Sato: Contextual Semantic Type Detection in Tables

Dan Zhang
UMASS Amherst
dzhang@cs.umass.edu

Yoshihiko Suhara
Megagon Labs
yoshi@megagon.ai

Jinfeng Li
Megagon Labs
jinfeng@megagon.ai

Madelon Hulsebos
The HEINEKEN Company
mmhulsebos@gmail.com

Çağatay Demiralp
Megagon Labs
cagatay@megagon.ai

Wang-Chiew Tan
Megagon Labs
wangchiew@megagon.ai

ABSTRACT

Detecting the semantic types of data columns in relational tables is important for various data preparation and information retrieval tasks such as data cleaning, schema matching, data discovery, and semantic search. However, existing detection approaches either perform poorly with dirty data, support only a limited number of semantic types, fail to incorporate the table context of columns or rely on large sample sizes for training data. We introduce Sato, a hybrid machine learning model to automatically detect the semantic types of columns in tables, exploiting the signals from the table context as well as the column values. Sato combines a deep learning model trained on a large-scale table corpus with topic modeling and structured prediction to achieve support-weighted and macro average F1 scores of 0.925 and 0.735, respectively, exceeding the state-of-the-art performance by a significant margin. We extensively analyze the overall and per-type performance of Sato, discussing how individual modeling components, as well as feature categories, contribute to its performance.

PVLDB Reference Format:
Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çağatay Demiralp and Wang-Chiew Tan. Sato: Contextual Semantic Type Detection in Tables. *PVLDB*, 13(11): 1835-1848, 2020.
DOI: https://doi.org/10.14778/3407790.3407793

1. INTRODUCTION

Many data preparation and information retrieval tasks including data cleaning, integration, discovery and search rely on the ability to accurately detect data column types. Automated data cleaning uses transformation and validation rules that depend on data types [23, 39]. Schema matching for data integration leverages data types to find correspondences between data columns across tables [38]. Similarly, data discovery benefits from detecting the types of data columns in order to return semantically relevant results for user queries [9, 10]. Discerning the semantics of table values helps aggregate information from multiple tabular data sources. Search engines also rely on the detection of semantically relevant column names to extend support to tables [48].

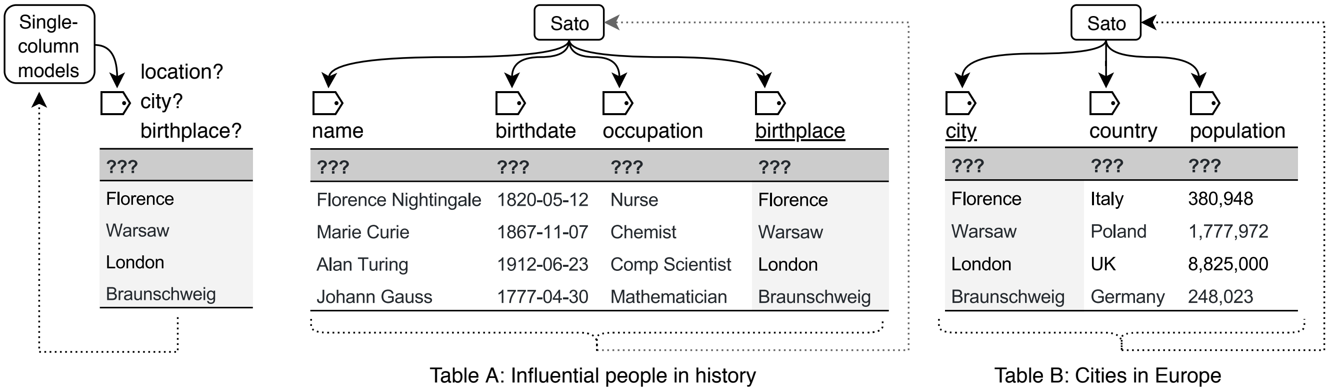
We can consider two categories of types for table columns: atomic and semantic. Atomic types like `boolean`, `integer`, and `string` provide basic, low-level type information about a column. On the other hand, semantic types like `location`, `birthDate`, and `name`, convey finer-grained, richer information about column values. Detecting semantic types can be a powerful tool, and in many cases may be essential for enhancing the effectiveness of data preparation and analysis systems. In fact, commercial systems such as Google Data Studio [17], Microsoft Power BI [32], Tableau [44], and Trifacta [46] attempt to detect semantic types, typically using a combination of regular expression matching and dictionary lookup. While reliable for detecting atomic types and simple, well-structured semantic types such as credit card numbers or e-mail addresses, these rule-based approaches are not robust enough to process dirty or missing data, support only a limited variety of types, and fall short for types without strict validations. However, many tables found in legacy enterprise databases and on the Web have column names that are either unhelpful (cryptic, abbreviated, malformed, etc.) or missing altogether.

In response, recent work [22] introduced Sherlock, a deep learning model for semantic type detection trained on massive table corpora [21]. Sherlock formulates semantic type detection as a multi-class classification problem where classes correspond to semantic types. It leverages more than 600K columns from real-world tables for learning with a multi-input feed-forward deep neural network, providing state-of-the-art results.

While Sherlock represents a significant leap in applying deep learning to semantic typing, it suffers from two problems. First, it under-performs for types that do not have a sufficiently large number of samples in the training data. Although this is a known issue for deep learning models, it nevertheless restricts Sherlock's application to underrepresented types, which form a long tail of data types appearing in tables at large. Second, Sherlock uses only the values of a column to predict its type, without considering the column's context in the table. Predicting the semantic type of a column based solely on the column values, however, is an under-determined problem in many cases.

Consider the example in Figure 1: for a column that contains 'Florence,' 'Warsaw,' 'London,' and 'Braunschweig' as values, `location`, `city`, or `birthPlace` could all be reason-

Proceedings of the VLDB Endowment, Vol. 13, No. 11
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3407790.3407793



???
Florence
Warsaw
London
Braunschweig

name	birthdate	occupation	birthplace
???	???	???	???
Florence Nightingale	1820-05-12	Nurse	Florence
Marie Curie	1867-11-07	Chemist	Warsaw
Alan Turing	1912-06-23	Comp Scientist	London
Johann Gauss	1777-04-30	Mathematician	Braunschweig

Table A: Influential people in history

city	country	population
???	???	???
Florence	Italy	380,948
Warsaw	Poland	1,777,972
London	UK	8,825,000
Braunschweig	Germany	248,023

Table B: Cities in Europe

Figure 1: Two actual tables with unknown column types (Table A and Table B) from the VizNet corpora. The last column of Table A and the first column of Table B have identical values: 'Florence,' 'Warsaw,' 'London,' and 'Braunschweig.' However powerful, a prediction model based solely on column values (i.e., single-column prediction) cannot resolve the ambiguity to infer the correct semantic types, `birthplace` and `city`. Sato incorporates signals from table context and performs a multi-column type prediction to help effectively resolve ambiguities like these and improve the accuracy of semantic type predictions.

able semantic types for the column. It can be hard to resolve such ambiguities using only column values because the semantic types also depend on the *context* of the table. Continuing with the example, it is highly likely that the column's type would be `birthPlace` if it came from Table A since the table contains biographical information about influential personalities. However, the same column in Table B would be more likely to have the type `city`, as the table's other columns present information about European cities.

In this paper, we introduce Sato (**S**em**A**ntic **T**ype detection with table c**O**ntext), a hybrid machine learning model that incorporates table contexts for semantic type prediction. Sato combines topic modeling [4] and structured learning [25] together with single-column type prediction based on the Sherlock model. Similar to earlier work [22], we consider 78 common semantic types and use the WebTables dataset from the VizNet corpus [21] to train our model.

We summarize our main contributions below:

1. Sato significantly outperforms the state-of-the-art in semantic type prediction, increasing the macro and support-weighted F1 scores by as much as 14.4% and 5.3%, respectively. Through a comparative analysis, we show that Sato's performance gains are primarily due to improved predictions for underrepresented semantic types in the long tail. To facilitate future research and applications, we have released Sato as an open-source project along with an online demo at `https://github.com/megagonlabs/sato`.

2. Sato achieves this high prediction accuracy using a novel hybrid model that regulates semantic type prediction using signals from the global context (values from the entire table) and the local context (predicted types of neighboring columns,) demonstrating the effectiveness of incorporating table context into semantic type detection.

3. Sato introduces a new extensible architecture for type detection with modules for modeling single columns, global context, and local context. One can easily plug in a different single-column model while keeping the rest intact. We demonstrate this extensibility in Section 6 by replacing the default single-column predictor in Sato with BERT [12].

