Semantic Web 0 (0) 1
IOS Press

Survey on Semantic Table Interpretation

Lahiru de Alwis [a] Achala Dissanayake [a] Manujith Pallewatte [a] Kalana Silva [a] and Uthayasanker Thayasivam [a]

[a] *Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka*
E-mails: lahirud.14@cse.mrt.ac.lk, achala.14@cse.mrt.ac.lk, manujith.14@cse.mrt.ac.lk, kalana.14@cse.mrt.ac.lk, rtuthaya@cse.mrt.ac.lk

Abstract.
The web contains a vast amount of tables which provide useful information across multiple domains. Interpreting these tables contribute to a wide range of Semantic Web applications. Aligning web tables against an ontology to understand their semantics is known as **Semantic Table Interpretation (STI)**. This paper presents a survey on Semantic Table Interpretation(STI). Goal of this paper is to provide an overview of STI algorithms, data-sets used, and their evaluation strategies and critically evaluate prior approaches. In the effort of providing the overview we developed a generic framework to analyze STI algorithms. Using this framework we analyzed the existing algorithms and point out their strengths and weakness. Additionally this enables us to categorize the prior works and be able to point out the key attributes of each categories. Our analysis reveals that search based approaches are better in terms of accuracy and overall completeness, while other categories perform better only in annotating columns with high precision. Also, We present the evaluation methodology utilized in algorithms and discuss the limitations of it while providing suggestions for future improvements. In addition, we point out the design choices in building an STI and their associated trade-offs, which could be of value for the future STI algorithm developers and users.

Keywords: Semantic Web, Semantic Table Interpretation

1. Introduction

The semantic web is considered as a critical component in the modern artificial intelligence process. Extracting information from different data sources in the web is vital for various semantic web applications. Structured data sources, such as web tables take a prominence among them.

The web contains over 100 million tables in formats of web tables[1]. Interpreting these tables is humanly impossible due to its size. Popular approaches map these tables to known Knowledge Bases so that agents can utilize them. This problem domain is known as **Semantic Table Interpretation (STI)**[2].

A knowledge base is essentially a directed labeled graph which comprises of concepts as nodes, and relationship between different concepts as directed edges. Across literature, different terms such as knowledge graph [3, 4], taxonomy[5], catalog[6] and knowledge base[7, 8] are used to identify the knowledge bases. Without the loss of generality we can model all above instances as Ontologies.

In recent years, several algorithms have been proposed for STI. Due to the variations among these algorithms and their inherent complexities, directly comparing them is challenging. In the absence of a survey on these algorithms, this paper aims to provide a common conceptual basis that would bridge the above gap, allowing comprehensive analysis and comparisons between different algorithms. The key contributions of this paper include: (1) a comprehensive survey on STI algorithms; (2) a mathematical formulation of the STI problem; (3) a generalized breakdown of STI problem that can be used to analyze different algorithms in a modular level; (4) a categorized view on recent STI algorithms; and (5) a summarized view on different evaluation metrics and Gold Standards used in literature.

This paper is structured as follows. A comprehensive background to STI and the mathematical formulation are provided in Section 2. Throughout Sections 3 - 5, the elements of the conceptual framework introduced in Section 2 are detailed. In Section 6, a classification of STI algorithms is introduced while ana-

lyzing features and capabilities of each category. Section 7 outlines the evaluation methodology and metrics used in literature and provides a summarized view on the analysis of 3 notable STI algorithms against 2 Gold Standards. Section 8 presents future work and a conclusion of overall content.

2. Background

Tables are used to present data in a structured manner. But they lack the semantic information regarding the data represented in them. In order to grasp the contextual meaning, tables can be aligned against ontologies.

Given a predefined Ontology O, and a table T, the semantic table interpretation problem can be defined as a search problem that searches the space of mapping matrices($\mathcal{M}$) for the matrix M^* which maximizes an objective function Q.

$$M^* = \underset{M \in \mathcal{M}}{\operatorname{argmax}}\, Q(T, O) \tag{1}$$

Definition 2.1. A table T has a cell matrix D, and a header array H. D has p rows and q columns indexed by $1 \leqslant r \leqslant p$ and $1 \leqslant c \leqslant q$ respectively. H is an array of size q, composed of column headers. The text in cell (r, c) will be called D_{rc}. The header text in column c will be called H_c.

Definition 2.2. An entity can be a row, a column, a header, a cell or the table itself. The set of all entities in a table is denoted as A.

Definition 2.3. Ontology O is defined as O = <V, E, L> where V is the set of labeled vertices representing the entities, E is the set of edges representing the relations, which is a set of ordered 2-subsets of V, and L is a mapping from each edge to its label[9].

Definition 2.4. We define C to be the set of all concepts in O. Thus $C = V \cup E$

Definition 2.5. The objective function Q scores the mappings denoted in M by considering the semantic relevance between T and O.

The mapping matrix M is a $|O| \times |A|$-matrix. Each entry in this matrix represents a mapping between particular ontology concept and a table entity. Most STI approaches model M as a binary matrix where each entry indicates a presence or an absence of a relation. Hence the size of the search space is $2^{|A||C|}$. But in most of these cases, table to ontology mapping is assumed to be many to one, which reduces the search space size to $|A+1|^{|C|}$. Given this context, we define a mapping matrix M as follows.

