A Domain Independent Framework for Extracting Linked Semantic Data from Tables

Varish Mulwad, Tim Finin and Anupam Joshi

Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD 21250 USA
{varish1,finin,joshi}@cs.umbc.edu

Abstract. Vast amounts of information is encoded in tables found in documents, on the Web, and in spreadsheets or databases. Integrating or searching over this information benefits from understanding its intended meaning and making it explicit in a semantic representation language like RDF. Most current approaches to generating Semantic Web representations from tables requires human input to create schemas and often results in graphs that do not follow best practices for linked data. Evidence for a table's meaning can be found in its column headers, cell values, implicit relations between columns, caption and surrounding text but also requires general and domain-specific background knowledge. Approaches that work well for one domain, may not necessarily work well for others. We describe a domain independent framework for interpreting the intended meaning of tables and representing it as Linked Data. At the core of the framework are techniques grounded in graphical models and probabilistic reasoning to infer meaning associated with a table. Using background knowledge from resources in the Linked Open Data cloud, we jointly infer the semantics of column headers, table cell values (e.g., strings and numbers) and relations between columns and represent the inferred meaning as graph of RDF triples. A table's meaning is thus captured by mapping columns to classes in an appropriate ontology, linking cell values to literal constants, implied measurements, or entities in the linked data cloud (existing or new) and discovering or and identifying relations between columns.

Keywords: linked data, RDF, Semantic Web, tables, entity linking, machine learning, graphical models

1 Introduction

The Web has become a primary source of knowledge and information, largely replacing encyclopedias and reference books. Most Web text is written in a narrative form as news stories, blogs, reports, letters, etc., but significant amounts of information is also encoded in structured forms as stand-alone spreadsheets or tables and as tables embedded in Web pages and documents. Cafarella et al.

[5] estimated that the Web contains over 150 million high quality relational html tables.

Tables are also used to present and summarize key data and results in documents in many subject areas, including science, medicine, healthcare, finance, and public policy. As a part of a coordinated open data and transparency initiative, nearly 30 nations are publishing government data on sites in structured formats. The US data.gov site shares more than 390,000 datasets drawn from many federal agencies and is complemented by similar sites from state and local government organizations. Tables are used to represent significant amount of information and knowledge, yet, we are not able to fully exploit it. Both integrating or searching over this information will benefit from a better understanding of the intended meaning of the data and its mapping to other reference dataset.

The goal of our research is to unlock knowledge encoded in tables. In this paper, we present a domain independent framework for automatically inferring the intended meaning and semantics associated with tables. Using the Linked Open Data [2] (or an provided ontology, knowledge base [KB]) as background knowledge, our techniques grounded in graphical models and probabilistic reasoning, map every column header to a class from an ontology, links table cell values to entities from the KB and discovers relations between table columns. The inferred information is represented as a graph of RDF triples allowing other applications to utilize the recovered knowledge.

2 Impact

Many real world problems and applications can benefit from exploiting information stored in tables including evidence based medical research [22]. Its goal is to judge the efficacy of drug dosages and treatments by performing meta-analyses (i.e systematic reviews) over published literature and clinical trials. The process involves finding appropriate studies, extracting useful data from them and performing statistical analysis over the data to produce a evidence report.

Key information required to produce evidence reports include data such as patient demographics, drug dosage information, different types of drugs used, brands of the drugs used, number of patients cured with a particular dosage etc. Most of this information is encoded in tables, which are currently beyond the scope of regular text processing systems and search engines. This makes the process manual and cumbersome for medical researchers.

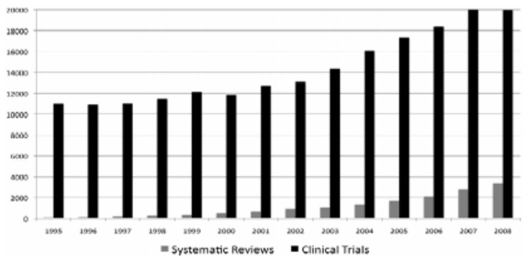


Fig. 1. The number of papers reporting on systematic reviews and meta-analyses is small compared to those reporting on individual clinical trials, as shown in this data from MEDLINE.