2. PROBLEM FORMULATION

We formulate the problem of semantic type prediction as multi-class classification, each class corresponding to a predefined semantic type. We consider the training data as a set of tables. Let $c_1, c_2, \ldots, c_m$ be the columns of a given table and $t_1, t_2, \ldots, t_m$ be the true semantic types of these columns, where $t_i \in \mathcal{T}$, the set of labels for possible semantic types considered (e.g., `city`, `country`, `population`). Similarly, let Φ be a feature extractor function that takes a single column c_i and returns an n-dimensional feature vector Φ_i. One approach to semantic typing is to learn a mapping $f_{\text{single}} : \Phi^n \to \mathcal{T}$ from values of single columns to semantic types. We refer to this model as *single-column prediction.* The Sherlock [22] model falls into this category.

In Sato, to make the best use of table contexts and resolve semantic ambiguity with single-column predictions, we formulate the problem as *multi-column prediction.* A multi-column prediction model learns a mapping $f_{\text{mult}} : \Phi^{n\times m} \to \mathcal{T}^m$ from the entire table (a sequence of columns) to a sequence of semantic types. This formulation allows us to incorporate table context into semantic type prediction in two ways. First, we use features generated from the entire table as table context. For example, the column values 'Italy,' 'Poland,' ... and '380,948,' '1,777,972,' ... are also used to predict the semantic type of the first column in Table B (in Figure 1.) Second, we can jointly predict the semantic types of columns from the same table. Again, for Table B, with the joint prediction the predicted types `country` and `population` of neighboring columns would help to make a more accurate prediction for the first column.

3. MODEL

Table context As demonstrated in Figure 1, the contextual information of a column can be used to resolve ambiguities and improve the semantic type prediction for the column. To this end, we identify two basic types of context that collectively characterize the context of a table column: global context and local context. We define the global context for a column to be the set of all the cell values in the table. In this sense, all the columns in a given table have the same global context. We show in Section 3.2 how the global context can be used to compute a global descriptor effectively capturing the intent of a table. We define the local context of a column as the set of independently predicted semantic types of the neighboring columns in the same table. Local context can be used to resolve semantic type ambiguities when combined with single-column predictions. The scope of such a local neighborhood is flexible and can be adjusted based on the desired trade-off between model performance and model complexity. In this work, we restrict the local neighborhood to immediately adjacent columns. We demonstrate in Section 3.3 how local context can be effectively used to improve the semantic type detection accuracy through structured predictions.

Next, we will show how Sato effectively captures contextual signals from both global and local sources using a hybrid machine learning model. It has two modeling components: (1) A topic-aware prediction component that estimates the *intent* (a global descriptor) of a table using topic modeling and extends the single-column prediction model with an additional topic subnetwork. (2) A structured output prediction model that combines the topic-aware predictions for all m columns and performs multi-column joint semantic type prediction. Figure 2 illustrates the high-level architecture of Sato. We next discuss each Sato component and its implementation in detail.

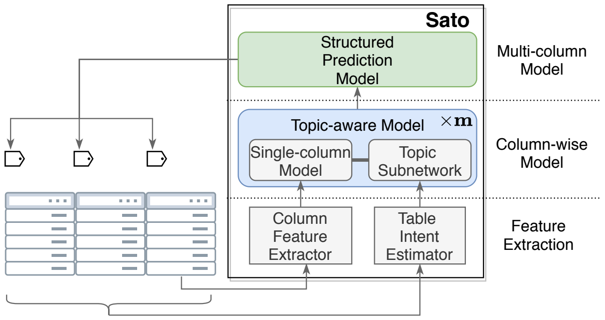


Figure 2: In Sato, the *topic-aware* module extends single-column models with additional topic subnetworks, incorporating a context modeling table intent into the model. The *structure prediction* module then combines the topic-aware results for all m columns, providing the final semantic type prediction for the columns in the table.

3.1 Single-column prediction model

As shown in Figure 2, Sato's topic-aware module is built on top of a single-column prediction model that uses a deep neural network. We first provide a brief background on deep learning and a description of the single-column model.

Deep learning Deep learning [27] is a form of representation learning that uses neural networks with multiple layers. Through simple but non-linear transformations of the input representation at each layer, deep learning models can learn representations of the data at varying levels of abstractions that are useful for the problem at hand (e.g., classification, regression). Deep learning combined with the availability of massive table corpora [8, 21] presents opportunities to learn from tables in the wild [19]. It also presents opportunities to improve existing approaches to semantic type detection as well as other research problems related to data preparation and information retrieval. Although prior research has used shallow neural networks for related tasks (e.g., [28]), it is only more recently that Hulsebos et al. [22] developed Sherlock, a large-scale deep learning model for semantic typing.

Deep learning for type prediction Sato builds on single-column predictions by using column-wise features and employs an architecture that allows any single-column prediction model to be used. In this work, we choose Sherlock as our single-column prediction model due to its recently demonstrated performance. The column-wise features used in Sato include character embeddings (Char), word embeddings (Word), paragraph embeddings (Para), as well as column statistics (e.g., mean, std) (Stat.)

A multi-layer subnetwork is applied to the column-wise features to compress high-dimensional vectors into compact dense vectors, with the exception of the Stat feature set, which consists of only 27 features. The output of the three subnetworks is concatenated to the statistical features, forming the input to the primary network. Then, in the primary network two fully-connected layers (ReLU activation) with BatchNorm and Dropout layers are applied before the output layer. The final output layer, which includes a softmax function, generates confidence values (i.e., probabilities) for the 78 semantic types.

3.2 Topic-aware prediction model

The first component of Sato is a topic-aware prediction module. This module first creates a vector representation for the global context of a given table by computing a topic vector from the values of the entire table. The topic-aware prediction module then feeds this topic vector as input to the column-wise prediction model. The column-wise prediction model extends the neural network model above with an additional subnetwork in order to take topic vectors as input. We next discuss how taking the global context of a table into account in semantic type prediction can help resolve ambiguities.

Table semantics Tables are collections of related data entities organized in rows. To incorporate table semantics in our model, we build on intuition by Venetis et al. [48] that a user constructing a table has a particular *intent* or schema in mind. We extend this intuition and argue that semantic types of the columns in a table can be considered a meaningful expression (or utterance) of the user intent. Each column of the table partially fulfills the intent by describing one attribute of the entities. As illustrated in Figure 3a, the intent of a table is a latent component determining the semantic types of the columns in the table, which in turn generates the column values. We refer to the set of all column values in a table as *table values*.

Thus, being able to accurately infer the table intent can help to improve the prediction of column semantics. Table captions or titles usually capture table intent. For example, in Figure 1, Table A intends to provide biographical information about influential personalities in history and Table B talks about geographical information about cities in Europe. However, as with column semantics, a clear and well-

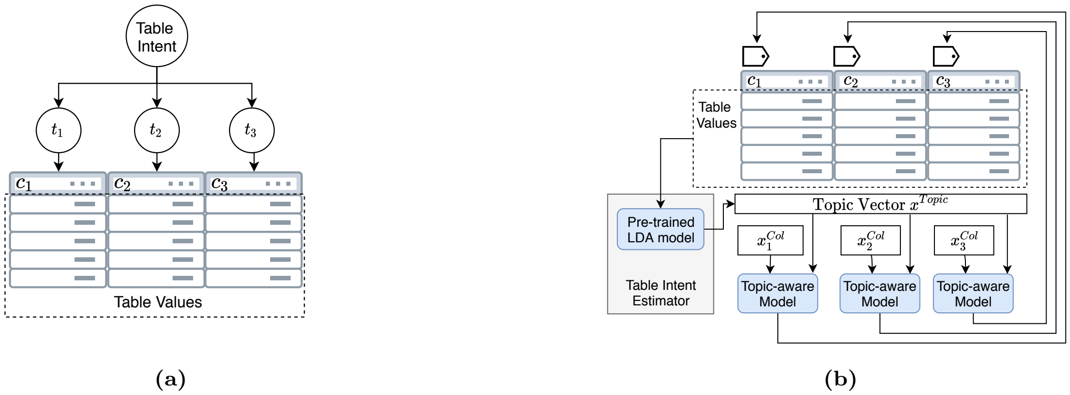


Figure 3: SATO's topic-aware modeling is based on the premise that every table is created with an *intent* in mind and that the semantic types of the columns in a table are expressions of that intent with thematic coherence. In other words, (a) the intent of a table determines the semantic types of the columns in the table, which in turn generate the column values, acting as latent variables. (b) SATO estimates the intent of a given table with a topic vector obtained from a pre-trained LDA model and combines it with the local evidence from per-column values using a deep neural network.

structured description of intent is not always available in real-world tables. Therefore we need to estimate the table intent without relying on any header or meta information.