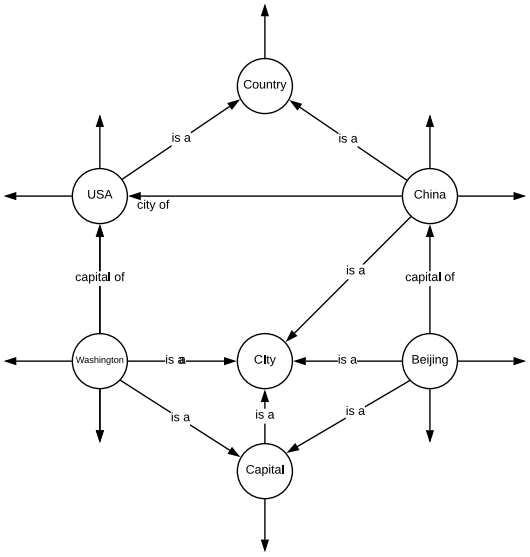

Fig. 1. An extract from an Ontology showing the vertices, edges and labels representing Country, City, capital and their relations

Definition 2.6. M is a $|A| \times |C|-$binary matrix where each entry-m_{ac} is defined as follows.

$$m_{ac} = \begin{cases} 1 & \text{If a and c are considered to be mapped} \\ 0 & \text{Otherwise} \end{cases}$$

where $a \in A$ is the table entity and $c \in C$ is the ontology concept.

Table 1
Sample Data of Countries, Population, capitals extracted from [7]

Rank	Country	Population	Capital
1	China	1,377,516,162	Beijing
2	India	1,291,999,508	New Delhi
3	USA	323,990,000	Washigton, D.C.
4	Indonesia	258,705,000	Jakarta
5	Brazil	206,162,929	Brasilia

Let's take an example to elaborate on above concepts. Table 1 (T1) shows a sample table which lists populations and capitals of five countries. Figure 1 (O1) represents an extract from an ontology that represents the vertices, edges and labels corresponding to

the concepts of Country, Population, Capital, and their relations. The task of semantic interpretation is to maximize the accurate mappings between the entities of table 1 against the ontology shown in figure 1. The set of table entities A and the set of ontology concepts C for this example are shown below.

$$\begin{aligned} A = \{&Table, Column_1, Column_2, \\ &Row_1, Row_2, Cell_{USA}, \\ &Cell_{China}, Cell_{Washington}, \\ &Cell_{Beijing} Header_{Country}, \\ &Header_{Capital}, ...\} \end{aligned} \tag{2}$$

$$\begin{aligned} C = \{&Country, Capital, China, \\ &USA, Beijing, Washington, \\ &City, ...\} \end{aligned} \tag{3}$$

Let a = Column$_2$ entity from T1 and c = Country concept from O1. Then according to the definition 2.6, m_{ac} = 1 since Column$_2$ is most related to the Country concept. Similarly we can determine a value for each entry in the mapping matrix. Finding the matrix M^* from the matrix space $\mathcal{M}$ that contains correct values for all the entries is goal of this process.

However in practical scenarios, table entities and ontology concepts are not mapped exhaustively. Mappings are bounded by the semantic model of the table. We observed two such models in literature:(1) table as a single class; and (2) table as a set of classes. These models reduce the matrix space $\mathcal{M}$.

Even though the matrix space is further reduced by the choice of the table model, due to the large size of the matrices, the problem still remains substantially complex. Thus algorithms divide the problem into several sub-problems by considering sub-matrices of the original matrices. By analyzing literature, we identified four such commonly used sub-problems. Each sub-problem clearly isolate the search process to a selected region of the initial matrix. In literature these sub-problems are referred to as Annotation Tasks. Annotation tasks can be carried out sequentially, iteratively, in parallel or in a hybrid manner. We identify them as Execution Approaches. The formal definitions and the implications of the different table models, annotation tasks and execution approaches are detailed later.

3. Table Modelling

A table can be modelled using ontologies in two ways. Those models are: (1) table as a single class; and (2) table as a set of classes.

Table as a single class Modelling table as a single class is a popular ideology in the algorithms focused on the task of interpreting web tables. T1 shows a table extracted from [7] consisting of countries ranked by their population. It can be considered as of the class "Country". Which in turn implies that the columns of the class corresponds to different properties of it while rows corresponds to instances. This model is used in algorithms [7, 10–12].

Several papers[7, 11–15] assume the presence of a label or a subject column in the tables that are classified as above. The subject column is considered as the naming column or the primary column of the table, which may or may not uniquely identify the instance of the table class represented by the respective row. In T1 the "Country" column is the subject column. Further, the rows of the table are considered as instances of the table class and columns of the table are mapped to properties. According to that, considering T1 Countries China, India and United States are instances of the "Country" class, while their properties are "Rank", "Country" (Name), "Population" and "Capital".

Table as a set of classes An alternate semantic model of tables corresponds each column of the table to a different class. This form of table model is presented in [1, 3, 6, 14]. The columns that consist of literal values (numbers, dates) rather than entities, are mapped to properties from the ontology. According to above ideology, in T1 columns "Country" and "Capital" correspond to respective classes in the ontology while "Rank" and "Population" correspond to respective properties. In contrast to *table as a single class* model, presence of inter-column relationships in tables is debatable in the *table as a set of classes* model. Some literature[6, 8, 14] strongly suggests the existence of such relationships.

In general web tables contain information regarding a single concept. Hence the table as a single class model could be considered as a better approach for modeling the table. We observed that trend rising in the recent literature[7, 8, 15]. However by analyzing prior literature, we could not arrive at a clear conclusion that supports either of the models in terms of efficiency and accuracy. We call out for a future work that

investigates the possible trade offs between these two choices.

4. Table Annotation tasks

An annotation task is a sub-problem of the initial STI process. It comprises of a selected set of table entities and ontology concepts. Searching for the best possible map for each table entity from the selected ontology concepts is the annotation process. It aims at identifying a candidate space for each table entity and narrowing it down until the best matching concept is found.

