every pair of tables T, T', we associate a random variable $B_{TT'}$ denoting the semantic match between T and T'. There can be either an X edge (represented by a set of one or more conversion rules) or an S edge or no edge at all; hence $B_{TT'}$ can take a value from the set $\{\mathcal{P}(\mathcal{R}) - \{\phi\}\} \cup \{S, NA\}$. Recall $\mathcal{R}$ denotes the set of conversion rules.

4.3 Node and Edge Potentials

We define node and clique potentials to combine diverse signals like local extraction of labels from the column headers and values, semantic matches computed using traditional schema matching techniques and label propagation.

4.3.1 Local extraction of SU labels

One important signal in assigning SU labels is the local extraction technique described in the staged approach, i.e., we assign an SU label $l \in \mathcal{L}$ to web table T iff either l or one of its synonyms occur in either the column header H_T of $T.B$ or column values of $T.B$. We define two features for this purpose. We define a binary feature function $f_H(T, l)$ which is set to 1 if H_T contains either the label l or a synonym of l, and 0 otherwise. For column values, we extract the strings that precede and follow the numeric values in column $T.B$. Let $PFStrings(T)$ denote the set of strings that *consistently* precede or follow the values across the entire column (e.g., in more than 80% of the rows). For example, in Figure 2, $PFStrings(T_2) = \{Euro\}$. We define a binary feature function $f_V(T, l)$ which is set to 1 if there exists a $s \in PFStrings(T)$ such that s either contains l or a synonym of l, and 0 otherwise. A set L_T of labels is a good assignment if either $f_H(T, l)$ or $f_V(T, l)$ is 1 for all or most labels in it. We define the node potential $\psi_{su}(T, L_T)$ as follows:

$$\psi_{su}(T, L_T) = \frac{\sum_{l \in L_T} \max(f_H(T, l), f_V(T, l))}{|L_T|}$$

SU labeling, especially with label propagation, can lead to inconsistencies. For example, the same table might be assigned both $Euro$ and USD as labels. We define hard constraints to control such inconsistent labels and integrate them into our node potential $\psi_{su}(T, L_T)$. We define a set of mutually exclusive groups (referred to as "mutex groups") such that any table can be assigned at most one label from a mutex group. Two strings l and l' are in the same mutex group if l is connected to l' via a chain of one or more conversion rules $l = \theta_1 \times p_1$, $p_1 = \theta_2 \times p_2$, $\cdots$, $p_{n-1} = \theta_n \times l'$. Suppose the system has the 8 conversion rules shown in Figure 3(a). The mutex groups for the 8 rules are shown in Figure 3(b). The mutex groups can be computed simply as follows. Construct a graph with each label $l \in \mathcal{L}$ as a node. Insert an edge between two nodes if there is a rule with the two strings as its LHS and RHS. Compute all the connected components of the graph. Each connected component corresponds to a mutex group. Let $Mutex(l, l')$ denote a binary variable which is true if there exists a mutex group containing both l and l' and false otherwise. We modify $\psi_{su}(T, L_T)$ to disallow inconsistent labeling by taking large

negative values for inconsistent labels. The final node potential $\psi_{su}(T, L_T)$ is:

$$\begin{aligned}\psi_{su}(T, L_T) &= -\infty \text{ if } l, l' \in L_T \text{ s.t. } Mutex(l, l') \text{ is true} \\ &= \frac{\sum_{l \in L_T} \max(f_H(T, l), f_V(T, l))}{|L_T|} \text{ otherwise}\end{aligned}$$

4.3.2 S and X edge computation

We discuss computing semantic matches between pairs of tables. The schema matching technique proposed for web tables in [21] is an important signal to determine S edges, i.e., there is likely to be an S edge between two web tables T and T' if the common entities in the two tables have equal values on the second column. Define tuple similarity $Sim_S(T, T')$ as the fraction of common entities that have approximately equal values:

$$Sim_S(T, T') = \frac{|(t, t') \in T \times T' \text{ s.t. } t.A = t'.A \wedge t.B \sim t'.B|}{|(t, t') \in T \times T' \text{ s.t. } t.A = t'.A|}$$

Note that we allow slight variations in values while checking the equalities. S is a good assignment for $B_{TT'}$ if $Sim_S(T, T')$ is above a certain threshold, say δ.

For X edges, we follow the computation technique described in the staged approach, i.e., there is likely to be an X edge associated with a set $R \in \mathcal{R}$ of conversion rules between T and T' if (i) T's labels contain the LHSs of R (ii) T''s labels contains the RHSs of R and (iii) the values of the common entities can be converted using the product of the conversion factors. Recall $Sim_X(T, T', R)$ denotes the fraction of common entities that can be converted using the product of the conversion factors. An X edge associated with set of rules R is a good assignment if (i) and (ii) are true and $Sim_X(T, T', R)$ is above a certain threshold, say δ. We define the final clique potential $\psi_e(T, T', L_T, L'_T, B_{TT'})$ as follows:

$$\begin{aligned}&\psi_e(T, T', L_T, L'_T, B_{TT'}) \\ =\ & Sim_S(T, T') \text{ if } Sim_S(T, T') > \delta \ \wedge \ B_{TT'} = S \\ =\ & Sim_X(T, T', R) \text{ if } Sim_X(T, T', R) > \delta \ \wedge \ B_{TT'} = R \\ & \wedge \bigcup_{r \in R} r.LHS \subseteq L_T \ \wedge \ \bigcup_{r \in R} r.RHS \subseteq L'_T \\ =\ & 0 \text{ otherwise}\end{aligned}$$

