Meimei: An Efficient Probabilistic Approach for Semantically Annotating Tables

Kunihiro Takeoka
NEC Corporation
k-takeoka@az.jp.nec.com

Masafumi Oyamada
NEC Corporation
m-oyamada@cq.jp.nec.com

Shinji Nakadai
NEC Corporation
s-nakadai@az.jp.nec.com

Takeshi Okadome
Kwansei Gakuin University
tokadome@acm.org

Abstract

Given a large amount of table data, how can we find the tables that contain the contents we want? A naive search fails when the column names are ambiguous, such as if columns containing stock price information are named "Close" in one table and named "P" in another table.

One way of dealing with this problem that has been gaining attention is the semantic annotation of table data columns by using canonical knowledge. While previous studies successfully dealt with this problem for specific types of table data such as web tables, it still remains for various other types of table data: (1) most approaches do not handle table data with numerical values, and (2) their predictive performance is not satisfactory.

This paper presents a novel approach for table data annotation that combines a latent probabilistic model with multi-label classifiers. It features three advantages over previous approaches due to using highly predictive multi-label classifiers in the probabilistic computation of semantic annotation. (1) It is more versatile due to using multi-label classifiers in the probabilistic model, which enables various types of data such as numerical values to be supported. (2) It is more accurate due to the multi-label classifiers and probabilistic model working together to improve predictive performance. (3) It is more efficient due to potential functions based on multi-label classifiers reducing the computational cost for annotation.

Extensive experiments demonstrated the superiority of the proposed approach over state-of-the-art approaches for semantic annotation of real data (183 human-annotated tables obtained from the UCI Machine Learning Repository).

1 Introduction

Given a large number of web tables, how can we find tables that contain stock price information? Conventional information retrieval techniques such as column-name based search sometimes fail to find relevant tables because the column/table names are ambiguous. For instance, stock price information may be stored with various column names, sometimes as "P" (for "price") or "Close" (for "closing price") as shown in Figure 1. This issue led to increased attention being paid by academia and industry to semantic annotation of table data, i.e., the unification of table semantics by annotating table cells and columns with canonical knowledge (Venetis et al. 2011; Deng et al. 2013; Mulwad, Finin, and Joshi 2013; Neumaier et al. 2016; Pham et al. 2016; Zhang 2017).

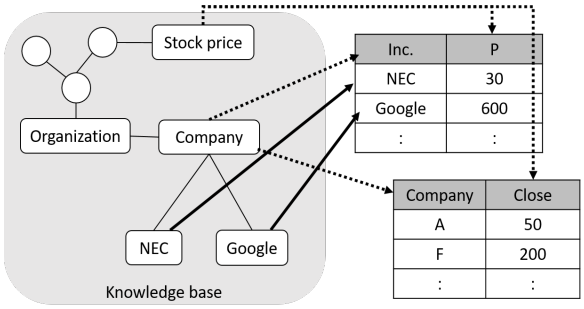

Figure 1: Illustration of semantic table annotation. Ambiguous column names (e.g., columns containing a concept "<Stock price>" are named either "P" or "Close") are disambiguated by linking to corresponding canonical concepts in a knowledge graph.

Most conventional semantic table annotation methods simply link column cells containing textual information to knowledge graph concepts (entities) (Venetis et al. 2011; Deng et al. 2013; Mulwad, Finin, and Joshi 2013; Zhang 2017). The columns containing these cells are called "named-entity columns" (**NE-columns**). The columns containing numerical values are not linked to knowledge, which are called "**literal-columns**." While several methods for annotating literal-columns have been reported, their predictive performance is not satisfactory (Pham et al. 2016; Neumaier et al. 2016).

We have developed a novel approach for annotating various types of table data in which probabilistic-model-based annotation is integrated with highly predictive multi-label classifiers (Tsoumakas and Katakis 2006). Our approach extends a state-of-the-art semantic annotation method based on a Markov random field model (Limaye, Sarawagi, and Chakrabarti 2010) with multi-label classifiers to enable it to support various types of columns including ones containing numerical data and to improve its predictive performance.

We summarize our contributions:

1. We annotate more versatile columns semantically by modeling a table with Markov random field and its potential functions based on multi-label classifiers.
2. The predictive performance of our approach is better than those of conventional methods due to highly predictive multi-label classifiers as potential functions.
3. Our approach to semantic table annotation is faster than conventional methods due to modeling tables and approximation techniques for prediction.

2 Related Work

2.1 Named-Entity-Column Annotation

Most studies for semantic table annotation have focused on named-entity-columns (NE-columns).

(Venetis et al. 2011) proposed using a probabilistic model and isA database constructed from the Web to semantically annotate NE-columns. The annotation of columns by the method has ambiguity because the isA database is not a canonical knowledge.