Candidate space selection A given table entity, tend to map with several concepts of the ontology when the ontology is considerably large. Candidate space selection aims to isolate these related concepts from the ontology to later process and single out the best mapping concept of them. Most algorithms[6–8, 11–14] in literature employ publicly available ontologies, prominently DBpedia[16], YAGO[17], Freebase[18], Google Knowledge Graph, and Wikitology[19]. The availability of a large number of properly curated concepts in these ontologies is the key incentive in using them. In such algorithms, the public API exposed by respective ontology is used to query the candidate space. Thus, the query process is limited to performing look-up or string matching for candidate search. This is a prevailing limitation in most of the existing methods. Several efforts have been presented to overcome this limitation including creating of local indices[3, 4, 7] and employing machine learning models[20–22]. While local indices address the querying limitations, the amount of concepts represented in these indices are considerably lower than that of aforementioned ontologies. Machine learning models on the other hand requires a substantial amount of manually annotated training data to function at a competent level. Hence, the candidate space selection process remains as a challenging aspect in STI.

To select the best map from the candidate space, score functions are defined in algorithms. The score provides an aggregated view of a set of similarity metrics between the source table entity and the candidate ontology concept. These metrics predominantly include lexical similarity[7, 14], semantic similarity[7, 11, 14] and popularity based ranking.

In literature, 4 major annotation tasks can be identified as follows: (1) cell annotation; (2) column annotation; (3) row annotation; and (4) table annotation. This section explains each task in detail.

4.1. Cell annotation

Table cells mostly contain entities(strings) and literals(numbers and dates). Out of which only entity cells can be mapped to instances of the ontology. This mapping process is the task of cell to instance annotation. Consider the previous example shown in T1 and O1. The cells of the columns "Country" and "Capital" of T1 are entity cells. Thus can be annotated with their corresponding instances from O1. But if we consider entity "China" from the column "Country" of T1, it can be annotated with both Q3(China, the country) and Q12(China, a city of California) of O1. To derive an accurate selection, more information about the context of the table is required. This context is obtained by the results of other annotation tasks.

Cell to instance annotation has been tackled in several ways in literature. Algorithms[7, 8, 10, 12, 14, 23] generally employ search based approaches such as look-up or regular expression search. Alternatively machine learning based methods have been presented in [22, 24, 25]. In literature, we observed that the cell to instance annotation has been given a lesser priority with respect to the rest of the annotation tasks. A plausible explanation being the relatively higher ambiguity associated with this task. Hence, cell annotation is only used as a preliminary task for further annotations.

4.2. Column annotation

Column annotation associates columns of tables with the corresponding concepts from an ontology. Based on the table model, it can be performed in two different ways. If the algorithm assumes the table as a set of classes model, it would independently map each table column to a different class in the ontology. Alternatively, algorithms that follow table as a single class model would map columns only to the properties of the selected table class. This essentially reduce the search space when compared to the former method.

Algorithms predominantly use column cell values to derive the mapping concepts for the columns. But some work[5, 11, 14] have utilized the column header in the process as well. Since, tables with missing headers and malformed headers affect the annotation results, algorithms often use headers only as an additional evidence. Columns in general can be categorized as Named Entity (NE) columns and literal columns.

NE columns contain entity strings as cell values while literal column contain dates or numbers. In most algorithms we observed, NE column annotation had been a key focus.

NE column annotation is carried out by first determining the cell annotations of the column and then using it as evidence for annotating the column. Since the cell annotations could suggest multiple possible candidates for the column concept, heuristics, scoring functions[24] or machine learning[20, 21] based approaches are used for determining the best candidate. Considering the above example of T1 and O1, for the table column "Country" there could be 2 different candidate concepts: (1) concept "City"; or (2) concept "Country". By considering the cell annotations, concept "City" has only 1 evidence while concept "Country" is suggested by all the entities. Hence, if we select the heuristic that the column concept is the candidate represented by a majority of the cell annotations, the suitable candidate concept is "Country".

In contrast to NE columns, literal columns tend to be given a lesser priority in the STI algorithms. In recent literature, few algorithms[3, 4, 8, 14, 22, 24] have emerged targeting to tackle the literal column annotation task. These work follow the table as a set of classes model. We observed that pre-processing the target ontology to create a hierarchical clustering for numeric values as the methodology followed in most of these algorithms[3, 4]. If we consider the columns "Rank" and "Population" of T1 above, they clearly show contrasting distribution characteristics. According to the claims of [3, 4] these two columns can be accurately clustered with proper pre-processing. Although these algorithms have shown promising results, utilizing the table header in the process could have improved the overall process.

4.3. Row annotation

A row comprises a set of table cells which contain interrelated content. Being properties of a particular instance from the ontology is the most frequently occurring interrelation. Thus, the row can be considered as a representation of the aforementioned instance. Row annotation can be initiated either by annotating the individual NE cells of the row or by identifying the entity label of the row. Individually annotated cells can be collectively utilized to infer the most appropriate instance represented by a row. On the other hand, if the subject column of the table can be identified, the corresponding cell of the row can be taken as the entity label of it. Subsequently by mapping the entity label to the ontology, the instance represented by the row can be identified.

Row annotation is often carried out by the algorithms that follow table as a class model.[7, 10] Annotated rows are taken as evidence for inferring the table class and in return the inferred table class is used to improve the row annotations. Hence, the 2 tasks are often observed to be executing iteratively. However Ritze et al.[11] has effectively shown ways of utilizing row annotations even while following the table as a set of classes model.

4.4. Table annotation

Modelling table as a single class requires the identification of a class which can provide the best representation for a given table. Identification of this class in such scenarios is the main goal of this task.

Table as a single class model in general assumes a presence of a subject column. This task first tries to find the subject column and then utilizes this column to determine the class corresponding to the table. Most algorithms[7, 11–15] rely on heuristics to identify the subject column. The heuristic presented in [7, 10, 11] identifies the subject column as the left most column of the table with maximum number of distinct values. Further Venetis et al.[1] have considered the possibilities of multiple subject columns and the absence of a subject column. Moreover [1] have presented a supervised learning based approach for determining the subject column in contrast to the heuristics presented in contemporary work.

An alternate approach for identifying the table class is presented by Ritze et al.[11]. The process first peforms row annotation as described in the preceding section. Then the class that represents a majority of these instances is considered as the table class.