4.3.3 Local Extraction of year labels

Just like SU labels, local extraction is an important clue in assigning year labels, i.e., we assign a year label $y \in \mathcal{Y}$ to T if the y occurs either in the column header H_T of T_B or in the context C_T of T. As in the previous case, we define two binary feature functions: (i) $f_H(T, y)$, which is set to 1 if H_T contains y and to 0 otherwise and (ii) $f_C(T, y)$ which is set to 1 if C_T contains y and to 0 otherwise. y_T is a good assignment if either $f_H(T, y_T)$ or $f_C(T, y_T)$ is 1. We define the node potential $\psi_y(T, y_T)$ as follows:

$$\psi_y(T, y_T) = \max(f_H(T, y_T), f_C(T, y_T))$$

4.3.4 SU and year label propagation

If there is an S edge between T and T', by definition, we can propagate all the labels $l \in L_T$ of T to T' and vice-versa. For example, in figure 2, we propagate the labels (*mil* and *Euro*) from T2 to T3. L_T and L'_T are good assignments to two tables T and T' connected by an S edge if all or most of their elements are the same, i.e., the set similarity (e.g., Jaccard Similarity) is high.

If there is an X edge (associated with a set R of rules) from T to T', we can propagate all labels $l \in (L_T - \bigcup_{r \in R} r.LHS)$ from T to T' and all labels $l \in (L'_T - \bigcup_{r \in R} r.RHS)$ from T' to T.[2] The insight here is that T and T' are semantically identical except in the scales and units present in the rules connecting them; so, all the other labels of T apply to T' and vice-versa. For example, consider two tables T and T': T has labels $L_T = \{mil, USD\}$ and there is an X edge associated with the rule $USD = 0.77 \times Euro$ from T to T'. We can propagate the label $\{mil, USD\} - \{USD\} = \{mil\}$ from T to T'. L_T and L'_T are good assignments to two tables T and T' connected by an X edge associated with the set R of rules if all or most of the elements in $L_T - \bigcup_{r \in R} r.LHS$ and $L'_T - \bigcup_{r \in R} r.RHS$ are the same, i.e., the set similarity (e.g., Jaccard Similarity) between those two sets is high. We define the edge potential $\psi_{lp}(T, T', L_T, L'_T, B_{TT'})$ as follows:

$$\begin{aligned}&\psi_{lp}(T, T', L_T, L'_T, B_{TT'}) \\ =\ & JaccSim(L_T, L'_T) \text{ if } B_{TT'} = S \\ =\ & JaccSim(L_T - \bigcup_{r \in R} r.LHS, L'_T - \bigcup_{r \in R} r.RHS) \text{ if } B_{TT'} = R \\ =\ & 0 \text{ otherwise}\end{aligned}$$

We can propagate the year label from T to T' if there is either a S edge or X edge between them. y_T and y'_T are good assignments to two tables T and T' connected by an S or X edge if $y_T = y'_T \neq NA$. We define the edge potential $\psi_{lp}(T, T', y_T, y'_T, B_{TT'})$ as follows:

$$\begin{aligned}\psi_{lp}(T, T', y_T, y'_T, B_{TT'}) &= 1 \text{ if } y_T = y'_T \wedge y_T \neq NA \\ &= 0 \text{ otherwise}\end{aligned}$$

4.3.5 Spurious edge elimination

Schema matching techniques can introduce spurious edges. One significant source of errors is time-varying attributes: if the values of an attribute (say, revenues) did not change significantly from one year to another, it can lead to a spurious match. In Figure 2, the revenues of Pfizer and Abbott Labs did not change significantly from 2010 to 2011; this leads to a spurious semantic match between T_1 and T_4. We disallow such spurious matches by taking large negative values when the year labels are not identical. We define the clique potential $\psi_{ec}(T, T', y_T, y'_T, B_{TT'})$ as follows:

$$\begin{aligned}\psi_{ec}(T, T', y_T, y'_T, B_{TT'}) &= -\infty \text{ if } y_T \neq y'_T \wedge y_T \neq NA \\ &\quad \wedge y'_T \neq NA \wedge B_{TT'} \neq NA \\ &= 0 \text{ otherwise}\end{aligned}$$

4.4 Overall Objective

The goal is to find the assignment to the variables L_T, y_T and $B_{TT'}$ such that the following objective is maximized:

$$\begin{aligned}&\sum_T \psi_{su}(T, L_T) + \sum_T \psi_y(T, y_T) \text{ (local extraction)} \\ &+ \sum_{T, T' \in \mathcal{T}} \psi_e(T, T', L_T, L'_T, B_{TT'}) \text{ (S and X edges)} \\ &+ \sum_{T, T' \in \mathcal{T}} \psi_{lp}(T, T', L_T, L'_T, B_{TT'}) \text{ (SU label propagation)} \\ &+ \sum_{T, T' \in \mathcal{T}} \psi_{lp}(T, T', y_T, y'_T, B_{TT'}) \text{ (year label propagation)} \\ &+ \sum_{T, T' \in \mathcal{T}} \psi_{ec}(T, T', y_T, y'_T, B_{TT'}) \text{ (edge constraint)} \qquad (1)\end{aligned}$$

[2]We consider the directionality here. However, considering the reverse direction (from T' to T) yields the same result.

4.5 Inference Algorithm

4.5.1 Independent Inference

We first show how to find the optimal node and edge
ing to maximize the above objective in the case of no
propagation. This optimization can be solved in polyn
time because the optimization for each objective comp
can be solved completely based on local information. T
similar to the *staged approach* described before. The p
code is shown in Algorithm 1. The local extraction of
scale and year is first performed based on feature fun
defined above. Once the node labels have been detern
we compute the S and X edges using the similarity fur
and the computation technique described above. The
labels can then be obtained accordingly. Finally, we c
the edge label to NA for those having different years o
sides to satisfy the edge constraint ψ_{ec}.