(Limaye, Sarawagi, and Chakrabarti 2010) proposed annotating columns with concepts by using a Markov random field model to model the semantic structure of a table. The cells, columns, and binary relationships between columns are modeled as corresponding latent variables that represent concepts or relations in a knowledge graph. Potential functions over latent variables, such as column-column and column-content relationships, are used to connect those table components. This model formalizes the semantic table annotation task as the potential maximization. Their method thus captures the interdependencies between table components such as columns and cells. The original potential maximization algorithm (Limaye, Sarawagi, and Chakrabarti 2010) is computationally expensive, so (Mulwad, Finin, and Joshi 2013) proposed an alternative light-weight message-passing algorithm.

While these methods can infer the concepts related to NE-columns, they ignore the existence of literal-columns. Our approach tackles the issue and leverages literal-columns to further improve the accuracy of semantic annotation of NE-columns because literal-columns sometimes have important information for annotating other columns.

2.2 Literal-Column Annotation

While mainstream of semantic annotation focuses on NE-columns, several studies tried semantic annotation for literal-columns recently.

(Pham et al. 2016) proposed annotating literal-columns by classifying columns. Given a non-annotated column, a classifier computes the distances between the column and the already annotated columns, which work as training data. For the distance between columns, their method aggregates several similarity measure scores such as the Jaccard index of trigrams and the statistics of the Kolmogorov-Smirnov test, in which the importance of each similarity is trained beforehand in a supervised manner. Since their approach is based on the classifier, which has to store all the annotated columns as training data, computational and space complexities could be high.

(Neumaier et al. 2016) proposed leveraging the hierarchical structure of numerical concepts (e.g., height $\rightarrow$ height of sportsperson $\rightarrow$ height of football player) extracted from knowledge graphs to annotate literal-columns. Literal-columns annotated with the same concept are assumed to have the same unit of measure (meter, inch, etc.). This means that performance may be poor if columns with the same concept have different units of measure in real data.

(Rümmele, Tyshetskiy, and Collins 2018) proposed using a multi-class classifier for semantic annotation whose textual/numerical features are similar to ours. Compared to their model, our model has several differences. Firstly, we extended the classification model into multi-label classifiers that assigns several concepts to a column because it is natural for a column to have several corresponding concepts (e.g., for a column named "Inc.", concepts "<Company>" and "<Organization>" should correspond). Secondly, we consider the column-interdependencies, such as "<Age>"-column and "<Height>"-column often co-occur in a table, in the form of a Markov random field model to improve the predictive performance.

2.3 Applications

Several applications use semantic table annotation such as InfoGather+ (Zhang and Chakrabarti 2013) and KATARA (Chu et al. 2015). InfoGather+ fills in missing values in a table by semantic table annotation. KATARA is aimed at data cleansing and enriching the knowledge graph by annotating tables semantically. Semantically annotated tables can be useful for several operations such as searching and joining tables (Venetis et al. 2011). It is difficult to do such operations effectively without semantic annotation for tables because columns have ambiguous values. Our method also helps these operations as preliminary.

3 Proposed Method

We model a table as a probabilistic model using a Markov random field with parametric potential functions. To obtain optimal parameters of the potential functions, we use a multi-label classification model. Note that we show the list of main symbols in Table 1.

3.1 Problem Definition

We formalize semantic annotation for table $\mathbf{X}$ as the following problem.

Let knowledge graph G be a graph $G = (V, E)$, where V corresponds a set of "concepts" that represent concepts such as "<stock price>" and "<company>," and E denotes the "is-a" relations between two concepts.

Problem 1 (Semantic annotation for table data)**.** *Given a knowledge graph G and a table $\mathbf{X}$ with R rows and C columns whose c-th column is $\mathbf{x}_c$ and (r, c)-th cell is $x_{r,c}$, find the concept $y_c \in V$ for each column c that best captures*[1] *c's content $\mathbf{x}_c$.*

[1] In previous work (Limaye, Sarawagi, and Chakrabarti 2010),

Table 1: List of main symbols.

notations	descriptions
$\|\mathcal{T}\|$	# of tables in the training dataset
$\|V\|$	# of concepts in the knowledge graph
$\|V^+\|$	# of concepts in the training dataset
C	average # of columns in the training dataset
R	average # of rows in the training dataset
C_t	# of columns in a target table
R_t	# of rows in a target table
$\mathbf{X}$	table contents
$\mathbf{y}$	column concepts $\subset V$
t	title concept $\in V$
$\phi(\cdot)$	potential function in our model
$\mathbf{f}$	multi-label classifier for column-content potential
$\mathbf{g}$	multi-label classifier for column-column potential
θ	parameters of a potential function
e	concept candidate $\in V$
$\mathbf{v}, \mathbf{u}$	a vector representation
d	# of dimensions in the embedding space
I	# of iterations in Gibbs sampling

3.2 Formulation with Markov Random Field

To solve the Problem 1, we reformulate the problem by defining a Markov random field over the input table data and assigned concepts.