SATO estimates a table's intent by mapping its values onto a low-dimensional space. Each of these dimensions corresponds to a "topic," describing one aspect of a possible table intent. The final estimation is a distribution over the latent topic dimensions generated using topic modeling approaches. Next, we provide a brief background on topic models and explain how SATO extracts topic vectors from tables and feeds them to topic-aware models.

Topic models Finding the topical composition of textual data is useful for many tasks, such as document summarization or featurization. Topic models [4] aim to automatically discover thematic topics in text corpora and discrete data collections in an unsupervised manner. Latent Dirichlet allocation (LDA) [6] is a simple yet powerful generative probabilistic topic model, widely used for quantifying thematic structures in text. LDA represents documents as mixtures of latent topics and each latent topic as a distribution over words. Although LDA was originally applied to text corpora, many variants have been developed to discover thematic structures in non-textual data (e.g., [5, 13, 51].)

Table intent estimator We use an LDA model to estimate a table's intent as a topic-vector, treating values of each table as a "document." As illustrated in Figure 3b, we implement the table intent estimator as a pre-trained LDA model. It takes table values as input and outputs a fixed-length vector named "table topic vector" over the topic dimensions. For SATO, we pre-train an LDA model with 400 topic dimensions on public tables that have had their headers and captions removed.

The topics are generated during training according to the data's semantic structure, so they do not have predefined meanings. However, by looking at the representative semantic types associated with each topic, we found some examples with good interpretations. For example, topic # 192 is closely associated with the semantic types "origin, nationality, country, continent, and sex" and thus possibly captures aspects about personal information, while topic # 264 corresponds to "code, description, create, company, symbol" and can be interpreted as a business-related topic. Detailed topic analysis can be found in Section 5.5.

Learning and prediction Figure 3b shows how topic-aware models take the values in a table topic vector as additional features for both learning and prediction. We augment the single-column neural network model with an additional subnetwork to take topic vectors as input and then append its output before feeding into the primary network. The topic-aware model will learn not only relationships between the input column and its type but also how the column type correlates to the table-level contextual information.

3.3 Structured prediction model

We have described how SATO captures the global context of a column by computing a topic vector for the entire table and passing it to the single-column model as additional input. Incorporating only the global context into the model may not be sufficient, however, as the topic-aware model does not directly take the relationships among the semantic types of neighboring columns (i.e., local context) into account. Therefore, we incorporate local context using a structured output prediction model which comprises the second component of SATO.

Through preliminary analysis, we confirm that certain pairs of semantic types co-occur in tables much more frequently than others. For example, in a WebTables sample, the most frequent pair `city` and `state` co-occurs 4 times more often than the tenth most frequent pair `name` and `type` (detailed co-occurrence statistics available in Section 4.1). Such inter-column relationships show the value of "local" contextual information from surrounding columns in addition to the "global" table topic. SATO models the relationships between columns through pairwise dependencies in a graphical model and performs table-wise prediction using structured learning techniques. Although the notion of local context is not limited to immediate neighbors, SATO only models pairwise relations between adjacent columns because of its simplicity, efficiency, and empirical accuracy. We leave

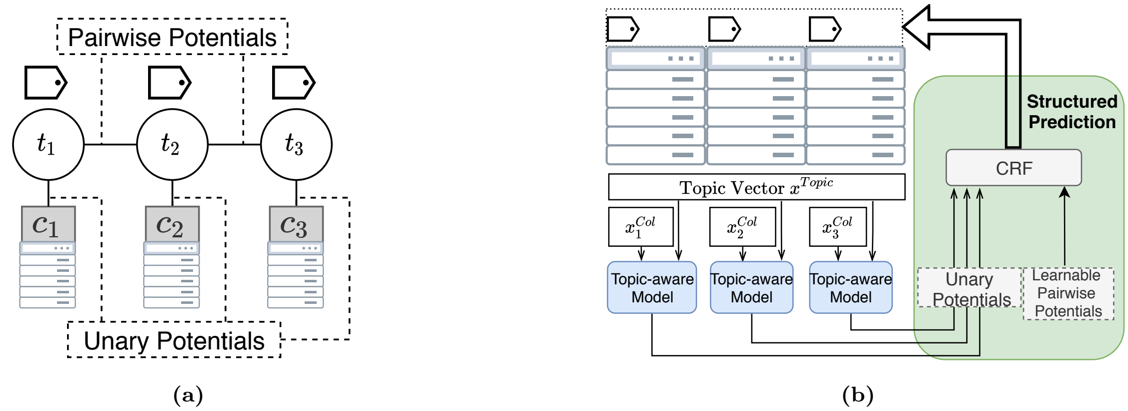


Figure 4: (a) Sato uses a linear-chain CRF to model the dependencies between columns types given their values. (b) For each column, Sato plugs in the column-wise prediction scores for each type as the unary potentials of the corresponding node in the CRF model. Then Sato learns the pairwise potential through backpropagation updates using stochastic gradient descent, maximizing the posterior probability $P(\mathbf{t}|\mathbf{c})$. Although we choose to use predictions from topic-aware models in the current implementation, the Sato architecture is flexible to support unary potentials from arbitrary column-wise models.

the study of the broader local context, which can be modeled using high-order graphical models (further discussed in Section 6), as future work.

Structured output learning In addition to semantic type detection, many other prediction problems such as named entity extraction, language parsing, and image segmentation have spatial or semantic structures that are inherent to them. Such structures mean that predictions of neighboring instances correlate to one another. Structured learning algorithms [3], including probabilistic graphical models [24] and recurrent neural networks [20, 42], model dependencies among the values of structurally linked variables such as neighboring pixels or words to perform joint predictions.

A conditional random field (CRF) [25] is a discriminative undirected probabilistic graphical model and a popular technique for structured learning with successful applications in labeling, parsing and segmentation problems across domains. Similar to Markov random fields (MRFs) [15, 24], the exact inference for general CRFs is intractable but there are special structures such as linear-chains that allow exact inference. There are also several efficient approximate inference algorithms based on message passing, linear-programming relaxation, and graph cut optimization for CRFs with general graphs [25].

Modeling column dependencies Sato uses a linear-chain CRF to explicitly encode the inter-column relationship while still considering features for each column. We encode the output of a column-wise prediction model (i.e., predicted semantic types of the columns) and the combinations of semantic types of columns in the same table as CRF parameters. As shown in Figure 4a, in the CRF model, each variable t_i represents the type of a column with corresponding column values c_i as the observed evidence. Variables representing the types of adjacent columns are linked with an edge. Given a *sequence* of columns $\mathbf{c}$ in a table, the goal is to find the best *sequence* of semantic types $\mathbf{t}$, which provides the largest conditional probability $P(\mathbf{t}|\mathbf{c})$.

The conditional probability can be written as a normalized product of a set of real-valued functions. Following the convention, we refer to these functions in log scale as "potential functions." *Unary potential* $\psi_{\text{UNI}}(t_i, c_i)$ captures the likelihood of predicting type t_i based on the content of the corresponding column c_i. *Pairwise potential* $\psi_{\text{PAIR}}(t_i, t_j)$ represents the "coupling degree" between types t_i and t_j.

We use a linear-chain CRF, where the conditional distribution is defined by the unary prediction potentials and pairwise potentials between adjacent columns:

$$P(\mathbf{t}|\mathbf{c}) = \frac{1}{Z(\mathbf{c})} \exp\left(\sum_{i=1}^{m} \psi_{\text{UNI}}(t_i, c_i) + \sum_{i=1}^{m}\sum_{j=i+1}^{m} \psi_{\text{PAIR}}(t_i, t_j)\right),$$

where

$$Z(\mathbf{c}) = \sum_{\mathbf{t}} \exp\left(\sum_{i=1}^{m} \psi_{\text{UNI}}(t_i, c_i) + \sum_{i=1}^{m}\sum_{j=i+1}^{m} \psi_{\text{PAIR}}(t_i, t_j)\right)$$

is an input-dependent normalization function.

Unary potential functions We use unary potentials to model the probability of a semantic type given the column content. In other words, the unary potential of a semantic type for a given column can be considered the probability of that semantic type based on the values of the column. The architecture of Sato supports using estimates of any valid column-wise prediction model as unary potentials. In this work, we obtain the unary potentials of the semantic types for a given column from the output of our topic-aware prediction model, which uses both table-level topic vector and column features as input. Using the examples of Figure 1, we expect that the highlighted column with values like 'Florence,' 'Warsaw,' would have high unary potential scores for location-related semantic types such as `location`, `city`, and `birthplace`. In other words, the unary potentials calculate column-wise prediction scores, which are used to *select* semantic type candidates for each column.