5. Execution Approaches

The annotation tasks outlined in section 4 are the unit level processes used in the mapping mechanism. The complete mechanism usually consist of one or more of the above tasks processed sequentially, parallely, iteratively; or in a hybrid manner.

Sequential Approach is the most commonly used approach in STI algorithms[13, 24]. Often the output of the preceding annotation task is used as evidence for

the latter task. Utilizing the output of cell annotation in column annotation[13] and the output of row annotation for table annotation are two such instances.

Parallel Approach executes annotation tasks without sharing the outputs of each other. But the resulting outputs are evaluated together to determine inputs that would collectively improve the accuracy of both tasks. Work by Limaye et al.[6] introduces such methodology where 3 of the above annotation tasks are executed parallel by mapping them onto a probabilistic graphical model. Then the search is iterated until a solution that maximizes the overall probability of the system is achieved. This algorithm has been further improved by Mulwad in TABEL[14].

Iterative Approach takes a set of annotation tasks and executes them repeatedly executed until all tasks produce the optimum results. Each iteration takes an input from previous iteration to improve the performance. For an instance, cell annotations of a column can be utilized for annotating the column and the resulting column annotation can be taken as evidence to improve the annotations of its cells. Algorithms provided in both TableMiner methods [2, 8] are based on this principle of iterative improvement. Furthermore, these algorithms claim that iterative approaches can perform more efficiently in comparison to sequential approaches by using partial annotations of annotation tasks to improvement each other.

Hybrid Approach Even-though the execution approaches of annotation tasks can be conceptually categorized as such, in practice algorithms comprise of a mixture or an hybrid of two or more of the aforementioned approaches[7]. We observed that a common pattern exists where either a sequential or parallel execution of tasks is combined with iteration to formalize the overall system.

6. Classification of algorithms presented in recent literature

This section outlines several algorithms from the recent literature. Considering the implementation of these algorithms, we classified them into 3 distinct classes: (1) Search based algorithms; (2) Supervised Learning based algorithms; and (3) Unsupervised Learning based algorithms. Below we discuss the features, strengths and limitations of each of these three kinds of algorithms in detail. Further, we review key tools and frameworks in each category.

6.1. Search based algorithms

A majority of recently presented algorithms utilizes a keyword search based approach in the implementation. Mainly, these algorithms search a particular ontology by using its query API[2, 7, 8, 10, 11, 14, 15, 26, 27] or by manually creating a search index[5, 7, 12, 13, 28, 29] and using it. A bulk of the ontologies used in literature[16, 17, 19] provide a public API to query the concepts of the ontology. Capabilities provided by these interfaces are in general limited to exact match and regular expression queries. Yet, for the purpose of obtaining the linked concepts of a given entity (parent class, properties, relations, instances), the functioning of the interfaces are quite adequate.

TableMiner[2] and its improvement **TableMiner+** [8] utilizes the public query APIs of DBpedia and Freebase for its candidate selection process. Key feature of both algorithms is the bootstrapping annotation approach used for efficiently annotating the columns and the cells. The presented approach uses the results of partial annotation of cells along a column to derive the column class. Next the annotated class is considered as evidence for improving the previous annotations of the cells and to infer further annotations. The overall process is iterated until the cell and column annotation properly validate each-other. The author claims that the iterative annotation approach is much more efficient than the exhaustive annotation methods presented in contemporary works. Following the table as a single class model, TableMiner presents efficient approach for carrying out cell annotation, NE column annotation, and literal column annotation tasks. Additionally TableMiner+ supports subject column detection and relation extraction.

TABEL[14] is a domain independent, extensible framework for inferring semantics of tables. It was built by extending the early ideas of Mulward et al.[30] and Limaye et al.[6]. Author identifies the algorithm as a framework justifying by the modular design and the inherent extensibility of it. It comprises of a workflow of 5 phases: (1) Pre-processing phase; (2) Query and Rank phase; (3) Joint inference; (4) RDF Linked data; and (5) Human in the loop. Pre-processing phase focuses on tackling pragmatic challenges present in the table data. In the Query and Rank phase, TABEL utilizes the API provided by Wikitology[19] for candidate search process and employs several similarity metrics for ranking the candidates. In the joint inference phase, the framework carry out the annotation tasks by using the probabilistic graphical model based

algorithm proposed by Limaye et al.[6]. Final 2 phases provide added features by utilizing the annotated results. TABEL follows the table as a set of classes model and provides annotations for cells, NE columns as well as literal columns. Moreover it provides binary relation extraction, RDF generation and ability to take human input as added features.

T2K Match[15] is designed to efficiently annotate large quantities of tables that are relatively smaller in size. While it depends on the DBpedia query interface for the candidate selection task, it remarkably stands out from the former two algorithms by successfully covering all 4 annotation tasks of STI. It follows the table as a single class model and initiates the process with subject column detection. By using the subject column, it first narrows down the candidate space search. Next through an iterative execution approach, the cells, rows and NE columns and the table are annotated. In comparison to the former algorithms T2K Match covers all STI annotation tasks and additionally provides subject column detection. On the downside it lacks literal column annotation and relation extraction. Yet in considering the key STI problem, T2K Match can be considered as one of the most comprehensive algorithms.

TableFinder[13] utilizes an "isA database" created by mining the web. This database consist of value pairs that signify instance - class relationships. For example, if we consider instances in O1, pairs would include (USA, Country), (China, City), and (China, Country). Further a ranking mechanism is defined for the pairs within the database. The algorithm initially attempts to identify the subject column. Unlike former algorithms where subject column detection was based on heuristics, TableFinder uses several table features to train a SVM[31] model and detect the subject column. It proceeds by annotating cells and NE columns by executing in a sequential approach.