Let c be a column identifier, $y_c \in V$ be a latent variable corresponding to c-th column contents $\mathbf{x}_c$ that represents the concept of c-th column, $t \in V$ be a latent variable that represents the concept of the table title, and $\mathbf{x}_c = \{x_{r,c}\}_{r=1}^{R}$ be column c's content. Note that $\mathbf{y}_{-c} = \mathbf{y} \setminus y_c$. We define the joint distribution of the random variables in our model as

$$p(\mathbf{X}, \mathbf{y}, t) = \frac{1}{Z} \prod_c \phi_{yx}(y_c, \mathbf{x}_c)\phi_{yy}(y_c, \mathbf{y}_{-c})\phi_{ty}(t, y_c), \tag{1}$$

where c is a column index, Z is a partition function, and $\phi(\cdot)$ is a potential function. There are three potential functions in our model: (1) column-content potential ϕ_{yx}, (2) column-column potential ϕ_{yy}, and (3) title-column potential ϕ_{ty}. The potential functions have parameters trained by a training dataset. Figure 2 shows an example of our model for a table that has two rows and three columns.

With this modeling, Problem 1 can be reformulated as maximization of the joint probability shown in Equation 1, given table contents $\mathbf{X}$. Before we introduce the preliminary for semantic table annotation (Section 4) and the maximization of joint probability (Section 5), we introduce the details of our model in the remainder of this section.

3.3 Potential Functions

We define three potential functions for our Markov random field model, each of which models correlation between two

they assumed that a column has the ground-truth concepts and the columns contents are instantiated from these concepts. Since ground-truth concepts cannot be obtained for actual data, they prepared human-annotated tables and evaluated the accuracy. In this paper, we followed this procedure.

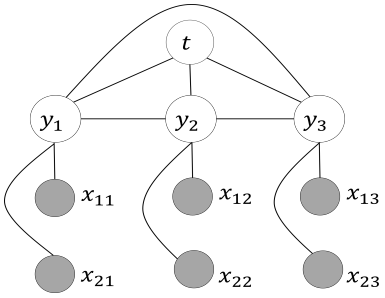

Figure 2: The graphical model of proposed model on a table with two rows and three columns.

random variables. We take a supervised approach to determining the parameters of the potential functions.

We propose an extensible and accurate approach for semantic table annotation based on Markov random field with potential functions based on multi-label classifiers (Tsoumakas and Katakis 2006). Our approach annotates columns accurately compared to conventional methods thanks to potential functions based on multi-label classifiers, which are used to deal with ambiguous column names. For instance, the "Inc." column annotated with "<Company>" in Figure 1 is sometimes annotated with "<Organization>." Multi-label classifiers work better than multi-class classifiers because they express ambiguity with multiple outputs.

Column-Content Potential The column-content potential $\phi_{yx}(y_c, \mathbf{x}_c)$ measures the similarity between observed cells $\mathbf{x}_c$ and candidate column concept y_c. We define the potential function as

$$\phi_{yx}(y_c, \mathbf{x}_c) = \mathbf{u}_{y_c}^{\mathrm{T}} \mathbf{f}_{\theta_1}(\mathbf{x}_c),$$

where $\mathbf{f}_{\theta_1} : \mathbf{x} \rightarrow [0,1]^{|V|}$ is defined by a multi-label classifier $\mathbf{f}_{\theta_1}$ with the trained parameters θ_1, and $\mathbf{u}_{y_c}$ is a one-hot vector of column concept y_c with $|V|$ dimension, which is the number of concepts in the knowledge graph. $\mathbf{f}_{\theta_1}$ has two steps: feature extraction and multi-label classification. We take different approaches to feature extraction for NE- and literal-columns because the characteristics of NE- and literal-columns are different. We explain the details of the feature extraction in Section 3.5. After the feature extraction, we input the extracted features to the multi-label classifiers (one for NE-columns and one for literal-columns) whose parameters are trained using a training dataset (the details are covered in Section 4 and Section 5).

We use a "binary relevance" approach for the multi-label classifiers (Tsoumakas and Katakis 2006). Each binary classifier corresponding to a concept determines whether a column-content relevant to or irrelevant to the concept. This binary relevance approach is the simplest approach to multi-label classification.