Pairwise potential functions Pairwise potentials capture the relationship between the semantic types of two columns in the same table. These relationships can be parameterized with a $|\mathcal{T}| \times |\mathcal{T}|$ matrix P, where $\mathcal{T}$ is the set of all possible types and P_{ij} $(= \psi_{\text{PAIR}}(t_i, t_j))$ is a weight parameter for the "coupling degree" of semantic types t_i and t_j

in adjacent columns. Such a coupling degree can be approximated by the co-occurrence frequency. We expect the pairwise weight of two semantic types to be proportional to their frequency of co-occurrence in adjacent columns. Pairwise potential weights in our CRF model are trainable parameters, updated by gradient descent. Through the training step, we expect that SATO updates the CRF parameters so that frequently co-occurred pairs like (`city`, `country`) and (`occupation`, `birthplace`) have higher pairwise potential scores. Thus, the trained model can resolve the disambiguate issue (shown in Figure 1) by using pairwise potentials and achieves context-aware predictions.

Learning and prediction We use the following objective function to train a SATO model. The objective function is the log-likelihood of semantic types of columns in the same table:

$$\log P(\mathbf{t}|\mathbf{c}) = \sum_{i=1}^{m} \psi_{\text{UNI}}(t_i, c_i) + \sum_{i=1}^{m} \sum_{j=i+1}^{m} \psi_{\text{PAIR}}(t_i, t_j) - \log Z(\mathbf{c}).$$

Here, the normalization term Z sums over all possible semantic type combinations. To efficiently calculate Z, we can use the forward-backward algorithm [37], which uses dynamic programming to cache intermediate values while moving from the first to the last columns. After the training phase, as shown in Figure 4b, SATO performs holistic type prediction with learned pairwise potential and unary potential provided by topic-aware prediction. To obtain prediction results, we conduct maximum a posteriori (MAP) inference of semantic types:

$$\hat{\mathbf{t}} = \underset{\mathbf{t}}{\operatorname{argmax}} \log P(\mathbf{t}|\mathbf{c}) = \underset{\mathbf{t}}{\operatorname{argmax}} \left(\sum \psi_{\text{UNI}} + \sum \sum \psi_{\text{PAIR}} \right).$$

$Z(\mathbf{c})$ does not affect argmax since it is a constant with respect to $\mathbf{t}$. Then we use the Viterbi algorithm [49] to calculate and store partial combinations with the maximum score at each step of the column sequence traversal, avoiding redundant computation.

4. EVALUATION

We compare SATO and its two basic variants obtained by ablation with the state-of-the-art Sherlock [22] implemented as the BASE method. We omit comparisons with matching-based algorithms, decision-tree-based semantic typing since they are outperformed by Sherlock as demonstrated in [22].

4.1 Datasets

We evaluate the effectiveness of the proposed models on the WebTables corpus from VizNet [21] and restrict ourselves to the relational web tables with valid headers that appear in the 78 semantic types. These types resulted from a selection process [22] from the T2Dv2 Gold Standard[1], which describes 237 DBpedia properties frequently occurring in the WebTables corpus. To avoid filtering out columns with slight variation in capitalization and representation, we convert all column headers to a "canonical form" before matching. The canonicalization process starts with trimming content in parentheses. We then convert strings to lower case, capitalize words except for the first (if there are more than one word) and concatenate the results into a single string. For example, strings 'YEAR,' 'Year' and 'year (first occurrence)' will all have the canonical form 'year,' and 'birth place (country)' will be converted to 'birthPlace.'

Since we formulate semantic typing as a multi-column type detection problem, we extract 80K tables, instead of columns, from a subset of the WebTables corpus as our dataset $\mathcal{D}$. The canonicalized column headers act as the groundtruth labels for semantic types. To help evaluate the importance of incorporating table semantics, we also create a filtered version $\mathcal{D}_{mult}$ with 33K tables. We filter out singleton tables (those containing only one column) since they lack context as defined in this paper. We then conduct 5-fold cross-validation where we use 80% of the tables for training and a held-out set (20%) for evaluation in each iteration.

Figure 5 shows the count of each semantic type in the dataset $\mathcal{D}$. The distribution is clearly unbalanced with a long tail. Single-column models tend to perform poorly on the less-common types that comprise the long-tail. By effectively incorporating context, SATO significantly improves prediction accuracy for those types.

To better understand relationships between the semantic types of columns in the same table, we conduct a preliminary analysis on the co-occurrence patterns of types. Figure 6, shown in log-scale for readability, reports the frequencies of selected pairs of semantic types occurring in the same table. Most frequently co-occurring pairs include (`city`, `state`), (`age`, `weight`), (`age`, `name`), (`code`, `description`).

4.2 Feature extraction

We use the public Sherlock feature extractors[2] to extract the four groups of base features, CHAR, WORD, PARA and STAT, generating a feature vector with 1587 dimensions for each column in a table. Those features have been proven effective for semantic type detection and provide good coverage of the granularity spectrum, ranging from character-level distribution features to global statistics. In addition, the WORD and PARA features take advantage of powerful pre-trained word and paragraph embeddings which enable a better understanding of natural language contents.

To make a fair comparison, these *base* features were used by both baseline methods and proposed methods in the experiments. To generate table topics as introduced in Section 3.2, we train an LDA model that captures the mapping from table values to the latent topic dimensions. Since LDA is an unsupervised model, we only need the vocabulary (i.e., set of all cell values) of the tables without any headers or semantic annotation. We convert numerical values into strings and then concatenate all values in the table to form a "document" for each table. Using the gensim [41] library, we train an LDA model with 400 topics on a held-out dataset of 10K tables. With the pre-trained LDA, we extract topic vectors for tables using values from the entire table as input. Every table has a single topic vector, shared across columns.

4.3 Model implementation

We implement the multi-input neural network introduced in [22] using PyTorch [33] as the BASE single-column model. Throughout the experiments discussed here, we train the BASE neural network model for 100 epochs using the Adam optimizer with a learning rate of $1e-4$ and a weight decay rate of $1e-4$.

[1] `http://webdatacommons.org/webtables/goldstandardV2.html`

[2] `https://github.com/mitmedialab/sherlock-project`

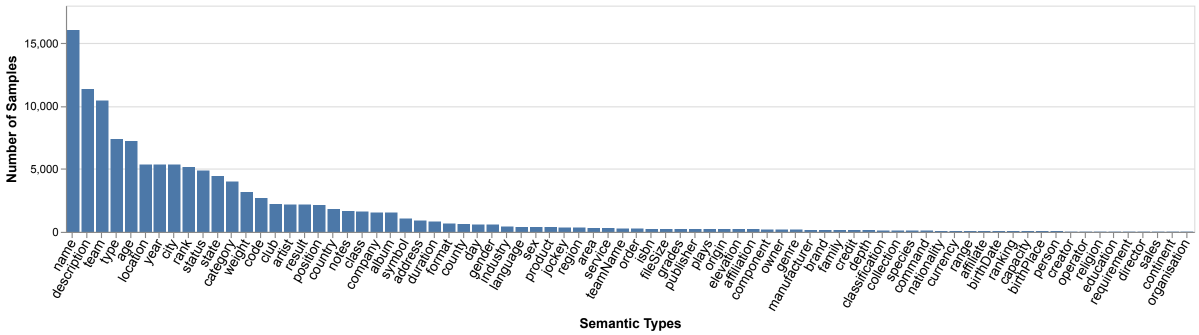


Figure 5: Counts of the 78 semantic types in the dataset $\mathcal{D}$ form a long-tailed distribution. Sato improves the prediction accuracy for the types with fewer samples (those in the long-tail) by effectively incorporating table context.

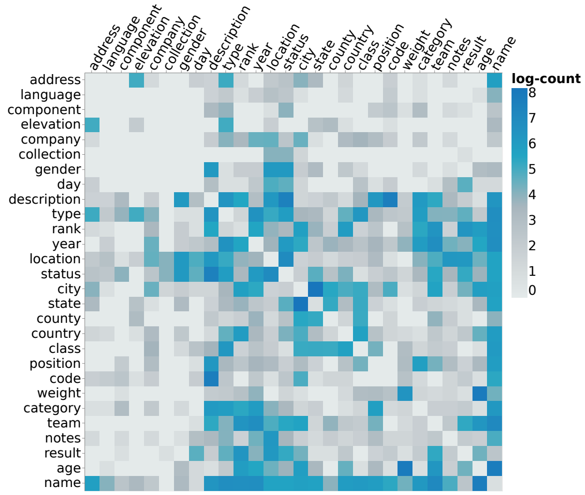


Figure 6: Co-occurrence frequencies in log scale for a selected set of types. Certain pairs like (`city`, `state`) or (`age`, `weight`) appear in the same table more frequently than others. There are non-zero diagonal values as tables can have multiple columns of the same semantic type.