Wang et al.[5] follow a similar approach to the TableFinder by utilizing a custom index, Probase[32]. In contrast to the isA database used by TableFinder, Probase contains Hearts patterns[33] which shows "such as" patterns between classes and instances. The algorithm utilizes the API methods provided by Probase for querying the instances and provides annotations for cells and NE columns. Moreover it is capable of identifying the subject column.

FactBase Look-up[7] is a look-up based algorithm introduced by Efthymiou et al. The approach consumes a manualy index "FactBase", a generic search index over Wikidata[34] entries. Following the table as a class model, it initiates the process with subject column detection followed by annotation of subject column cells. For the cell annotation a refined look-up method is presented. By using the previous annotation results, rows and NE columns are annotated. The execution of the algorithm can be modeled as hybrid approach where sequential and iterative approaches are combined. The authors claim that the FactBase Look-up algorithm performs well in comparison to similar look-up and search based algorithms.

By observing the search based STI algorithms, several conclusion can be drawn; (1) algorithms that utilize API query based search approach predominantly follow the table as a single class model [2, 15] while the custom index based algorithms are keen to follow the table as a set of classes model.[5, 13]; (2) cell and column annotations are carried out in all of them; (3) a majority of them has the capability to detect the subject column [2, 5, 7, 13, 15].

6.2. Supervised Learning based algorithms

In recent literature, a tendency for applying supervised learning for STI process can be seen. In general such algorithms[20–22, 25, 35, 36] target on obtaining a high accuracy for a particular annotation task rather than the overall STI process. We observed NE column annotation as the most prominent target in a majority of the algorithms. The process is introduced as "Semantic Labeling"[20–22] in some of the above literature, which in-fact can be classified as a subtask of the STI process.

DSL[20] (Domain-independent Semantic Labeler) is a supervised learning based algorithm focused on accurately annotating columns. The algorithm first selects the candidate set of classes from the ontology and training data corresponding to these classes. Training data is a table corpus with the columns already annotated. Next these annotations along with the column cell values are processed to obtain feature vectors. Processing involves calculating 5 similarity metrics; (1) attribute name similarity; (2) standard Jaccard similarity and a modified similarity for numeric data; (3) TF-IDF cosine similarity; (4) distribution similarity; and (5) histogram similarity. A binary classifier is trained per column class in the training data set. Authors claim that the resulting model works relatively better than the search based approaches as well as similar supervised learning approaches.

DINT[21] (Data INTegrator) follows a similar supervised approach as of DSL. In contrast to DSL, 26

hand engineered features are used to train a Random Forrest Classifier using the training data corpus. DINT points out the issue of lack of annotated table columns for supervised learning and propose bagging of the existing training data as a solution. Further, DSL is capable of handling columns that cannot be accurately classified by the model by training for a "unknown" class.

Karma[35] is a semi-automatic system, which takes the source tables and the ontology as inputs and generate a semantic model. This model incorporates the user input to fine tune the annotations and use CRF for learning. Generated semantic model is then used for further annotation of the columns. While the algorithm show promising results, limitation of the method such as unknown column classes and lack of training data are not discussed by the authors.

TabEl[25] considers the STI problem as an entity linking task where each cell in the table corresponding is a string that need to be linked to a entity(concept) in the ontology. Hence the algorithm focuses only on the cell annotation task of STI process. It consists of 3 steps: (1) mention identification; (2) entity candidate generation; and (3) disambiguation. The candidate space selection process is done by calculating the prior probability for a given cell and a concept. The probability estimate is generated using the Web hyperlinks. In steps (1) and (2), the algorithm finds the most related candidate per cell using thus calculated probabilities. For disambiguation, TabEl employs a pre-trained local classifier that ranks the candidates by the maximum likelihood. The algorithm is executed in an iterative manner until no further changes occur in the disambiguation process. In contrast to previous mentioned supervised learning algorithms, TabEl is focused on cell annotation than on NE column annotation.

Supervised learning based approaches have shown competitive accuracy when compared to the contemporary search based approaches. Yet, the capabilities of a majority of the systems are limited to cell annotation[25] and column annotation[20–22, 25, 35]. Further, constructing training data for the supervised learning approaches is a tedious task and often involves a substantial amount of manual work. This issue has been highlighted by Ruemmele et al.[21]. Considering these limitations, it is apparent that supervised learning algorithms lags in the overall STI process when compared to search based algorithms.

6.3. Unsupervised Learning based algorithms

Unsupervised learning based approaches have been used only in a few algorithms[3, 4] through literature. Interestingly these algorithms are solely focused on the task of accurately annotating literal columns. In general clustering based approaches along with ontology prepossessing have been employed for achieving the task.

Neumaier et al.[4] introduces a hierarchical clustering based approach specifically focusing the task of numeric column annotation. The algorithm initially obtains classes and corresponding numerical properties and value sets from DBpedia. By using the interrelations among the obtained classes and considering the property types, a hierarchical cluster is generated. In the annotation process, the numerical column values are sampled and compared against the generated clusters using k-nearest neighbour search. Authors show that this approach work exceptionally well for numeric columns.

Nguyen et al.[3] proposes a similar system to that of Neumaier et al.[4], employing Wikidata[34] as the ontology. In contrast to the previous work, the algorithm is capable of annotating literal columns accurately regardless of the unit of measurement used in the source table. This is a critical improvement since previous work rely heavily on the assumption that the table data would employ the units of measurements as of the reference ontology. Thus it achieves better accuracy in annotating than the predecessors.

Although the aforementioned algorithms considerably improve the annotation accuracy for literal columns, support for other annotation tasks would have greatly increased the impact of the algorithm. Moreover, utilizing the table header annotation approach within along with the column annotator could have increased their overall accuracy. Considering the above factors, unsupervised learning based approaches can not be considered as comprehensive algorithms for the STI process.