Column-Column Potential As shown in Figure 3, concepts of numerical values are sometimes hard to determine since numerical statistics can be similar among different concepts (e.g., age and temperature). To deal with this problem, we prepare the column-column potential $\phi_{yy}(y_c, \mathbf{y}_{-c})$ measures similarity between candidate concept of a column

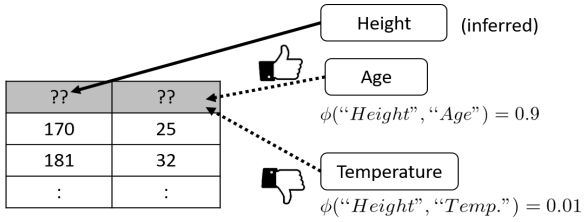

Figure 3: Semantic annotation with column-interdependency consideration improves accuracy for ambiguous literal-columns.

(y_c), and the currently assigned concepts of other columns ($\mathbf{y}_{-c}$). Note that $\mathbf{y}_{-c} = \mathbf{y} - y_c$. We define the potential function as

$$\phi_{yy}(y_c, \mathbf{y}_{-c}) = \mathbf{u}_{y_c}^{\mathrm{T}} \mathbf{g}_{\theta_2}(\mathbf{y}_{-c}),$$

where $\mathbf{g}_{\theta_2} : \mathbf{y}_{-c} \rightarrow [0,1]^{|V|}$ is also defined by a multi-label classifier $\mathbf{g}_{\theta_2}$ with trained parameters θ_2, and $\mathbf{u}_{y_c}$ is a one-hot vector corresponding to y_c. The multi-label classifier $\mathbf{g}_{\theta_2}$ predicts the probabilities of all concepts when given other columns' concept assignments $\mathbf{y}_{-c}$. We give the multi-label classifier $\mathbf{g}_{\theta_2}$ the vector representation of currently assigned concepts of other columns $\mathbf{y}_{-c}$ as $\mathbf{v}_{\mathbf{y}_{-c}} = \frac{1}{C-1} \sum_{c' \neq c} \mathbf{v}_{y_{c'}}$. The multi-label classifier captures column-interdependency, co-occurrence of column-concepts. We also take binary relevance approach for multi-label classification. Though each classifier is trained separately, column interdependencies are captured because input variable consists of several columns.

Title-Column Potential The title-column potential $\phi_{ty}(t, y_c)$ measures the similarity between a title concept t and a column concept y_c. We define the potential function as

$$\phi_{ty}(t, y_c) = \exp\{-d(t, y_c)\}, \tag{2}$$

where d denotes the distance between embedded vectors corresponding to title concept t and column concept y_c such as $||\mathbf{v}_t - \mathbf{v}_{y_c}||$. We expect the potential makes regularization of column concepts because the potential takes a low value when column concepts are not close.

3.4 Knowledge Graph Embedding

In computing the values of potential functions, previous semantic table annotation methods conduct a graph traversal on knowledge graphs, which is inefficient. To reduce the computational cost of potential functions, we use knowledge graph embedding methods (Bordes et al. 2013; Trouillon et al. 2016; Nickel and Kiela 2017). Our model enjoys two positive effects from using knowledge graph embedding in computing potentials: (1) it is not necessary to search on the knowledge graph because the information in the knowledge graph can be translated into vector representations, and (2) missing link between concepts in the knowledge graph can be found by using knowledge graph embedding.

In general, there are several methods for knowledge graph embedding such as TransE (Bordes et al. 2013) and ComplEx (Trouillon et al. 2016). While they can be useful, we employ Poincarè embedding (Nickel and Kiela 2017) because it maintains the semantic hierarchy of the knowledge graph by embedding in a small hyperbolic space $\mathcal{B}^d$ ($\mathcal{B}^d = \{\mathbf{v} \in \mathbb{R}^d | \, ||\mathbf{v}|| < 1\}$). This enables the hierarchy to be captured with small dimensions and facilitates computation of the potentials on the basis of the embedding vectors $\mathbf{v}$.

3.5 Feature Extraction

In our model, the parameters of potential functions are trained in a supervised manner where inputs are column cells and outputs are the corresponding concepts of the column. Our model extracts features from column cells, trains multi-label classifiers to obtain potential function parameters, and uses them in the potential maximization for prediction. We design two types of feature extractions for NE- and literal-columns. The features designed for NE-columns differ from those designed for literal-columns because the characteristics of NE- and literal-columns are naturally different. We introduce two feature extraction approaches.

Features for Named-Entity-Columns For NE-columns, the features are designed on the basis of entity-linking and knowledge graph embedding (Section 3.4). In Limaye's method, each cell corresponds to a concept. We assume each cell is related to one or more concepts.

1. Link cells to concept candidates: find concept candidates $\{e_{r,c}\}_{r=1}^{R} \subset V$ corresponding to cell values $\{x_{r,c}\}_{r=1}^{R}$.
2. Obtain vector representations: transform candidates $\{e_{r,c}\}_{r=1}^{R}$ into embedded vectors with knowledge graph embedding $\{\mathbf{v}_{r,c}\}_{r=1}^{R}$ ($\mathbf{v}_{r,c} \in \mathcal{B}^d$).
3. Calculate statistics: calculate the mean and standard deviation of the set of embedded vectors $\{\mathbf{v}_{r,c}\}_{r=1}^{R}$.