For topic-aware prediction in Sato, the table topic features go through a separate subnetwork with an architecture identical to the subnetworks of the Base feature groups. Before going into the primary network, the outputs of all four subnetworks are concatenated with Stat to form a single vector. We train Sato's CRF layer with a batch size of 10 tables, using the Adam optimizer with a learning rate of $1e-2$ for 15 epochs. We initialize the pairwise potential parameters of the CRF model with the column co-occurrence matrix calculated from a held-out set of the WebTables corpus. We set the CRF unary potentials for columns to be their normalized topic-aware prediction score.

4.4 Evaluation metrics

We measure the prediction performance on each target semantic type by calculating $F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$. Since the semantic type distribution is not uniform, we report two types of average performances using the support-weighted F_1 and macro average F_1. The support-weighted F_1 score is the average of per-type F_1 values weighted by support (sample size in the test set for the respective type) and reflects the overall performance. The macro average F_1 score is the unweighted average of the per-type F_1 scores, treating all types equally, and is therefore more sensitive to types with small sample sizes compared to support-weighted F_1.

5. RESULTS

Table 1 reports improvements of the Sato variants over the Base method on both the dataset $\mathcal{D}_{mult}$, which includes only tables with more than one column, and the complete dataset $\mathcal{D}$. We implemented Base using features and neural network structure of the Sherlock [22] model. On multi-column tables, Sato improves the macro average F_1 score by 0.093 (14.4%) and the support-weighted F_1 score by 0.046 (5.3%) compared to the single-column Base. When evaluated on all tables we still see a 0.064 (9.3%) improvement on macro average F_1 score and 0.035 (4.0%) improvement on support-weighted F_1, although these scores are diluted by the inclusion of tables without valid table context. The results confirm that Sato can effectively improve the accuracy of semantic type prediction by incorporating contextual information embedded in table semantics.

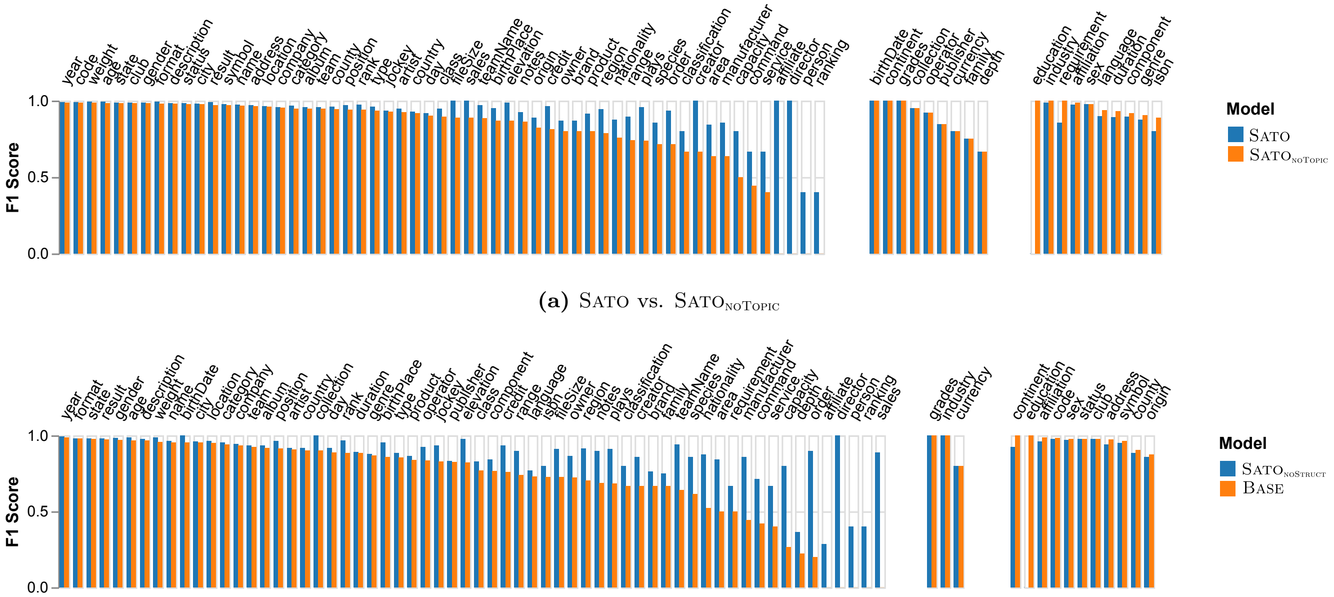
We also evaluate the variants of Sato with single components: Sato$_{\text{NoStruct}}$ only performed topic-aware prediction using table values and Sato$_{\text{NoTopic}}$ conducted structured prediction using Base output as unary potential without considering table topic features. As shown in Table 1, both Sato$_{\text{NoStruct}}$ and Sato$_{\text{NoTopic}}$ provide improvements over the Base model but are outperformed by the combined effort in Sato. The results indicate that the structured prediction model and the topic-aware prediction model make use of different pieces of table context information. We note that there are always larger improvements on macro average F_1 scores than support-weighted F_1 scores, suggesting that a significant amount of Sato's improvements come from boosting accuracy for the less represented types. To better understand the influence of techniques used in Sato, we next perform a per-type evaluation for both Sato components on multi-column tables.

5.1 Topic-aware prediction

Figure 7 shows the per-type comparison of F_1 scores between models with and without the topic-aware component. More specifically, Figure 7a compares the full Sato against Sato without table values (i.e., Sato$_{\text{NoTopic}}$,) and Figure 7b compares Sato$_{\text{NoStruct}}$ (only topic-aware model) against Base. Including information in table values improved 59 out of 78

Table 1: Performance comparison of the methods across the datasets $\mathcal{D}_{mult}$ (multi-column only) and $\mathcal{D}$ (the full dataset) Numbers are the average values over a 5-fold cross validation. ± denotes 95% CI. () shows the relative improvements in percentage over Base. We conducted statistical tests using paired t-test with Bonferroni correction for multiple comparisons. Sato, $\text{Sato}_{\text{noStruct}}$, $\text{Sato}_{\text{noTopic}}$ perform significantly better than Base ($p < .005$ in all metrics.) Sato performs significantly better than $\text{Sato}_{\text{noStruct}}$ ($p < .005$ in all metrics) and $\text{Sato}_{\text{noTopic}}$ ($p < .005$ on $\mathcal{D}_{mult}$ and n.s. on $\mathcal{D}$.)

	Multi-column tables $\mathcal{D}_{mult}$		All tables $\mathcal{D}$	
	Macro average F_1	Support-weighted F_1	Macro average F_1	Support-weighted F_1
Base	0.642 ±0.015	0.879 ±0.002	0.692 ±0.007	0.867 ±0.003
Sato	**0.735** ±0.022 (14.4%↑)	**0.925** ±0.003 (5.3%↑)	**0.756** ±0.011 (9.3%↑)	**0.902** ±0.002 (4.0%↑)
$\text{Sato}_{\text{noStruct}}$	0.713 ±0.025 (11.0%↑)	0.909 ±0.002 (3.5%↑)	0.746 ±0.011 (7.8%↑)	0.891 ±0.003 (2.8%↑)
$\text{Sato}_{\text{noTopic}}$	0.681 ±0.016 (6.6%↑)	0.907 ±0.002 (3.2%↑)	0.711 ±0.006 (2.9%↑)	0.884 ±0.002 (2.0%↑)



(a) Sato vs. $\text{Sato}_{\text{noTopic}}$

(b) $\text{Sato}_{\text{noStruct}}$ vs. Base

Figure 7: F_1 scores for each type obtained with (blue) and without (orange) topic-aware prediction. (a) compares Sato and $\text{Sato}_{\text{noTopic}}$ (Sato without the topic-aware module), (b) compares $\text{Sato}_{\text{noStruct}}$ (Base with topic) and Base, showing improvements on the majority of types. The effect is significant for many underrepresented types.