Algorithm 1: Independent Inference

```
1  foreach table T ∈ 𝒯 do
2      Initialization: L_T = {}, y_T = NA
       // local extraction of unit and scale
3      foreach label l ∈ ℒ do
4          if max(f_H(T, l), f_V(T, l)) = 1 then
5              L_T.add(l);
       // local extraction of year
6      foreach year y ∈ 𝒴 do
7          if max(f_H(T, y), f_C(T, y)) = 1 then
8              y_T = y;
   // S and X edge computation
9  foreach table pair T, T' do
10     B_TT' = arg max ψ_e(T, T', L_T, L'_T, B_TT');
   // Spurious edge elimination
11 foreach table pair T, T' do
12     if ψ_ec(T, T', y_T, y'_T, B_TT') = −∞ then
13         B_TT' = NA;
14 return L_T, y_T, B_TT';
```

4.5.2 Hardness of Collective Inference

We next show that finding the labels that maximize the final objective function with label propagation is NP-Hard, even for a simple case where we only have one variable y_T. In this simple case, our objective is turned to

$$\max_{\mathbf{y}} \quad \sum_{T\in\mathcal{T}} \psi_y(T, y_T) + \sum_{T,T'\in\mathcal{T}} \psi_{ec}(T, T', y_T, y'_T) + \sum_{T,T'\in\mathcal{T}} \psi_{lp}(T, T', y_T, y'_T) \qquad (2)$$

THEOREM 1. *Finding the optimal labeling for objective (2) is NP-Hard.*

PROOF. We prove it by reduction from metric labeling problem [10]. Metric labeling is defined over a weighted undirected graph $G = (V, E)$ and a label set L, where the weight (nonnegative) $w(v, v')$ indicates the strength of the relationship between node v and v'. We are given: (1) a nonnegative cost function $c(v, l)$ for assigning label l to the node v; (2) a distance $d(l, l')$ between label l and l'. The objective is to find a labeling function $f : V \to L$ such that the following total cost is minimized:

$$\mathcal{C}(f) = \sum_{v\in V} c(v, f(v)) + \sum_{(v,v')\in E} w(v, v') \cdot d(f(v), f(v'))$$

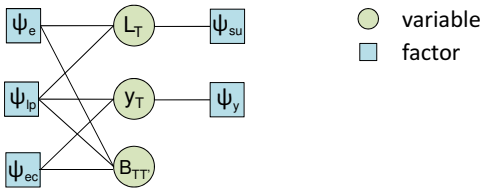


Figure 6: Factor graph.

To reduce the metric labeling to our problem, we let each node $v \in V$ corresponds to one $T \in \mathcal{T}$. The variable y_T can only take labels from set L and can not be NA. Then we define node potential ψ_y and edge potential ψ_{ec}, ψ_{lp} as follows:

$$\begin{aligned} \psi_y(v, l) &= -c(v, l) \\ \psi_{ec}(v, v', l, l') &= -w(v, v') \cdot d(l, l') \\ \psi_{lp}(v, v', l, l') &= 0 \end{aligned}$$

Since the the cost c and distance d are both nonnegative, maximizing $\psi_y(v, l)$ is equivalent to minimizing the cost for assigning l to v. Similarly, maximizing $\psi_{ec}(v, v', l, l')$ is equivalent to minimizing the cost for assigning l and l' to a pair of nodes. We let ψ_{lp} be zero so that it does not affect the final labeling. Hence, the objective function (2) is consistent with the total cost in metric labeling. If a labeling function f can be found in polynomial time to maximize the objective (2), it can also minimize the total cost $\mathcal{C}$ in metric labeling problem. □

4.5.3 Approximation Algorithm

We develop an approximation algorithm to resolve the collective inference by adapting the techniques proposed in probabilistic graphical model, specifically, the belief propagation algorithm [11]. Belief propagation algorithm operates on a factor graph which contains variable nodes and factor nodes, and proceeds by passing "belief" via messages between variable nodes and factor nodes. In our problem, there are three variables and five factors. We show the factor graph in Figure 6. There is an edge between a variable x and a factor ψ iff x appears in ψ.

The algorithm starts with initializing the node potential ψ_{su} and ψ_y for each table T by doing the local extraction of scale, unit and year labels. The collective inference is then processed by iteratively passing messages between neighboring nodes. At each iteration, we schedule the message passing process in three steps. The first step is centered on edge potential ψ_e. The messages are sent from the scale/unit label L_T to ψ_e, then to the edge label $B_{TT'}$ and then sent back. This step tries to create more edges between tables based on current belief on node labeling and, at the same time, tries to select labels that are beneficial for edge creation. At the second step, the focus is on the edge potential ψ_{lp} for label propagation. The messages are passed from L_T, y_T and $B_{TT'}$ to ψ_{lp}; ψ_{lp} then sends back messages according to the set of labels that produce the highest potential. The final step deals with the constraint ψ_{ec} for eliminating spurious edges. The passing schedule for this step is similar to Step 1 except that the messages are sent between year label y_T, edge label $B_{TT'}$ and ψ_{ec}. The algorithm stops when the message values converge or the number of iteration reaches the max limit. The final values of variables are derived from the messages. The details of the algorithm is presented in Algorithm 2.