We first calculate the textual similarities between the concept lemmas and cell values to find concept candidates $\{e_{r,c}\}_{r=1}^{R}$ [2]. Lemmas are generally defined as representations of concepts in the knowledge graph. For instance, the concept "<United Kingdom>" has lemmas such as "UK" and "Great Britain." We obtain vectors corresponding to the concept candidates by using knowledge graph embedding. Then we compute the mean and standard deviation of the set of embedded vectors. Finally, we concatenate the mean and the standard deviation vector and use it as the feature vector for the column.

It is natural that we use the mean (and standard deviation) of embedding vectors as the feature vector of a column given the analogy that the mean of word embedding vectors is often used as the feature vector of a sentence or documents (Arora, Liang, and Ma 2017). The mean of word embedding vectors is often used as the feature vector of a sentence even if embedding approaches based on long short-term memory appeared. Since the characteristics of columns are similar to those of bag-of-words, we use the mean of embedding vectors as a feature vector.

[2] While arbitrary textual similarity can be used to compute candidates similarity, we found that edit-distance works well. The threshold of determining the concept candidates is computed as $\alpha \max(len(lemma), len(cellvalue))$. The value of α is determined by cross-validation, and it is 0.5 in our experiments.

Features for Literal-Columns For literal-columns, we compute several numerical statistics of the column cells. Since literal-columns may also have textual information such as measurement units (e.g., "20 s"), we also use several textual features such as the frequency of each letter and the length of each string. Our feature extraction for literal-columns comprises three steps:

1. Calculate numerical statistics: we extract numerical values from cells $\mathbf{x}_c$ and calculate several statistics $\mathbf{v}_{\mathrm{num}}$ from the values.
2. Calculate textual features: we calculate several textural features $\mathbf{v}_{\mathrm{txt}}$ such as the frequency of each letter.
3. Concatenate features: we concatenate numerical statistics $\mathbf{v}_{\mathrm{num}}$ with textual features $\mathbf{v}_{\mathrm{txt}}$.

4 Training

4.1 Training for Knowledge Graph Embedding

As mentioned in Section 3.4, we use Poincarè embedding (Nickel and Kiela 2017) as knowledge graph embedding. We train network parameters in Poincarè embedding to obtain vector representations of concepts. We initialize each vector corresponding to the concept and minimize the loss function with maintaining the hierarchical relationship to obtain the vector representations of concepts.

4.2 Training Parameters of the Potential Functions

Since multi-label classifiers $\mathbf{f}_{\theta_1}$ and $\mathbf{g}_{\theta_2}$ which are used in potential functions, are parametrized by the parameters θ_1 and θ_2, we first train these parameter values using the annotated tables by

$$\min_{\theta_1,\theta_2} \sum_{T\in\mathcal{T}} \sum_{c} l_1(y_c, \mathbf{f}_{\theta_1}(\mathbf{x}_c)) + l_2(y_c, \mathbf{g}_{\theta_2}(\mathbf{y}_{-c})), \qquad (3)$$

where $\mathcal{T} = \{T_1, T_2, \dots T_n\}$ are the annotated tables used for optimizing the parameters, and l_1 and l_2 are the loss functions corresponding to multi-label classifiers $\mathbf{f}_{\theta_1}$ and $\mathbf{g}_{\theta_2}$. Since there are no interactions between the parameters, we optimize the parameters separately as ordinary multi-label classification problems.

5 Prediction

As mentioned in the Problem 1, semantic table annotation problem can be reformulated as the potential maximization of our model:

$$\max_{\mathbf{y},t} \prod_{c} \phi_{yx}(y_c, \mathbf{x}_c)\phi_{yy}(y_c, \mathbf{y}_{-c})\phi_{ty}(t, y_c). \qquad (4)$$

Given table contents as the observed variables $\mathbf{X}$, we maximize the potential to obtain optimal column concepts $\mathbf{y}$. To this end, we consider two approaches: naive approach and approximation approach.

5.1 Naive Approach

A naive approach to maximizing the potential calculates the potentials for all combinations of latent variable candidates. Since the number of candidate sets grows exponentially with respect to the number of columns, taking this naive approach in the real world is unrealistic.

5.2 Approximation Approach

Since the naive approach is intractable, we propose an approximate approach for the potential maximization. First, we apply the Gibbs sampling technique to obtain optimal parameter distributions in a realistic time. Second, to further reduce the computational time of the Gibbs sampling, we propose a heuristics that reduces the number of latent variable candidates.