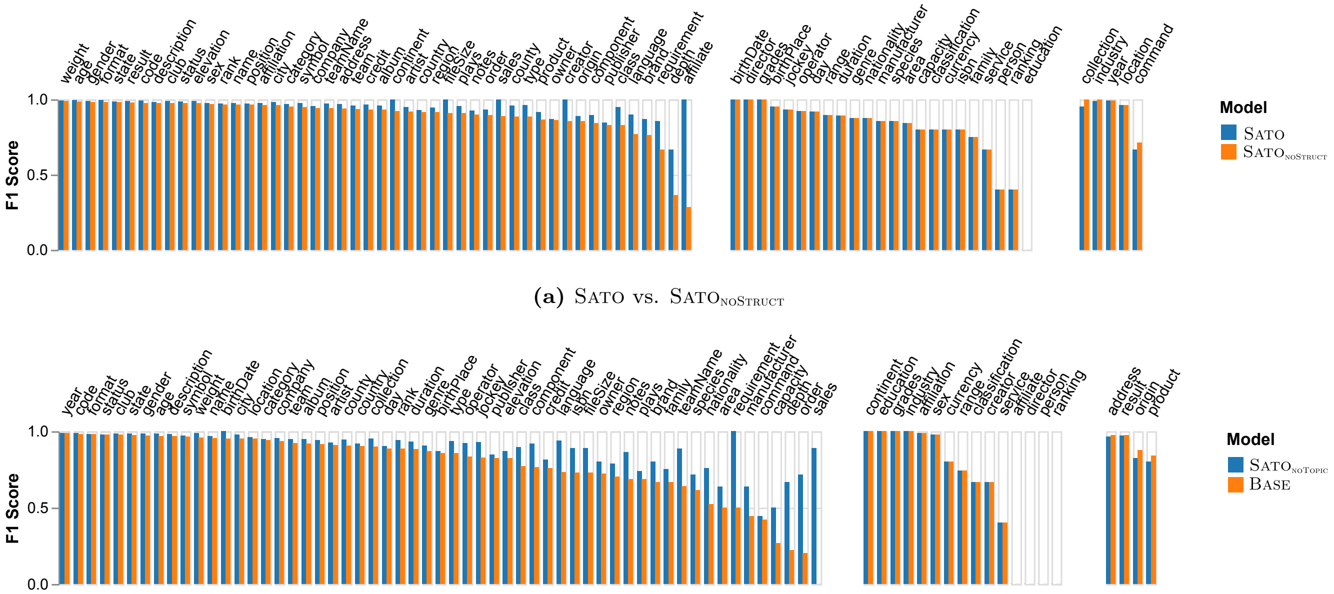
semantic types for $\text{Sato}_{\text{noTopic}}$ with 9 types getting equal and 10 types getting worse performances. Similarly, $\text{Sato}_{\text{noStruct}}$ improves the performance for 64 types and decreases it for 11 types. The prediction performance stays unchanged for 3 types. We also see significant improvements in the previously "hard" semantic types with small support size. The types with the highest accuracy increases, `affiliate`, `director`, `person`, `ranking`, and `sales`, all come from the fifteen least represented types as shown in Figure 5. This shows incorporating table values effectively alleviates the problem of lacking training data for the rare types.

5.2 Structured prediction

To evaluate the contribution of structured prediction, we compare Sato with its variant without structured prediction, $\text{Sato}_{\text{noStruct}}$ (Figure 8a). Similarly, we compare the performance of $\text{Sato}_{\text{noTopic}}$ (structured prediction directly on Base output) with that of Base (Figure 8b). Base is improved on 50 types and $\text{Sato}_{\text{noStruct}}$ is improved on 59 types. For a subset of rare types (e.g., `depth`, `sales`,) the prediction accuracy is dramatically improved. While for others (e.g., `person`, `director`,) there is no noticeable improvement as with topic-aware prediction. This shows structured prediction is less effective in boosting the accuracy of rare types compared to topic-aware prediction. However, at the same time, both the number of types that get worse accuracy (4 and 5 respectively) and the drop in F_1 scores for those types are smaller with structured prediction as compared to topic-aware prediction. Enforcing table-level context can be too aggressive sometimes, leading to worse performance for certain types. Through modeling relationships between columns, the structured prediction module in Sato "salvages" some overly aggressive predictions. Qualitative analysis in Section 5.7 further confirms this effect. In conclusion, structured multi-column prediction model, with or without topic modeling, outperforms the column-wise models.

5.3 Efficiency

We show that the Sato model successfully improves prediction accuracy by introducing the topic-aware features and



(a) Sato vs. Sato$_{\text{noStruct}}$

(b) Sato$_{\text{noTopic}}$ vs. Base

Figure 8: F$_1$ scores for each type obtained with (blue) and without (orange) structured prediction (a) compares Sato and Sato$_{\text{noStruct}}$ (Sato without the structured prediction module), (b) compares Sato$_{\text{noTopic}}$ (Base with structured prediction) and Base, showing improvements on the majority of types. Although the improvements on long-tail types are less significant compared to the topic-aware model in Figure 7, fewer types get worse predictions (shown in the right panels). Structured prediction can correct mispredictions by directly modeling column relationships.

the CRF layer. However, the additional components may cause additional time cost. To evaluate the efficiency of Sato, we repeated the training and prediction procedures for 5 times and measured the training and prediction time of Base and Sato on the multi-column dataset $\mathcal{D}_{mult}$. The training data contains 26K tables and the test data contains 6.4K tables. For further investigation on the cost of the topic-aware features and the CRF layer, we separately measured the time for training the main model, and the time for training the CRF layer. We use the same hyperparameters used in the experiment (described in 4.3) for both of the models for a fair comparison. The experiment was conducted on a single machine with 2.1GHz CPUs (64 cores) and 512GB RAM. Table 2 summarizes the average training and prediction time for those models.

From the results, we confirm that adding the topic-aware features and the CRF layer increases approximately 81 s and 367 s for training time, respectively. We would like to emphasize that we do not need to retrain a model unless we obtain a significant amount of additional training data. Thus, we consider that the difference is not critical. On average, Sato takes +1.4 s than Base to generate predictions for all 6.4K tables in the test set of $\mathcal{D}_{mult}$, which is 0.2 ms per table. We believe the overhead will be mostly unnoticeable in practice, and the average prediction time per table (0.8 ms) can support the interactive use of Sato. Furthermore, in practice deployment, pre-trained column-wise models can be reused to shorten the average end-to-end training time.

5.4 Feature importance

To better understand the influence of the different feature groups, we perform permutation importance [1] analysis on Base and Sato variants. For each fitted model and a specific feature group, we take the input tables and shuffle by only swapping features in the specified feature group with randomly selected tables. Such feature mismatches will cause less accurate predictions. Shuffling crucial features will break the strong relationships between input and output, leading to a significant drop in accuracy. We took the average of the normalized drop in F$_1$ scores over five random trials as the feature importance measurement. Figure 9 shows that for Base and Sato$_{\text{noTopic}}$, the Word and Char feature groups are the most important feature groups, matching the conclusions in [22]. Considering the global context, the additional *Topic* feature group has comparable or greater importance than Word and Char, espeacilly for the macro average F$_1$ metric. This confirms the help of table values information on less-represented types.

Table 2: Average training and prediction time over 5 trials $\mathcal{D}_{mult}$. ± denotes 95% CI. Training time for the column-wise features (Features) and the CRF layer (Structured) is reported separately.

	Training time [s]		Prediction time [s]
	Features	Structured	
Base	596.9 ± 9.2	N/A	3.8 ± 0.04
Sato	678.5 ± 15.1	366.9 ± 66.8	5.2 ± 0.06

5.5 Topic interpretation

We conduct qualitative analyses on the LDA model to investigate how the model captures semantics from each ta-

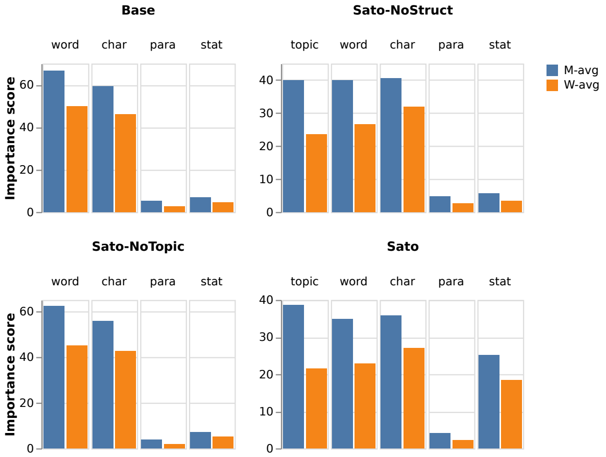


Figure 9: Importance scores for the feature categories obtained by measuring the drop in macro-avg (M-avg) and weighted-avg (W-avg) aggregated F_1 values from permutation experiments. *Topic* features are the most important feature category with respect to the macro average F_1 score in the full SATO model, providing additional evidence for the contribution of topic modeling in predicting underrepresented semantic types.

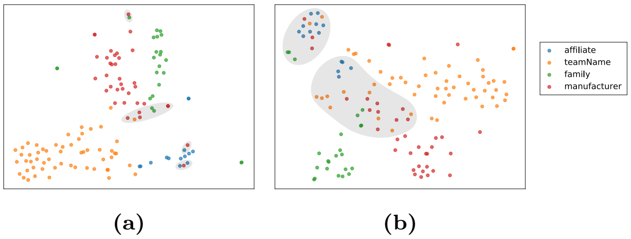
Table 3: Examples of the topics learned by the LDA model, semantic types associated with each topic obtained by using a saliency metric, and our interpretation for each topic.