Table 2 summarizes the algorithms presented above, focusing on the key analysis points based on the conceptual basis introduced in this paper. Further, it can be used as a guide for initiating a novel STI algorithm, with desired capabilities and features. By observing the table, it is apparent that search based approaches more concise in overall capabilities in comparison with the alternate approaches.

Table 2
Summary of the features and capabilities of notable STI algorithms outlines in Section 6. C - Table as a single class model, N - Table as a set of classes model

Algorithm		Table Model	Cell Annotation	Column Annotation	Table Annotation	Row Annotation	Subject Column Identification	Relation Annotation	Execution	Ontology
Search	TableMiner+[2]	C	✓	✓	✓		✓		Iterative	DBpedia
	TABEL[14]	N	✓	✓				✓	Hybrid	Wikitology
	T2K Match[15]	C	✓	✓	✓	✓	✓		Iterative	DBpedia
	TableFinder[13]	N	✓	✓			✓		Sequential	Custom
	Wang et al.[5]	N	✓	✓			✓			Probase
	FactBase Lookup[7]	C	✓	✓	✓	✓	✓		Hybrid	Wikidata
Supervised Learning	DSL[20]	N		✓						Generic
	DINT[21]	N		✓						Generic
	Karma[35]	N		✓						Custom
	TabEl[25]	N	✓						Iterative	YAGO
Unsupervised Learning	Neumaier et al.[4]	N		✓						DBpedia
	Nguyen et al.[3]	N		✓						DBpedia

7. Evaluation

In this section we review the evaluation methodology adapted in STI algorithms in general and the gold standards used. Further we present the evaluation of 3 selected algorithms from recent literature in a concise manner. Finally we provide insights into ways of improving the evaluation methodology in future.

7.1. Evaluation mechanism

The artifacts of the STI process consist of table, its cells, columns and rows annotated with corresponding concepts from the ontology. In general, the output is the URI of the ontology concept and the confidence score. Based on the focus on the algorithm there could be minor variations, yet the format agrees across algorithms in an abstract level. Although there are numerous STI algorithms presented in literature, they have not yet agreed upon a common gold standard for measuring the performance. Hence, in this paper we outline 3 popular gold standards observed in the recent literature.

7.2. Gold standards

Gold standards for evaluating STI algorithms consist of 3 components; (1) the table corpus; (2) annotations; and (2) the ontology. The table corpus is a set of tables extracted from the web. Tables comprise of varying properties of cells, rows, columns and characteristics. The ontology in general is a freely available web ontology, primarily DBPedia[16], YAGO[17], Freebase[18], and Wikitology[19]. The ontology is mentioned but not bundled with the gold standards. We observed 3 gold standards that are commonly used across literature and outline them in the proceeding sections.

T2D[37] is one of the largest and comprehensive gold standards. It is based on the table as a single class model. The table corpus consist of tables extracted from the English language subset of the Web Data Commons Web Table Corpus[38] and DBpedia is used as the reference ontology. This gold standard provides annotations for tables, columns and rows. Recently in an attempt to improve the quality of T2D, an improved version, T2D*[23] was presented. Ermilov et al. claims that T2D* addresses several shortcomings of T2D by manually curating the table corpus and the annotations in it. However T2D* only provides annotations for tables and columns.

DBD[23] (DBpedia Table Data set)[1] consist of a table corpus generated from DBpedia concise bounded

[1] https://github.com/aksw/TAIPAN-DBD-Datagen

descriptions[2]. Similar to T2D, it follows the table as a single class model. Apart from table and row annotation is also provide ground truth for subject column identification. Although the quality of the data and annotations is high in DBD, the noise inherent in web tables is missing.

Limaye et al.[6] presented a manually compiled gold standard which maps Wikipedia tables to Freebase concepts. This was later used as a gold standard in several works[2, 7]. It consists of 4 parts, and consists of cell and table annotations. Later [7] improved the data set by converting the tables to CSV and manually providing annotations using DBpedia than Freebase[3]. Considering the retirement of Freebase, this made the gold standard available for testing using DBpedia.

Apart from the gold standards outlined above, several other data sets have been used across literature for evaluation purposes. But different algorithms have not yet agreed upon a common standard for table corpus, annotation output or the ontology. Initial gold standards presented tables as HTML[6] or XML[6]. As a result STI algorithms were required to implement a parser component for reading the data set. Latter gold standards introduced the table corpus in CSV[23, 37] which had comparatively less overhead on the algorithm for parsing the data. Efthymiou et al.[7] contributed to the process by converting both T2D and Limaye data sets to JSON format.

7.3. Evaluation Methodology

In literature STI algorithms are predominantly evaluated using the confidence of the annotation. For which Precision (P), Recall (R) and F1 metrics are used. These metrics are defined[11] as follows,

$$P = \frac{TP}{TP + FP} \tag{4}$$

$$R = \frac{TP}{TP + FN} \tag{5}$$

$$F1 = \frac{2PR}{P + R} \tag{6}$$

[2]https://www.w3.org/Submission/CBD/

[3]http://www.cs.toronto.edu/ oktie/webtables

Where TP, FP and FN are the number of true positives, false positive and and false negatives respectively. Since the annotations are generated table wise and a given data set contains multiple tables, the final P, R and F1 scores are given by the micro averages of the individual metrics of tables. For instance if the data set contains N tables and the corresponding annotation results are (TP_1, FP_1,FN_1),...,(TP_N, FP_N, FN_N) respectively, then the micro averaged TP, FP and FN are defined as,

$$TP = \frac{TP_1 + ... + TP_N}{N} \tag{7}$$

$$FP = \frac{FP_1 + ... + FP_N}{N} \tag{8}$$

$$FN = \frac{FN_1 + ... + FN_N}{N} \tag{9}$$

In Table 3 we present the evaluation results of 3 search based algorithms against T2D and Limaye gold standards outlined above. The algorithms T2K Match[15], TableMiner+[8] and FactBase Lookup[7] are chosen based on their overall capabilities and performance. Moreover these algorithms are considered as state-of-the-art in the field of STI[39]. The results are extracted from literature[7, 39] and aggregated to provide a comprehensive understanding. Precision and Recall values for TableMiner+ was not found in literature and hence left empty, but the overall F1 measure provides comparable insights.