Gibbs Sampling The joint posterior distribution of the latent variables of our model is given by

$$p(\mathbf{y}, t|\mathbf{X}) \propto \prod_{c} \phi_{yx}(y_c, \mathbf{x}_c)\phi_{yy}(y_c, \mathbf{y}_{-c})\phi_{ty}(t, y_c)p(\mathbf{y})p(t).$$

Instead of directly sample from the joint posterior, which is intractable, we use the approximated sampling algorithms such as Markov Chain Monte Carlo (MCMC) method. The MCMC method samples each latent variable from its conditional posterior distribution with the remaining variables fixed to their current values and approximates the posterior distribution from the sampled latent variables. Namely, the MCMC method samples a column concept y_c from its conditional posterior distribution $p(y_c|\mathbf{X}, \mathbf{y}_{-c}, t)$ and the title concept t from $p(t|\mathbf{X}, \mathbf{y}_c)$, vice versa.

The complete procedure for the Gibbs sampling by the MCMC method is shown in the Algorithm 1.

Algorithm 1 Approximate prediction with Gibbs sampling

Input: Table $\mathbf{X}$, trained parameters θ,
number of iterations I
Output: approximate distributions of each column concept y_c and table concept t

Initialize each column concept $y_c^{(0)} \sim p(y_c|\mathbf{x}_c)$ and a title concept $t^{(0)} \sim p(t|\mathbf{y}^{(0)})$
for iteration $i = 1, 2, \dots, I$ **do**
  $y_1^{(i)} \sim p(y_1|\mathbf{X}, y_2^{(i-1)}, y_3^{(i-1)}, \dots, y_C^{(i-1)}, t^{(i-1)}, \theta)$
  $y_2^{(i)} \sim p(y_2|\mathbf{X}, y_1^{(i)}, y_3^{(i-1)}, \dots, y_C^{(i-1)}, t^{(i-1)}, \theta)$
  $\vdots$
  $y_C^{(i)} \sim p(y_C|\mathbf{X}, y_1^{(i)}, y_2^{(i)}, \dots, y_{C-1}^{(i)}, t^{(i-1)}, \theta)$
  $t^{(i)} \sim p(t|\mathbf{y}^{(i)})$
end for

Pruning Latent Variable Candidates Even under the MCMC technique, sampling a latent variable value from its conditional posterior is computationally heavy because the number of candidate values for each latent variable corresponds to the number of concepts in the knowledge graph and is enormous ($\approx$ 17M in YAGOv3). Therefore, we limit the candidate concepts for each latent variable into those

who appeared in the training datasets reducing the time for whole Gibbs sampling process.

6 Computational Complexity

Table 2 summarizes the computational complexities of the training and prediction phase in our proposed method and other popular approaches (Limaye's method (Limaye, Sarawagi, and Chakrabarti 2010) and Pham's method (Pham et al. 2016)). Our method has smaller complexities in both the training phase and the prediction phase compared to other methods. In the table, $|V|$ is the number of concepts in the knowledge graph, $|V^+|$ is the number of concepts appeared in the training data, $|\mathcal{T}|$ is the number of tables in the training dataset, C and R are the mean numbers of columns and rows in a table of the training dataset, C_t and R_t are the numbers of columns and rows in the target table, I is the number of iterations for prediction, and d is the dimensions of embedded space by knowledge graph embedding.

Since Pham's method computes similarities between all the pairs of columns in the training phase, its computational time grows quadratically with the number of available columns ($= |\mathcal{T}|C$), resulting in $O(|\mathcal{T}|^2C^2R)$. Limaye's method assigns concepts for the columns, the column-column interactions, and the cells. The method needs $O(C_t \log |V|)$ for the column concept estimation, $O(C_t^2)$ for the column-column interaction estimation, and $O(C_tR_t \log |V|)$ for the cell concept estimation, resulting in $O(C_tR_t \log |V| + C_t^2)$ for a iteration.

Table 2: Computational complexities for training and prediction.

Method	Train	Prediction
Limaye	$O(\|\mathcal{T}\|\|V\|CR)$	$O(I(C_tR_t \log \|V\| + C_t^2))$
Pham	$O(\|\mathcal{T}\|^2C^2R)$	$O(\|\mathcal{T}\|CC_tR_t)$
Proposed	$O(\|\mathcal{T}\|\|V^+\|Cd)$	$O(I\|V^+\|C_td)$

7 Evaluation and Discussion

7.1 Setting

Data We evaluated the predictive and computational performance of our method and conventional methods on real data. The dataset we used consists of 183 human-annotated tables (with 781 NE-columns and 4,109 literal-columns) obtained from the UCI Machine Learning repository (Lichman 2013). Tables in the UCI Machine Learning repository contain 37,527 rows on average. As a knowledge graph that contains the canonical concepts for the annotation, we used WordNet (Miller 1995), a popular English lexical database ($|V| = 117,798$).