Topic	Top-5 semantic types	Interpretation
192	origin, nationality, country, continent, sex	person
99	affiliate, class, person, notes, language	person
394	religion, family, address, teamName, publisher	person, book
264	code, description, creator, company, symbol	business

ble and provides contextual information to SATO. To obtain the topic distribution of each semantic type, we calculate the average topic distribution based on the topic distributions θ_i of the i-th table that contains the semantic type. For each topic, we chose top-k semantic types as representative semantic types by the probability of the topic.

We find that some topics had "flat" distributions where most semantic types have almost the same probabilities. Since these topics are not very useful for classifying semantic types, we compute a saliency score for each topic and sort the topics by their saliency. Our saliency score averages the probabilities of the top-k semantic types for each topic.

Table 3 shows the top-5 salient topics and the representative semantic types. Following the standard approach in topic model analysis [4, 6], we manually devise an interpretation for each topic. For example, topic dimension #192 and #99 are activated by personal information in table values, whereas #264 is closely related to business tables. These examples demonstrate that semantic space learned using LDA could capture intent information from tables.



(a) (b)

Figure 10: Two-dimensional visualizations of column embeddings by (a) SATO$_{\text{NoStruct}}$, and (b) Sherlock. Colors denote semantic types. Gray-colored regions are manually added to emphasize the areas of "ambiguity"in the column embeddings. SATO$_{\text{NoStruct}}$ appears to separate similar semantic types better.

5.6 Column embeddings (Col2Vec)

To verify how the table intent features help the SATO model capture the table semantics, we analyze and compare the embedding vectors from the final layer of the SATO model and the baseline Sherlock model as *column embeddings.* We can consider these embeddings as *column embeddings* since the final layer combines input signals to compose semantic representations. For comparison, we used the final layer of the single-column prediction model of SATO, before the CRF layer. Therefore, we assume that the Table Intent features account for the difference in the embeddings.

Following prior examples (e.g., [52]), we analyze column embeddings of the test columns used in the experiments. We use t-SNE [47] to reduce the dimensionality of the embedding vectors to two and then visualize them using a two-dimensional scatterplot. To embed vectors of the two methods in a common space, we fit a single t-SNE model for all data points, and then visualize major semantic types that are related to organizations (**`affiliate`**, **`teamName`**, **`family`**, and **`manufacturer`**) to investigate how the SATO model with the Table Intent features can distinguish columns of those ambiguous semantic types.

Figure 10 shows the visualization of embedding vectors of SATO and Sherlock. With Sherlock, the column embeddings of each semantic type partially form a cluster, but some clusters are overlapped compared to the column embeddings by SATO. In Figure 10 (a), we observe a clearer separation between the organization-related semantic types with little perturbation. The results qualitatively confirm that topic-aware prediction helps SATO distinguish semantically similar semantic types by capturing the table context of an input table. Note that these column embeddings are from the test set, and any label information from these columns was not used to obtain the column embeddings. Thus, we can also confirm that SATO appropriately generalizes and learns column embeddings for these semantic types.

5.7 Qualitative analysis

To better understand how structured prediction further helped SATO with the existence of topic-aware predictions, we conducted qualitative analysis by identifying examples where table-wise prediction "salvages" bad predictions in the column-wise (i.e., BASE and SATO$_{\text{NoStruct}}$) predictions.

Table 4a shows a selected set of example tables from the test sets where the incorrect predictions from BASE are corrected by applying structured prediction using our trained

Table 4: Examples of the mispredictions that are corrected by performing a structured prediction using the linear-chain CRF.

(a) Corrected tables from Base predictions

Table ID	True Columns	Base (w/o structured prediction)	Sato$_{\text{NoTopic}}$ (w/ structured prediction)
6299	code, name, city	symbol, team, city	code, name, city
898	company, location	name, city	company, location
2816	product, language	name, notes	product, language
4575	symbol, company, isbn, sales	symbol, name, isbn, duration	symbol, company, isbn, sales
5712	type, description	weight, name	type, description
3865	year, teamName, age	year, city, weight	year, teamName, age

(b) Corrected tables from Sato$_{\text{NoStruct}}$ predictions

Table ID	True Columns	Sato$_{\text{NoStruct}}$ (w/o structured prediction)	Sato (w/ structured prediction)
4289	age, city, country, rank	age, city, team, rank	age, city, country, rank
410	brand, weight	artist, code	brand, weight
5655	code, name, city	club, name, name	code, name, city
4369	day, location, notes	name, location, location	name, location, notes
30	language, name, origin	language, name, description	language, name, origin
4531	rank, name, city	rank, location, location	rank, location, city

CRF layer. For example, with table #4575, the columns `company` and `sales` were incorrectly predicted as `name` and `duration` by the single-column Base model. By modeling inter-column dependencies, Sato$_{\text{NoTopic}}$ correctly predicts the types `company` and `sales`, which tend to co-occur more with surrounding columns `symbol` and `isbn` for tables about books and magazines. Table 4b shows selected examples where Sato$_{\text{NoStruct}}$ made incorrect predictions using table values and was corrected by the use of structured prediction (i.e., Sato). Table #4369 and table #4531 are examples where location-related vocabulary in tables made a large impact. It produced overly aggressive predictions with multiple `location` columns, whereas Sato with the structured inference step successfully corrected one of the columns.

Furthermore, considering surrounding types, structured prediction effectively improves performance for numerical columns like `duration/sales` from table #4575, `age/weight` from table #3865, `code/weight` from table #410.

6. DISCUSSION

Using learned representations Sato's single column prediction module based on Sherlock incorporates four categories of features that characterize different aspects of column values, amassing more than 1.5K feature values per column. However, the availability of large-scale table corpora presents a unique opportunity to develop pre-trained representation models and eschew manual feature extraction. To test the viability of using representation models, we fine-tuned the BERT model [12], a state-of-the-art model for language representation, for our semantic type detection task. Models based on fine-tuning BERT have recently improved prior art on several NLP benchmarks without manual featurization [12, 30, 31]. We trained the BERT model using the default BERT parameters, achieving a support-weighted F1 score of 0.866, which is slightly better than 0.852 achieved by the Sherlock model. This result is promising because a "featurization-free" method with default parameters is able to achieve a prediction accuracy comparable to that of Sherlock. However, our multi-column prediction still outperforms the BERT model by a large margin, indicating the importance of incorporating table context into column type prediction. A promising avenue of future research is to combine our multi-column model with BERT-like pre-trained learned representation models.

Exploiting type hierarchy through ontology In this paper, we consider semantic types without hierarchy. However, it is possible to form natural parent-child relationships between many types. For instance, `country` and `city` are types (subclasses) of `location` and `club` and `company` are types of `organization`. Factoring hierarchical type relations into prediction (e.g., [29, 45]) requires an ontology codifying the type hierarchy and, crucially, additional annotation over training dataset, which can be infeasible to manually carry out for large training datasets such as the one used here. Nevertheless, modeling and predicting hierarchical semantic types can provide richer information for downstream tasks. It can further improve the prediction accuracy, especially for the types with fewer training samples.

High-order CRFs Several studies [11, 26, 36] developed high-order CRF models that implement potential functions that take n ($n > 2$) predictions into account. However, the computational complexity of exact inference steps for training and prediction becomes exponentially expensive: $\mathcal{O}(L^K)$, where L is the input sequence length (i.e., # of columns) and K is the number of states (i.e., # of semantic types.) The computational cost is significantly expensive compared to the original linear-chain CRFs $\mathcal{O}(KL^2)$. As Sato with the linear-chain CRF model significantly improved the performance for the semantic type detection task, we decided not to use the degree of the order for efficiency.

Additionally, we believe that high-order dependency between predictions is not always necessary if we incorporate contextual features into the model. [36] shows that contextual features that take into account surrounding information are more useful than a high-order CRF architecture for named entity recognition tasks. Since table topic features provide table-wise contextual information, we consider the original CRF model with pairwise potential functions as the right choice for improving the model accuracy efficiently.

7. RELATED WORK

Regular expression and dictionary lookup Semantic type detection enhances the functionality of commercial data preparation and analysis systems such as Microsoft Power BI [32], Trifacta [46], and Google Data Studio [17]. These commercial tools typically rely on manually defined rule-based approaches such as regular expression patterns dictionary lookups to detect semantic types. For instance, Trifacta detects around 10 types and Power BI only supports time-related semantic types. Open source libraries such as messytables [14], and csvkit [18] similarly use heuristics to detect a limited set of types.

Ontology-based Prior work, with roots in the semantic web and schema matching literature, provide alternative approaches to semantic type detection. One body of work leverages existing data on the web, such as WebTables [8], and ontologies (or, knowledge bases) such as DBPedia [2], Wikitology [43], and Freebase [7]. Venetis et al. [48] construct a database of value-type mappings, then assign types using a maximum likelihood estimator based on column values. Syed et al. [43] use column headers and values to build a Wikitology query mapping columns to types.