Algorithm 2: Collective Inference

foreach *table* $T \in \mathcal{T}$ **do**
  Compute $\psi_{su}(T, L_T), \psi_y(T, y_T)$;
Initialization: $M = 0$
while *Iteration* $\leq$ *max_iteration* **do**
  foreach *table pair* T, T' **do**
    // Step 1
    $M(L_T \to \psi_e) = \psi_{su}(T, L_T) + \sum_{T'} M(\psi_{lp} \to L_T)$;
    $M(L_{T'} \to \psi_e) = \psi_{su}(T, L_{T'}) + \sum_{T''} M(\psi_{lp} \to L_{T'})$;
    $M(\psi_e \to B_{TT'}) = \max_{L_T, L_{T'}}[\psi_e + M(L_T \to \psi_e) + M(L_{T'} \to \psi_e)]$;
    $M(B_{TT'} \to \psi_e) = M(\psi_{lp} \to B_{TT'}) + M(\psi_{ec} \to B_{TT'})$;
    $M(\psi_e \to L_T) = \max_{L_{T'}, B_{TT'}}[\psi_e + M(L_{T'} \to \psi_e) + M(B_{TT'} \to \psi_e)]$;
    $M(\psi_e \to L_{T'}) = \max_{L_T, B_{TT'}}[\psi_e + M(L_T \to \psi_e) + M(B_{TT'} \to \psi_e)]$;
    // Step 2
    $M(L_T \to \psi_{lp}) = \psi_{su}(T, L_T) + \sum_{T'} M(\psi_e \to L_T)$;
    $M(L_{T'} \to \psi_{lp}) = \psi_{su}(T, L_{T'}) + \sum_{T''} M(\psi_e \to L_{T'})$;
    $M(y_T \to \psi_{lp}) = \psi_y(T, y_T) + \sum_{T'} M(\psi_{ec} \to y_T)$;
    $M(y_{T'} \to \psi_{lp}) = \psi_y(T, y_{T'}) + \sum_{T''} M(\psi_{ec} \to y_{T'})$;
    $M(B_{TT'} \to \psi_{lp}) = M(\psi_e \to B_{TT'}) + M(\psi_{ec} \to B_{TT'})$;
    $M(\psi_{lp} \to B_{TT'}) = \max_{L_T, L_{T'}, y_T, y_{T'}}[\psi_{lp} + M(L_T \to \psi_{lp}) + M(L_{T'} \to \psi_{lp}) + M(y_T \to \psi_{lp}) + M(y_{T'} \to \psi_{lp})]$;
    $M(\psi_{lp} \to L_T) = \max_{L_{T'}, y_T, y_{T'}, B_{TT'}}[\psi_{lp} + M(L_{T'} \to \psi_{lp}) + M(y_T \to \psi_{lp}) + M(y_{T'} \to \psi_{lp}) + M(B_{TT'} \to \psi_{lp})]$;
    $M(\psi_{lp} \to L_{T'}) = \max_{L_T, y_T, y_{T'}, B_{TT'}}[\psi_{lp} + M(L_T \to \psi_{lp}) + M(y_T \to \psi_{lp}) + M(y_{T'} \to \psi_{lp}) + M(B_{TT'} \to \psi_{lp})]$;
    $M(\psi_{lp} \to y_T) = \max_{y_{T'}, L_T, L_{T'}, B_{TT'}}[\psi_{lp} + M(y_{T'} \to \psi_{lp}) + M(L_T \to \psi_{lp}) + M(L_{T'} \to \psi_{lp}) + M(B_{TT'} \to \psi_{lp})]$;
    $M(\psi_{lp} \to y_{T'}) = \max_{y_T, L_T, L_{T'}, B_{TT'}}[\psi_{lp} + M(y_T \to \psi_{lp}) + M(L_T \to \psi_{lp}) + M(L_{T'} \to \psi_{lp}) + M(B_{TT'} \to \psi_{lp})]$;
    // Step 3
    $M(y_T \to \psi_{ec}) = \psi_y(T, y_T) + \sum_{T'} M(\psi_{lp} \to y_T)$;
    $M(y_{T'} \to \psi_{ec}) = \psi_y(T, y_{T'}) + \sum_{T''} M(\psi_{lp} \to y_{T'})$;
    $M(\psi_{ec} \to B_{TT'}) = \max_{y_T, y_{T'}}[\psi_{ec} + M(y_T \to \psi_{ec}) + M(y_{T'} \to \psi_{ec})]$;
    $M(B_{TT'} \to \psi_{ec}) = M(\psi_{lp} \to B_{TT'}) + M(\psi_e \to B_{TT'})$;
    $M(\psi_{ec} \to y_T) = \max_{y_{T'}, B_{TT'}}[\psi_{ec} + M(y_{T'} \to \psi_{ec}) + M(B_{TT'} \to \psi_{ec})]$;
    $M(\psi_{ec} \to y_{T'}) = \max_{y_T, B_{TT'}}[\psi_{ec} + M(y_T \to \psi_{ec}) + M(B_{TT'} \to \psi_{ec})]$;
  if *converged* **then**
    break;
return $\mathbf{L_T}$, $\mathbf{y_T}$, $\mathbf{B_{TT'}}$;

4.5.4 Scalable Implementation of Collective Inference

There are hundreds of millions of tables on the web; to build the semantic graph over all web tables, the belief propagation algorithm need to scale to factor graphs containing billions of nodes. Many large scale graph processing tasks can be expressed as a sequence of iterations, in each of which a vertex can receive messages sent in the previous iteration, send messages to other vertices, and modify its own state and that of its outgoing edges or mutate graph topology. Hence, several systems have been developed for above computation model over very large-scale graphs (e.g., Pregel [13], Trinity [17]). Since the belief propagation algorithm follows the same computational model, we can leverage those technologies for scalable implementation of collective inference.