By observing the evaluation results against T2D gold standard, it is evident that all 3 algorithms perform well with F1 scores above 0.80. This can be explained by the high level of structuredness of the data set. Limaye data set on the other hand contains tables extracted from Wikipedia. As a result a majority of the tables are small and sparse, posing a disadvantage on the algorithms that rely on sampling rows for annotations. This could be considered as the reason for the low results of T2K Match in comparison other 2 algorithms. Hence we can consider TableMiner+ and FactBase Lookup as competitive algorithms for STI. Yet we can not conclude them as the overall best since the results are highly subjective of the data set evaluated upon.

Table 3
Evaluation of T2K Match[15], TableMiner+[8] and FactBase Lookup[7] algorithms against T2D and Limaye data sets

Algorithm	T2D Gold Standard			Limaye Gold Standard		
	P	R	F1	P	R	F1
TableMiner+	-	-	0.81	-	-	0.84
T2K Match	0.90	0.76	0.82	0.70	0.63	0.66
Factbase Lookup	0.88	0.78	0.83	0.84	0.78	0.81

In literature providing the micro averaged values of all annotations has been the preferred way. Yet we believe that such results do not provide an accurate insight on the individual annotation performances. As we observed in table 3 the performance of overall algorithm is subjective to the data set being evaluated on. Hence evaluating each capability and feature and capability individually will be a better approach for evaluating STI algorithms. Based on the conceptual framework introduced in this paper, we propose the following criteria for STI algorithm evaluations.

- Precision, Recall and F1 for cell annotation
- Precision, Recall and F1 for NE column annotation
- Precision, Recall and F1 for literal column annotation
- Precision, Recall and F1 for row annotation
- Precision, Recall and F1 for table annotation
- Percentage of identification of subject column

By following the aforementioned criteria, different algorithms can be easily analyzed capability wise without relying on the overall performance of the algorithm. We believe this would be of immense use for developing novel approaches for STI by improving the lacking areas rather than reinventing the overall algorithms.

8. Conclusion and Future Work

This paper introduces a conceptual basis for analyzing STI algorithms, presents a categorical view on recent STI algorithms and analyzes the categories to discuss the features and limitations of them. The analysis concludes that search based approaches outperform the other categories in the basis of accuracy and overall completeness. Further observations concludes that the supervised learning approaches perform well in annotating NE columns to relatively small, domain specific ontologies and that the unsupervised learning based approaches perform well for annotating literal columns. This paper identifies several challenging aspects prevailing in STI. Firstly candidate space selection with high precision and recall was seen to be an unresolved challenge across algorithms. Secondly subject column detection being carried out majorly using heuristics could impose unexpected bias to the algorithms. Lastly the numerous limitations that exist in the current evaluation methodology are listed in section 7.

In section 7 we outlined possible improvements for the evaluation methodology of algorithms. Yet the gold standards used at present is not equipped to evaluate all listed criteria. Hence we propose the future directions to be invested in introducing a gold standard that could facilitate evaluation of algorithms based on the whole proposed criteria. Moreover, we recommend to generalize the format for all future data sets and annotation results as an initiation of standardizing STI gold standards. We believe that these actions would lead to progressive development of STI algorithms in future.

References

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

[2] Z. Zhang, Towards efficient and effective semantic table interpretation, in: *International Semantic Web Conference*, Springer, 2014, pp. 487–502.

[3] P. Nguyen and H. Takeda, Semantic labeling for quantitative data using Wikidata.

[4] S. Neumaier, J. Umbrich, J.X. Parreira and A. Polleres, Multi-level semantic labelling of numerical values, in: *International Semantic Web Conference*, Springer, 2016, pp. 428–445.

[5] J. Wang, H. Wang, Z. Wang and K.Q. Zhu, Understanding tables on the web, in: *International Conference on Conceptual Modeling*, Springer, 2012, pp. 141–155.

[6] G. Limaye, S. Sarawagi and S. Chakrabarti, Annotating and searching web tables using entities, types and relationships, *Proceedings of the VLDB Endowment* **3**(1–2) (2010), 1338–1347.

[7] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro and V. Christophides, Matching web tables with knowledge base entities: From entity lookups to entity embeddings, in: *International Semantic Web Conference*, Springer, 2017, pp. 260–277.