Statistical similarity Several earlier approaches rely on measures of data similarity to match columns with types. Ramnandan et al. [40] first separate numerical and textual column types, then compare column values to those with labels from a dataset using the Kolmogorov-Smirnov (K-S) test and Term Frequency-Inverse Document Frequency (TF-IDF,) respectively. Pham et al. [34] use additional features and tests, including the Mann-Whitney test for numerical data and Jaccard similarity for textual data, to train logistic regression and random forest models.

Synthesized Puranik [35] proposes combining the predictions of "experts," including regular expressions, dictionaries, and machine learning models. More recently, Yan and He [50] introduced a system that, given a search keyword and a set of positive examples, synthesizes type detection logic from open source GitHub repositories. It provides a novel approach to leveraging domain-specific heuristics for parsing, validating, and transforming semantic types.

Learned Another line of prior work employs machine learning, including probabilistic graphical models. Goel et al. [16] split each cell value in a table into tokens and attempted to predict the field and token labels using CRF models with different graph structures capturing dependencies among tokens and fields. For instance, a cell value 'Mountain View, CA' is split into a sequence of tokens 'Mountain', 'View',',', 'CA'. Then a multi-layer CRF model is used to assign labels `cityName`, `cityName`, `symbol`, and `state` for those tokens along with the cell label `place`. This approach requires curating cell- and token-level annotations for training, which is impractical for large-scale table corpora. Furthermore, it has limited robustness over missing, dirty, and heterogeneous data, as well as semantic data types with highly variable formatting. Sato avoids the need for fine-grained token-level annotations and only uses automatically annotated column labels.

Limaye et al. [29] use a Markov random field (MRF) model to annotate values with entities, columns with types, and column pairs with relationships. This approach assumes the existence of a catalog specifying entities, types, and relations between them and relies on good matches between entity lemma and cell text to make accurate predictions of both cell and column types. However, in practice, an accurate catalog can be expensive or impossible to obtain for large corpora or new domains and many tables have missing or noisy (incomprehensible, malformed, etc.) headers. Takeoka et al. [45] extend Limaye et al. [29]'s work with multi-label classifiers to support additional types, including numerical data types, and improve its predictive performance. However, this approach also relies on training data (183 tables) collected through human annotation and its application to massive table corpora can get extremely expensive.

Similar to earlier approaches [16, 29, 45] discussed above, Sato also uses a probabilistic graphical model for structured output prediction. However, in contrast to this earlier work, Sato employs a CRF model to combine the topic-aware predictions of a large-scale deep learning model, leveraging a large number of real-world tables for training. These tables are automatically annotated without resorting to human labeling, which makes Sato easier to extend and scale than prior work using probabilistic graphical models.

Although prior research used shallow neural networks for related tasks (e.g., [28]), Sherlock [22] is the first deep learning model directly applied to semantic type detection for table columns. Trained on a large number of columns, Sherlock uses a multi-input neural network to make type prediction based on features of column values. Sato builds on Sherlock and addresses its two related drawbacks; the low prediction accuracy for underrepresented types and the lack of consideration for table context in prediction.

8. CONCLUSION

Automated semantic typing is becoming more important than ever due to a rapid increase in the demand for better data preparation tools. The semantics of a table column (or any other data source for that matter) are embodied by its context as well as its raw data values. Here, we introduce Sato to automatically detect the semantic types of table columns, leveraging the signals from the table context of columns as well as the data values of columns. Sato combines the power of large-scale deep learning together with structured prediction and topic modeling to achieve a prediction performance that significantly exceeds the state-of-the-art. Through ablation and permutation experiments, we evaluate Sato extensively and show how individual modeling choices as well as feature types contribute to the performance. To facilitate future applications and extended research, we are publicly releasing our trained model and source code for training along with an interactive web application demonstrating Sato's use at `https://github.com/megagonlabs/sato`.

9. ACKNOWLEDGMENTS

We thank Jonathan Engel for suggesting the name Sato and his proofreading help. We also thank Kevin Hu for his help in making the Sherlock source code accessible.

10. REFERENCES

[1] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: a nucleus for a web of open data. *ISWC*, pages 722–735, 2007.

[3] G. Bakır, T. Hofmann, B. Schölkopf, A. J. Smola, and B. Taskar. *Predicting structured data*. MIT press, 2007.

[4] D. M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, 2012.

[5] D. M. Blei and M. I. Jordan. Modeling annotated data. In *SIGIR*, pages 127–134. ACM, 2003.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

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

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

[9] R. Castro Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. Aurum: A data discovery system. In *ICDE*, pages 1001–1012, 04 2018.

[10] R. Castro Fernandez, E. Mansour, A. Qahtan, A. Elmagarmid, I. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. Seeping semantics: Linking datasets using word embeddings for data discovery. In *ICDE*, 04 2018.

[11] N. V. Cuong, N. Ye, W. S. Lee, and H. L. Chieu. Conditional random field with high-order dependencies for sequence labeling and segmentation. *Journal of Machine Learning Research*, 15:981–1009, 2014.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, 2019.

[13] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, volume 2, pages 524–531, 2005.

[14] O. K. Foundation. Messytables · pypi, 2019.

[15] S. Geman and C. Graffigne. Markov random field image models and their applications to computer vision. In *Proceedings of the international congress of mathematicians*, volume 1, page 2, 1986.

[16] A. Goel, C. A. Knoblock, and K. Lerman. Exploiting structure within data for accurate labeling using conditional random fields. In *(ICAI)*, 2012.

[17] Google. Google Data Studio, 2019.

[18] C. Groskopf and contributors. *csvkit*, 2016.

[19] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[20] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[21] K. Hu, N. Gaikwad, M. Bakker, M. Hulsebos, E. Zgraggen, C. Hidalgo, T. Kraska, G. Li, A. Satyanarayan, and Ç. Demiralp. Viznet: Towards a large-scale visualization learning and benchmarking repository. In *CHI*. ACM, 2019.

[22] M. Hulsebos, K. Z. Hu, M. A. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, Ç. Demiralp, and C. A. Hidalgo. Sherlock: A deep learning approach to semantic data type detection. In *KDD*, pages 1500–1508, 2019.

[23] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *CHI*, 2011.

[24] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[25] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[26] T. Lavergne and F. Yvon. Learning the structure of variable-order CRFs: a finite-state perspective. In *Proc. EMNLP '17*, pages 433–439, 2017.

[27] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[28] W.-S. Li and C. Clifton. Semantic integration in heterogeneous databases using neural networks. In *PVLDB*, 1994.

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

[30] Y. Liu. Fine-tune BERT for extractive summarization. *preprint arXiv:1903.10318*, 2019.

[31] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *preprint arXiv:1907.11692*, 2019.

[32] Microsoft. Power BI — Interactive Data Visualization BI, 2019.

[33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

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

[35] N. W. Puranik. A specialist approach for classification of column data. Master's thesis, University of Maryland, Baltimore County, August 2012.

[36] X. Qian, X. Jiang, Q. Zhang, X. Huang, and L. Wu. Sparse higher order conditional random fields for improved sequence labeling. In *Proc. ICML '09*, page 849856, 2009.

[37] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

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

[39] V. Raman and J. M. Hellerstein. Potter's wheel: An interactive data cleaning system. *PVLDB*, 1:381–390, 2001.

[40] S. K. Ramnandan, A. Mittal, C. A. Knoblock, and P. Szekely. Assigning semantic labels to data sources. In *ESWC*, pages 403–417. Springer, 2015.

[41] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *LREC*

Workshop, pages 45–50. ELRA, May 2010.

[42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:9, 1986.

[43] Z. Syed, T. Finin, V. Mulwad, A. Joshi, et al. Exploiting a web of semantic data for interpreting tables. In *WebSci*, 2010.

[44] Tableau. Tableau Desktop, 2019.

[45] K. Takeoka, M. Oyamada, S. Nakadai, and T. Okadome. Meimei: An efficient probabilistic approach for semantically annotating tables. In *AAAI*, pages 281–288, 2019.

[46] Trifacta. Data Wrangling Tools & Software, 2019.

[47] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *JMLR*, 9:2579–2605, 2008.

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

[49] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Information Theory*, 13(2):260–269, 1967.

[50] C. Yan and Y. He. Synthesizing type-detection logic for rich semantic data types using open-source code. In *SIGMOD*, pages 35–50. ACM, 2018.

[51] J. Yuan, Y. Zheng, and X. Xie. Discovering regions of different functions in a city using human mobility and pois. In *KDD*, pages 186–194, 2012.

[52] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014.