5. QUERY PROCESSING

We focus on entity augmentation queries in this section. Our system can support other types of web table search queries [21, 14]. We first define the query interface of our system.

5.1 Query Interface

Definition 1 (Query Interface). *An entity augmentation query* $Q = \langle A, E, SU, Y \rangle$ *consists of four components:*
A: keywords describing attribute name.
E: the set of entities of interest.
SU: unit and scale (optional).
Y: timestamp (optional).

The proposed API significantly extends the previously proposed entity augmentation API which takes only attribute name and entity set as input [21]. We extend the query language with SU and Y to unambiguously specify the augmentation task. Furthermore, it better leverages the semantic graph we build in this paper.

5.2 Query Time Processing

We briefly introduced the two main steps for EA queries in Section 3. We also described the indexes built for efficient query time processing. We now present the processing in detail. The pseudo code is shown in Algorithm 3.

Step 1: Identify matching tables. Given the query Q, we first leverage the EI index to identify the set of tables that contains at least one of the entities in $Q.E$. Among these tables, we use the NLI index to identify the subset of tables that satisfy the following conditions: (1) The column header matches with the attribute name keywords in $Q.A$; (2) The year label matches with the query label in $Q.Y$ (if any); (3) The unit and scale labels match with each of the query labels in $Q.SU$. A unit/scale label is considered a match if it occurs in the same mutex group as $Q.SU$. This is because they are convertible from one to another; we can convert values from the units they are available in to the desired query unit. Finally, for each qualified table, we search for its semantically matching tables using GI index. The qualified tables together with their matching tables become our sources for filling in values at next step.

Step 2: Fill-in values. For each table identified at Step 1, we collect the values provided in the table for each of

Algorithm 3: Query Processing

```
   // Step 1: Identify matching tables
1  S = EI(Q);
2  S = S ∩ NLI(Q);
3  S = S ∪ ⋃_{T∈S} GI(T);
   // Step 2: Fill-in values
4  foreach table T ∈ S do
5      foreach tuple t ∈ T do
6          if t.e ∈ Q.E then
7              v = Convert(t.v, L_T, Q.SU);
8              Aggregate(t.e, v);
9  AugmentEntity(Q.E);
```

the query entities. We compare the table's semantic labels (unit and scale) with the query labels. If they do not agree, we convert the values to the desired unit and scale. The converted values are then aggregated. Finally, we augment the query entities with the values with the highest aggregate score.

EXAMPLE 2. *Consider the web tables shown in Figure 2. Suppose the collective inference approach has discovered the semantic labels and edges shown in Figure 2. Consider the EA query in Figure 1(b). The query processing algorithm will identify T_1, T_2 and T_3 as matching tables because (1) their column headers contain "revenues" (2) their year labels match with '2010' and (3) they contain unit/scale labels that are in the same mutex group as USD (USD, Euro and Euro respectively). For Eli Lilly, T_1 provides* 21.8 *which is already in USD bil. For Merck, T_1 provides* 27.4 *and T_2 provides* 21091*; the latter is converted to USD bil which also results in* 27.4*. For Roche and Novartis, T_3 provides the values* 36113 *and* 33762 *respectively; they are converted to USD bil resulting in* 46.9 *and* 43.8 *respectively. This produces the desirable, semantically consistent results (all from 2010 and all in USD bil) shown in Figure 1(b).*

6. EXPERIMENTS

We present an experimental evaluation of the techniques proposed in the paper. The goals of the experimental study are:

- To evaluate the quality of the semantic labels (unit, scale and year) and semantic matches discovered by our collective inference approach
- To evaluate the impact of the discovered labels and matches on entity augmentation (EA) queries and to compare with the baseline approach
- To evaluate the efficiency of processing EA queries specified using the proposed query interface

6.1 Experimental Setup

Datasets: We conducted experiments on three real-life datasets of web tables: Company, Country and City, which are extracted from a recent snapshot of Microsoft Bing search engine. Table 1 shows the detailed statistics of the datasets including the number of web tables, the total number of numeric attributes across all the tables and the average number of numeric attributes per table.

EA Queries and golden truth: We conducted experiments for the following EA queries:

- *Company revenue and profit:* We compiled the golden truth by extracting the Forbes Global 2000 list of year 2011 from [2]. We chose various subsets of those companies as the entities. We considered multiple EA queries with "revenue" and "profit" as attribute keywords, in various units and scales and from year 2011.
- *Country area:* We compiled the golden truth by extracting the list of 249 countries along with their total area from a Wikipedia page [3]. We chose various subsets of those countries as entities. We considered multiple EA queries with "area" as attribute keyword and in different units.
- *Country taxrate:* We compiled the golden truth by extracting the list of corporate income tax rates of 34 OECD member countries from Tax Foundation [4]. We chose various subsets of those countries as entities. We considered multiple EA queries with "tax rate" as attribute keywords and from different years.
- *City population:* We compiled the golden truth by extracting the list top 600 largest cities in 2011 from the City Mayors site [1]. We chose various subsets of those cities as entities. We considered multiple EA queries with "population" as attribute keyword, in various scales and from year 2011.

Table 1: Datasets statistics

Domain	Tables	Numeric Attrs	Average
Company	39,223	80,149	2.04
City	81,977	140,459	1.71
Country	159,730	344,509	2.16

Conversion rules: The set of conversion rules are shown in Table 2. We list only the ones relevant to the EA queries reported in the paper. We omit the conversion factors and only show the units/scales involved in the rules. For example, for the city population EA query, the relevant rules are those for converting between four scale labels, i.e. billion, million, thousand and NA. We also have a database containing synonyms of unit and scale descriptors.