Evaluation Metrics We measured the predictive performance of semantic table annotation in three metrics: MAP@k, nDCG@k, and sim@k. MAP and nDCG are general metrics for ranking quality (Clarke et al. 2008). Sim@k is computed as the average of *Wup similarities* (Wu and Palmer 1994) between the ground truth concepts and the predicted concepts. Wup similarity computes the similarity between two concepts based on graph-based distance in a knowledge graph. In the experiments, we set $k = 5$ to follow the experimental settings of the previous work.

Methods We compared the following methods including the state-of-the-art baselines and our proposed method:

1. (Limaye) Limaye's method (Limaye, Sarawagi, and Chakrabarti 2010) with/without column interdependency. Since the exact inference for their model is intractable, we used the Gibbs sampling technique for the inference as they did in their experiments.
2. (Pham) Pham's method (Pham et al. 2016) without column interdependency. We used several numerical and textual similarities as column features such as Jaccard index of trigrams and Kolmogorov-Smirnov test score. We also used the random forest classifier.
3. (Hybrid) A hybrid approach that combines Limaye's method and Pham's method. Limaye's method was used to model column-interdependencies in NE-columns. For literal-columns, Pham's method (Pham et al. 2016) is applied.
4. (Proposed) Proposed method with/without column interdependency.

Detailed Settings of Our Model As the knowledge graph embedding method used in the potential functions of our model, we used Poincarè embedding (Nickel and Kiela 2017) and embedded WordNet into the five-dimensional space. We used several numerical statistics as literal-column features: minimum, maximum, median, mean, standard deviation, the 3rd moment, skewness, and kurtosis of numerical values in cells. In addition to numerical features, we also used textual features such as the frequency of each letter. We used a binary relevance method with random forest classifiers as multi-label classifiers in the potential functions. The parameters of potential functions were trained by the training dataset and multi-label binary loss function. We set the number of iterations in Gibbs sampling to 300 because we observed the convergence at that point and further iterations did not affect the accuracy of the model.

7.2 Predictive Performance for the Whole Table

We first evaluated the performance of semantic table annotation on the whole table contents (e.g., including both title concepts and column concepts). We compared the proposed method with/without column interdependency (CID) with Limaye's method (Limaye, Sarawagi, and Chakrabarti 2010). Table 3 shows the results of annotation on tables that contain both NE- and literal-columns. Our proposed approach with column interdependency had the best performance for all metrics, which indicates that column-interdependence consideration is effective in semantic table annotation. From this result, we confirm the first (our model is versatile) and the second contribution (our model is accurate) of our work.

Table 3: Result of annotation on the whole tables.

Method	MAP@5	nDCG@5	Sim@5
Hybrid	0.225	0.291	0.480
Proposed (w/o CID)	0.351	0.413	0.537
Proposed (w/ CID)	**0.464**	**0.741**	**0.635**

Named-Entity-Columns First, we focused on the performance of semantic table annotation for named-entity columns (NE-columns).

As shown in Table 4, our method outperformed the other methods on all metrics. This is because our method uses a supervised approach with rich features that consists of scale-invariant statistics and the knowledge graph embedding. Since Limaye's method relies on a knowledge graph strongly, it may perform poorly when the domain of the knowledge graph does not equal to the domain of the table data. In contrast, our method takes a supervised manner and trains multi-label classifiers with various features, which alleviates the domain difference. While Pham's method is also a supervised approach that obtains the column concept by using the similarity-based classifier, our method outperformed Pham's method because the features in Pham's method do not capture the characteristics of NE-columns effectively.

Table 4: Result of annotation with NE-columns without column-interdependency consideration.

Method	MAP@5	nDCG@5	Sim@5
Limaye	0.021	0.077	0.411
Pham	0.196	0.328	0.470
Proposed	**0.314**	**0.365**	**0.526**

Literal-Columns Next, we focused on the performance of semantic table annotation for literal-columns. Since Limaye's (Limaye, Sarawagi, and Chakrabarti 2010) method does not handle literal-columns, we evaluated two methods: our method and Pham's method (Pham et al. 2016). As shown in Table 5, our method outperformed Pham's method on all metrics.

There are several concerns in annotating literal-columns. One concern is the potential difference in measurement units (e.g., centimeters vs. inches). For instance, some measurements of human height might be in centimeters while others might be in inches. Our method is robust to the differences in measurement units because it extracts unit-invariant features from column values. In contrast, Pham's method suffers from such differences because it relies on the similarities of the columns, which might take completely different values for different measurement units. Another concern is feature-engineered values in a table, such as one-hot-encoding and log-scale-transformation. For instance, categorical columns (e.g., occupation) are often encoded in a one-hot-vector representation to improve the performance of machine learning. Without explicit side information that indicates each dimension of the vector is derived from a column, it is difficult to annotate each dimension of the vector correctly. How to annotate these feature-engineered columns is one of the interesting future research direction.