[8] Z. Zhang, Effective and efficient semantic table interpretation using tableminer+, *Semantic Web* **8**(6) (2017), 921–957.
[9] U. Thayasivam and P. Doshi, Improved Convergence of Iterative Ontology Alignment using Block-Coordinate Descent., in: *AAAI*, 2012.
[10] D. Ritze, Web-Scale Web Table to Knowledge Base Matching, PhD thesis, 2017.
[11] D. Ritze and C. Bizer, Matching web tables to DBpedia-a feature utility study, *context* **42**(41) (2017), 19.
[12] X. Zhang, Y. Chen, J. Chen, X. Du and L. Zou, Mapping entity-attribute web tables to web-scale knowledge bases, in: *International Conference on Database Systems for Advanced Applications*, Springer, 2013, pp. 108–122.
[13] P. Venetis, A. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao and C. Wu, Table Search Using Recovered Semantics (2010).
[14] V.V. Mulwad, *Tabel–a domain independent and extensible framework for inferring the semantics of tables*, University of Maryland, Baltimore County, 2015.
[15] D. Ritze, O. Lehmberg and C. Bizer, Matching HTML tables to DBpedia, in: *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, ACM, 2015, p. 10.
[16] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z. Ives, Dbpedia: A nucleus for a web of open data, in: *The semantic web*, Springer, 2007, pp. 722–735.
[17] F.M. Suchanek, G. Kasneci and G. Weikum, Yago: a core of semantic knowledge, in: *Proceedings of the 16th international conference on World Wide Web*, ACM, 2007, pp. 697–706.
[18] K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, AcM, 2008, pp. 1247–1250.
[19] Z.S. Syed, T. Finin and A. Joshi, Wikitology: Using wikipedia as an ontology, in: *proceeding of the second international conference on Weblogs and Social Media*, 2008.
[20] M. Pham, S. Alse, C.A. Knoblock and P. Szekely, Semantic labeling: a domain-independent approach, in: *International Semantic Web Conference*, Springer, 2016, pp. 446–462.
[21] N. Ruemmele, Y. Tyshetskiy and A. Collins, Evaluating approaches for supervised semantic labeling, *arXiv preprint arXiv:1801.09788* (2018).
[22] S.K. Ramnandan, A. Mittal, C.A. Knoblock and P. Szekely, Assigning semantic labels to data sources, in: *European Semantic Web Conference*, Springer, 2015, pp. 403–417.
[23] I. Ermilov and A.-C.N. Ngomo, TAIPAN: Automatic Property Mapping for Tabular Data, in: *European Knowledge Acquisition Workshop*, Springer, 2016, pp. 163–179.
[24] G. Hignette, P. Buche, J. Dibie-Barthélemy and O. Haemmerlé, Fuzzy annotation of web data tables driven by a domain ontology, in: *European Semantic Web Conference*, Springer, 2009, pp. 638–653.
[25] C.S. Bhagavatula, T. Noraset and D. Downey, TabEL: entity linking in web tables, in: *International Semantic Web Conference*, Springer, 2015, pp. 425–441.
[26] T. Knap, Towards Odalic, a Semantic Table Interpretation Tool in the ADEQUATe ProjectâĂŤ, in: *Proceedings of the 5th International Workshop on Linked Data for Information Extraction co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria*, 2017, pp. 26–37.
[27] E. Munoz, A. Hogan and A. Mileo, Triplifying Wikipedia's Tables., *LD4IE@ ISWC* **1057** (2013).
[28] Z. Syed, T. Finin, V. Mulwad and A. Joshi, Exploiting a web of semantic data for interpreting tables, in: *Proceedings of the Second Web Science Conference*, Vol. 5, 2010.
[29] H. Bian, Y. Chen, X. Du and X. Zhang, MetKB: enriching RDF knowledge bases with web entity-attribute tables, in: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, ACM, 2013, pp. 2461–2464.
[30] V. Mulwad, T. Finin, Z. Syed and A. Joshi, Using Linked Data to Interpret Tables., *COLD* **665** (2010).
[31] B.E. Boser, I.M. Guyon and V.N. Vapnik, A Training Algorithm for Optimal Margin Classifiers, in: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, ACM, New York, NY, USA, 1992, pp. 144–152. ISBN 0-89791-497-X. doi:10.1145/130385.130401. http://doi.acm.org/10.1145/130385.130401.
[32] W. Wu, H. Li, H. Wang and K.Q. Zhu, Probase: A probabilistic taxonomy for text understanding, in: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ACM, 2012, pp. 481–492.
[33] M.A. Hearst, Automatic acquisition of hyponyms from large text corpora, in: *Proceedings of the 14th conference on Computational linguistics-Volume 2*, Association for Computational Linguistics, 1992, pp. 539–545.
[34] D. Vrandečić and M. Krötzsch, Wikidata: a free collaborative knowledgebase, *Communications of the ACM* **57**(10) (2014), 78–85.
[35] C.A. Knoblock, P. Szekely, J.L. Ambite, A. Goel, S. Gupta, K. Lerman, M. Muslea, M. Taheriyan and P. Mallick, Semi-automatically mapping structured sources into the semantic web, in: *Extended Semantic Web Conference*, Springer, 2012, pp. 375–390.
[36] M. Taheriyan, C.A. Knoblock, P. Szekely and J.L. Ambite, Learning the semantics of structured data sources, *Web Semantics: Science, Services and Agents on the World Wide Web* **37** (2016), 152–169.
[37] D. Ritze, O. Lehmberg and C. Bizer, T2D Gold Standard for Matching Web Tables to DBpedia. http://webdatacommons.org/webtables/goldstandard.html#toc6.
[38] O. Lehmberg, D. Ritze, R. Meusel and C. Bizer, A large public corpus of web tables containing time and context metadata, in: *Proceedings of the 25th International Conference Companion on World Wide Web*, International World Wide Web Conferences Steering Committee, 2016, pp. 75–76.
[39] B. Kruit, P. Boncz and J. Urbani, Extracting New Knowledge from Web Tables: Novelty or Confidence? (2018).
[40] C. Chen and Y. Zhao, Research on Column Concept Vector Based Web Table Matching, in: *Web Information System and Application Conference (WISA), 2015 12th*, IEEE, 2015, pp. 165–168.
[41] T. Berners-Lee, J. Hendler and O. Lassila, The semantic web, *Scientific american* **284**(5) (2001), 28–37.
[42] D. Rinser, D. Lange and F. Naumann, Cross-lingual entity matching and infobox alignment in Wikipedia, *Information Systems* **38**(6) (2013), 887–907.
[43] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*, MIT press, 2009.
[44] T. Mikolov, K. Chen, G. Corrado, J. Dean, L. Sutskever and G. Zweig, word2vec, *Google Scholar* (2014).

[45] S. Zwicklbauer, C. Seifert and M. Granitzer, DoSeR-a knowledge-base-agnostic framework for entity disambiguation using semantic embeddings, in: *International Semantic Web Conference*, Springer, 2016, pp. 182–198.

[46] G.A. Miller, WordNet: a lexical database for English, *Communications of the ACM* **38**(11) (1995), 39–41.