Table 2: Conversion rules

Query Attribute	Conversion tokens
Company revenue/profit	euro usd
	billion million thousand NA
City population	billion million thousand NA
Country area	sq.meter sq.feet sq.km sq.mile

Our algorithms are implemented in C# and the experiments were performed on a Windows server with a Quad-Core AMD Opteron 2.3 GHz CPU and 128GB RAM.

6.2 Evaluating Quality of Semantic Graph

We evaluate the graph quality in terms of the quality of the semantic labels (unit, scale and year) and semantic matches. In this set of experiments, we report the results on Company dataset.

6.2.1 Node Label Quality

We manually selected a random set of 1000 EAB tables whose attributes are related to company revenue. We compare the labels on these tables discovered by our collective inference approach (CI in short) with the ones discovered by the independent inference approach (II in short). Table 3 shows the quality results of the node labels. Recall that II computes labels based on local extractions only and X edges based on those labels. For SU labels (scale and unit), *CI discovers more labels* compared with II (labels for 35% more nodes) and *with high precision* ($\sim$ 97%). II annotates 510 out of 1000 tables; CI annotate 180 more tables (due to label propagation).

We first analyze the new tables annotated by CI approach. We randomly sampled 100 (out of 180) tables and manually checked the labels. We found 97 tables are labeled correctly and completely and 3 tables (all in <million, usd>) are partially correct. We further analyze those 3 partially correct tables. CI annotates two of them with million only and one with usd only. By manually checking the original web page, we found that the first two tables contain revenue values of year 1955. Since the majority of web tables contain recent information and those values are very different from the ones of recent years, they are not connected to the majority of tables in the semantic graph. Hence, CI fails to propagate the correct unit to those two tables. The other error occurs on the table which contains information about Revenue per Employee instead of total revenue.

We now evaluate the labels discovered by both II and CI. We observe 428 out of 510 labels are identical. CI annotates 81 of those 510 tables with additional labels (e.g., usd → <usd, million>). Interestingly, CI even made a correction to one of the tables[3]. The table header specifies million, whereas the values are in fact in billion.

For year labels, *CI labels 4X more nodes* compared with II and *with a precision of* ∼ 89%. II annotates only 129 out of 1000 tables while CI annotates 527 tables (due to label propagation). Again, we take a random sample of 100 tables and manually check the year annotation. The accuracy is 89%. From a careful analysis of the data, the mistakes made on the 11% tables can be mostly attributed to the similar values between different years. For example, we notice there are two tables, one of year 2006 and one 2011. However, there are 6 out of 8 entities in the 2006 table having values very close to 2011 revenue values. Similar to the SU label, our approach can also make correction to year labels. While analyzing the annotation results, we encounter a 2011 revenue table[4] that happens to have "2010 revenue" in the header – our approach correctly assigned the label 2011 to it.

Table 3: Node label quality

SU	II	510	same: 428 diff: 1
	CI	690	more: 81 (100%) new : 180 (∼97%)
Year	II	129	same: 127 diff: 2
	CI	527	more: – new : 398 (∼89%)

6.2.2 Edge Quality

We study the quality of the semantic matches (i.e., edges) produced by our techniques. We evaluate it on the Company dataset as well. We consider the edges between the 1000 revenue tables. Table 4 shows the edge quality results. Recall that our approach creates new X edges and eliminates spurious edges (between tables that contain information from different years). In the table, we show the number of X edges created and the spurious edges eliminated by II and CI approaches. For X edge creation, II creates 205 edges while CI creates 405. For spurious edge elimination, II removes only 4 edges while CI eliminates 336 edges (out of 1935 S edges). Both are due to label propagation: more labels lead to more X edge creation and more spurious edge elimination. To evaluate the accuracy of the edges we created and eliminated, we randomly took 100 sample edges from each of the set and performed another round of manual checking. The accuracy results are included in the Table 4 as well. The accuracy of CI is 83% and 79% for created and eliminated edges respectively. By analyzing the data, we found that the mistakes are again caused by the same (or similar) values shared by tables from different years.

Table 4: Edge quality

	X Edge		**Elimination**	
	II	CI	II	CI
Number	205	405	4	336
Accuracy	∼90%	∼83%	100%	∼79%

6.3 Evaluating Entity Augmentation Quality

We implemented four different approaches (2 baseline approaches and 2 variations of our approach):

• *S:* This is a baseline approach which treats unit/scale and year specified in the query as normal keywords in attribute name. It considers a table relevant to query Q only if the table header contains all of the keywords in $Q.A$, $Q.SU$ and $Q.Y$. The edges between tables are simple S edges.

• *S-Syn:* This is similar to S approach except that we leverage synonyms of attribute names. We mine the attribute synonyms using the approach proposed in [21]. For attribute name, a table T is considered a match to Q if the union of the synonyms of T's attribute covers all the keywords in $Q.A$, $Q.SU$ and $Q.Y$.

• *Independent Inference(II):* This approach computes SU and year labels based on local extraction and X edges based on those labels. It also eliminates spurious edges based on those local labels. It uses the query processing algorithm proposed in Section 5.

• *Collective Inference(CI):* This is our proposed approach based on the collective inference algorithm. It also uses the query processing algorithm proposed in Section 5.

In this section, we report the EA quality for the above approaches. We use two metrics to measure EA quality: coverage and precision.

$$coverage = \frac{\#entities\_augmented}{\#entities\_in\_query}$$

$$precision = \frac{\#entities\_correctly\_augmented}{\#entities\_augmented}$$

We first present the results for two simple scenarios where the query specifies only SU (no year) and only Y (no SU) (Country-area and Country-tax rate respectively). We then present the results for the scenario where both SU and Y are specified in the query.