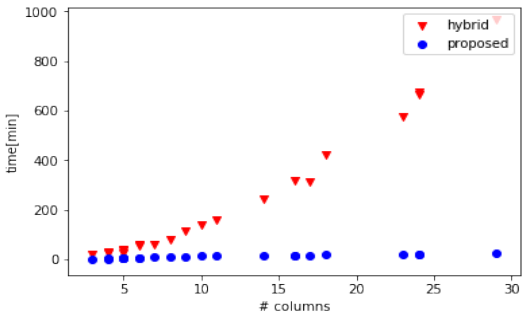

Figure 4: Computational performance for prediction with column interdependency that confirms Table 4.

Table 5: Result of annotation with literal-columns without column-interdependency consideration.

Method	MAP@5	nDCG@5	Sim@5
Pham	0.270	0.330	0.467
Proposed	**0.397**	**0.430**	**0.590**

7.3 Computational Performance

We evaluated the computational performance of our method and a conventional method (a hybrid approach combined Limaye's and Pham's methods), which confirms the computational complexities as mentioned in Table 2. We used Gibbs sampling for inference in both approaches to facilitate comparison of the computational performance. As shown in Figure 4, our method outperformed the hybrid approach and we confirm the third contribution (our model is efficient) of our work.

8 Conclusion

We proposed a novel approach for table data annotation with latent probabilistic model and multi-label classifiers. By employing highly predictive multi-label classifiers in potential functions, our approach enjoyed three advantages over previous approaches:

1. It is versatile: potential functions based on multi-label classifiers in the probabilistic model allow supporting various types of data such as numerical values.
2. It is accurate: integrating multi-label classifier with probabilistic model improves the predictive performance of semantic annotation for table data.
3. It is efficient: potential functions based on multi-label classifiers reduce the computational complexity for annotating tables.

Extensive experiments showed the superiority of our approach over state-of-the-art approaches on real-data, which

includes human-annotated 183 tables collected from UCI Machine Learning Repository.

References

Arora, S.; Liang, Y.; and Ma, T. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, 2787–2795.

Chu, X.; Morcos, J.; Ilyas, I. F.; Ouzzani, M.; Papotti, P.; Tang, N.; and Ye, Y. 2015. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 1247–1261. ACM.

Clarke, C. L.; Kolla, M.; Cormack, G. V.; Vechtomova, O.; Ashkan, A.; Büttcher, S.; and MacKinnon, I. 2008. Novelty and diversity in information retrieval evaluation. In *SIGIR*, SIGIR '08, 659–666. New York, NY, USA: ACM.

Deng, D.; Jiang, Y.; Li, G.; Li, J.; and Yu, C. 2013. Scalable column concept determination for web tables using large knowledge bases. *PVLDB* 6(13):1606–1617.

Lichman, M. 2013. UCI machine learning repository.

Limaye, G.; Sarawagi, S.; and Chakrabarti, S. 2010. Annotating and searching web tables using entities, types and relationships. *PVLDB* 3(1-2):1338–1347.

Miller, G. A. 1995. Wordnet: A lexical database for english. *Commun. ACM* 38(11):39–41.

Mulwad, V.; Finin, T.; and Joshi, A. 2013. Semantic message passing for generating linked data from tables. In *ISWC*, 363–378. Springer.

Neumaier, S.; Umbrich, J.; Parreira, J. X.; and Polleres, A. 2016. Multi-level semantic labelling of numerical values. In *ISWC*, 428–445. Springer.

Nickel, M., and Kiela, D. 2017. Poincaré embeddings for learning hierarchical representations. In *NIPS*. Curran Associates, Inc. 6338–6347.

Pham, M.; Alse, S.; Knoblock, C. A.; and Szekely, P. 2016. Semantic labeling: A domain-independent approach. In *ISWC*, 446–462. Springer.

Rümmele, N.; Tyshetskiy, Y.; and Collins, A. 2018. Evaluating approaches for supervised semantic labeling. *arXiv preprint arXiv:1801.09788*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *ICML*, 2071–2080.

Tsoumakas, G., and Katakis, I. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining* 3(3).

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

Wu, Z., and Palmer, M. 1994. Verbs semantics and lexical selection. In *ACL*, ACL '94, 133–138. Stroudsburg, PA, USA: Association for Computational Linguistics.

Zhang, M., and Chakrabarti, K. 2013. Infogather+: Semantic matching and annotation of numeric and time-varying attributes in web tables. In *SIGMOD*, SIGMOD '13, 145–156. New York, NY, USA: ACM.

Zhang, Z. 2017. Effective and efficient semantic table interpretation using tableminer+. *Semantic Web* 8(6):921–957.