6.3.1 SU-only Scenario

Figure 7 shows the coverage and precision results of augmenting country area in square km. The x-axis represents different entity sets: the top-k countries in descending order of the country area for various values of k. We did this to study the sensitivity of the approaches to head and tail entities [5] All the four approaches achieve high coverage; the coverage decreases as we encounter more tail entities. However, S and *S-Syn* approaches have poor precision: the highest is 0.22 on 250 countries. In contrast, II and CI achieve consistently high precision across all entity sets: an average

[3] http://www.americanet.de/html/wirtschaft_grosste_firmen.html

[4] http://waste360.com/corporate-sustainability-50/2011

[5] This assumes that the countries with larger area are head countries and the ones with smaller area are tail. There is a correlation between area and query popularity but this, in general, is not true; hence, the sensitivity aspect of this study is not conclusive.

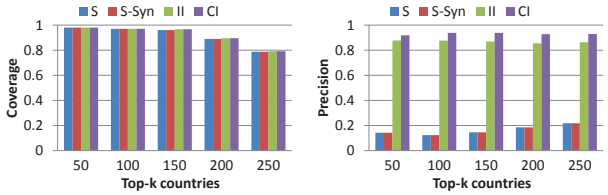


Figure 7: Country area: square km

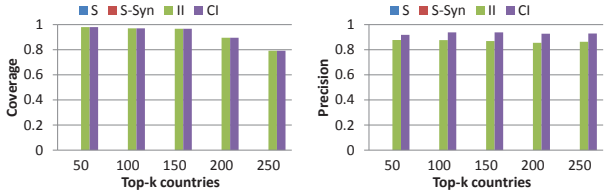


Figure 8: Country area: sqft

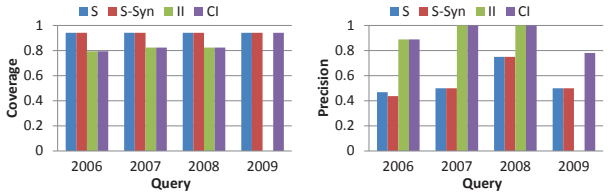


Figure 9: Country taxrate

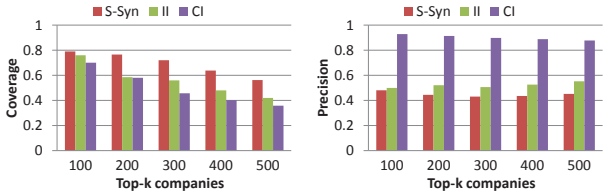


Figure 10: Company revenue: million $

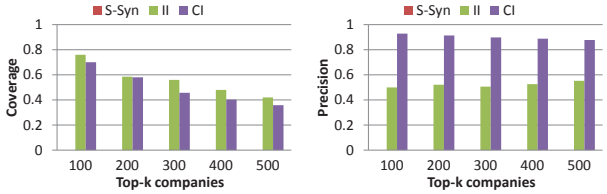


Figure 11: Company revenue: billion $

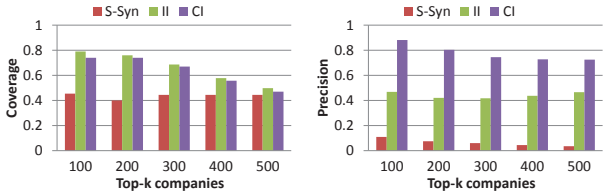


Figure 12: Company profit: million $

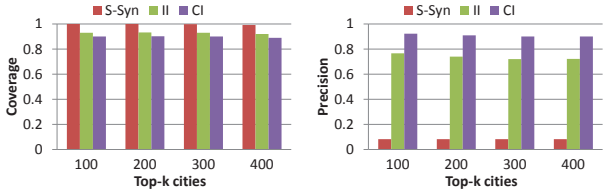


Figure 13: City population: million

precision of 0.87 for II approach and 0.93 for CI. This confirms that adding X edges and semantic labels significantly impacts the quality of EA queries. The improvement is more significant in CI where we create more X edges and labels via label propagation.

Figure 8 shows the results of augmenting country area using a different unit: sqft. The drawback of baseline approaches is more obvious in this query. The baseline approaches fail to augment even a single entity. This is because country area is not available in the desired unit; hence, *S* and *S-Syn* finds zero relevant tables. On the other hand, our approaches achieve high coverage and precision as we can convert the values from the units they are available in to the desired unit.

6.3.2 *Year-only Scenario*

Figure 9 shows the results of augmenting country tax rate for year 2006-2009. The baseline approaches reach high coverage for all years. However, the average precision is only 0.55 and 0.54 for *S* and *S-Syn* respectively. This is due to spurious edges between tables that contain information from different years; hence, the results contain values from years different from the query year. II and CI improves the average precision to 0.72 and 0.92 respectively by eliminating spurious edges. Note that II returns zero result for year 2009. This is because II fails to hit the tables that contain the query answer without propagating the years. Overall, CI significantly outperforms II and the baselines as it does a better job in propagating the labels and eliminating spurious edges (as shown in Table 4).

6.3.3 *SU+Year Scenario*

We compare the approaches for the scenario where the query specifies both unit/scale and year. We present results for four entity augmentation queries on Company and City datasets. For this set of experiments, we omit the *S* approach in the figures, because *S* either returns zero results or similar results as *S-Syn*.

Figure 10 compares the 3 approaches for the company-revenue query in <million, $> for year 2011. The baseline approach has good coverage but suffers from poor precision ($\sim$ 0.4). II does not improve the precision significantly but CI improves the precision to 0.9. This indicates that the higher quality labels and edges produced by CI significantly improves the quality of EA queries.

Figure 11 shows the results for the company-revenue query in <billion, $> for year 2011. Similar to augmenting country area in sqft, the baseline approach returns zero results, whereas CI is not affected by changing of query scale from million to billion. This is due to the unavailability of information in <billion, $> and our ability to convert the values from the units they are available into <billion, $>.

Figure 12 plots the results of the company-profit query in <million, $> for year 2011. CI consistently outperforms both II and the baseline approach. Figure 13 shows the results for the city-population in million for year 2011. Again, CI significantly outperforms both II and the baseline approach on precision and achieves high coverage as well. The average coverage and precision are 0.9 and 0.9 respectively.

6.4 EA Query Efficiency

We evaluate the efficiency of processing EA queries using the CI approach. Figure 14 shows the query response time as we increase the number of entities from 50 to 400. The response time is within 1 second and increases sub-linearly with the number of entities, making our approach usable in interactive applications.

In summary, our experiments show that our collective inference approach produces higher quality semantic matches and semantic labels which significantly improves the quality of entity augmentation queries.

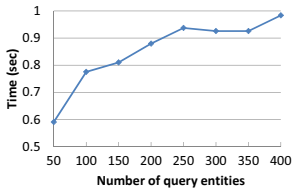


Figure 14: Query response time

7. RELATED WORK

Our work is most related to works on annotating web tables [12, 19] and semantic matching between web tables [21]. Given a catalog (e.g., YAGO), Limaye et. al. annotates columns in web tables with those catalog types, pairs of columns with those catalog relationships and table cells with those catalog entities [12]. In contrast, Venetis et. al. leverages a database of classes, instances and relationships extracted automatically from the web to perform such annotations. Our annotations are fundamentally different from those annotations: we annotate columns with unit, scale and year information. To the best of our knowledge, such annotations have not been studied before in the context of web tables. Our work significantly extends the semantic matching in InfoGather [21]. InfoGather focuses only on S edges which is inadequate for numeric and time-varying attributes. InfoGather+ discovers X edges and semantic labels in addition to S edges; as a result, it performs significantly better compared with InfoGather.

Search over web tables has received significant research attention recently [6, 5, 21, 14]. Cafarella et. al. proposes a simple keyword search that returns a ranked list of web tables [5]. More recent works propose to consolidate matching tables to synthesize the desired table for the user. One example is the entity augmentation operation proposed in [5, 21]. Another example is the user specifying sets of keywords she wishes to see in the answer table (e.g., "Pain killer|Side effects"); the system consolidates a single answer table for such queries [14]. Our work on semantic graph is orthogonal to these works; it is critical for all consolidation-based approaches. Secondly, we significantly extend the entity augmentation API for numeric and time-varying attributes.

Our work on semantic graph is related to the vast body of work on schema matching [15, 9]. Traditional schema matching works find semantic matches between two database schemas. Most works focus on one-to-one mappings; the works that deal with complex matches rely only on "local" extraction [8] (i.e., looks for labels in the column headers and values). In this paper, we argue that this is inadequate for web tables as such information is often missing from column headers. We leverage the fact that there are millions of tables on the web; even if the local header does not contain the information, some semantically matching table will contain it. Furthermore, we annotate tables with semantic labels which is not considered in schema matching works.

Recently, researchers have started studying ways to assign a timestamp (or time span) to facts in a knowledgebase (like YAGO or Freebase) [20, 18]. In this paper, we study year annotation in conjunction with scale and unit annotation and semantic matches in the context of web tables.

8. CONCLUSION

In this paper, we present the InfoGather+ system to answer entity augmentation queries accurately for numeric and time-varying attributes. We build a semantic graph that (i) labels columns with unit, scale and year information and (ii) computes semantic matches between columns even when the same numeric attribute is expressed in different units or scales. Our experiments demonstrate that by leveraging the semantic graph, we can answer such queries much more accurately compared with the state-of-the-art.

Our work can be extended in multiple directions. In this paper, we focused on leveraging the semantic graph for entity augmentation; how to leverage it for other modes of web table search is an item of future work.

9. REFERENCES

[1] City Mayors. http://www.citymayors.com/statistics/largest-cities-mayors-1.html.

[2] Forbes Global 2000. http://www.forbes.com/lists/2012/18/global2000_2011.html.

[3] List of countries by area. http://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_area.

[4] Tax Foundation. http://taxfoundation.org/article/oecd-corporate-income-tax-rates-1981-2012.

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

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

[7] M. J. Cafarella, A. Y. Halevy, Y. Zhang, D. Z. Wang, and E. Wu. Uncovering the relational web. *WebDB*, 2008.

[8] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: discovering complex semantic matches between database schemas. *SIGMOD*, pages 383–394, 2004.

[9] A. Doan and A. Y. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine*, 26:83–94, 2005.

[10] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. *J. ACM*, 49(5):616–639, Sept. 2002.

[11] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

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

[13] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. *SIGMOD*, pages 135–146, 2010.

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

[15] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.

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

[17] B. Shao, H. Wang, and Y. Li. The trinity graph engine.

[18] P. P. Talukdar, D. Wijaya, and T. Mitchell. Coupled temporal scoping of relational facts. *WSDM*, pages 73–82, 2012.

[19] 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.

[20] Y. Wang, B. Yang, L. Qu, M. Spaniol, and G. Weikum. Harvesting facts from textual web sources by constrained label propagation. *CIKM*, pages 837–846, 2011.

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