Presently medical researchers perform keyword based search on systems such as PubMed's MEDLINE which end up producing many irrelevant studies, requiring researchers to manually evaluate all of the studies to select the relevant ones. Figure 1 obtained from [6] clearly shows the huge difference in number of meta-analysis and number of clinical trials published every year. By adding semantics to tables like Figure 2, we can develop systems that can easily correlate, integrate and search over different tables from different studies to be combined for a single meta-analysis.

Web search is another area that can benefit from understanding information stores in tables. Search engines work well at searching over text in web pages, but poorly when searching over tables. If recovered semantics are available, search engines can answer queries like *dog breeds life span*, *wheat production in Africa* or *temperature change in the Arctic*,with tables or web pages containing them as results. We also see our work helping to generate high quality semantic linked data, which in turn will aid the growth of the Semantic Web.

3 Inferring the Semantics of Tables

Analyzing tables provide unique challenges. One might be tempted to think that regular text processing might work with tables as well. After all tables also store text. However that is not the case. To differentiate between text processing and table processing consider the text "Barack Hussein Obama II (born August 4, 1961) is the 44th and current President of the United States. He is the first African American to hold the office."

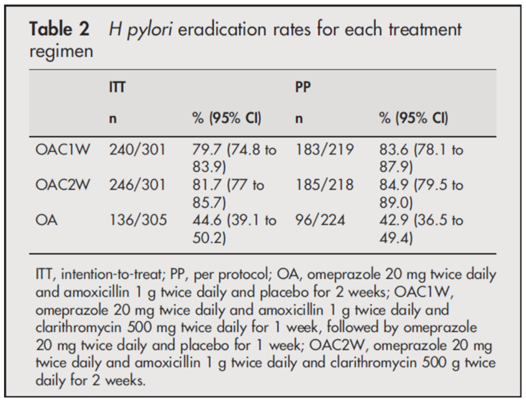


Table 2 *H pylori* eradication rates for each treatment regimen

	ITT		PP	
	n	% (95% CI)	n	% (95% CI)
OAC1W	240/301	79.7 (74.8 to 83.9)	183/219	83.6 (78.1 to 87.9)
OAC2W	246/301	81.7 (77 to 85.7)	185/218	84.9 (79.5 to 89.0)
OA	136/305	44.6 (39.1 to 50.2)	96/224	42.9 (36.5 to 49.4)

ITT, intention-to-treat; PP, per protocol; OA, omeprazole 20 mg twice daily and amoxicillin 1 g twice daily and placebo for 2 weeks; OAC1W, omeprazole 20 mg twice daily and amoxicillin 1 g twice daily and clarithromycin 500 mg twice daily for 1 week, followed by omeprazole 20 mg twice daily and placebo for 1 week; OAC2W, omeprazole 20 mg twice daily and amoxicillin 1 g twice daily and clarithromycin 500 g twice daily for 2 weeks.

Fig. 2. Tables in clinical trials literature have characteristics that differ from typical, generic Web tables. They often have row headers well as column headers, most of the cell values are numeric, cell values are often structured and captions can contain detailed metadata. (From [32])

The over-all meaning can be understood from the meaning of words in the sentence. The meaning of each word can be can be recovered from the word itself or by using context of the surrounding words. Now consider the table shown in Figure 2. In some ways, this information is easier to understand because of its structure, but in others it is more difficult because it lacks the normal organization and context of narrative text. The message conveyed by table in Figure 2 is different eradication rates for different drug dosages and treatment regimes for the disease *H pylori*. Similarly consider the table shown in Figure 3. The table represents information about cities in the United States of America. A closer

City	*State*	*Mayor*	*Population*
Baltimore	MD	S.C.Rawlings-Blake	640,000
Philadelphia	PA	M.Nutter	1,500,000
New York	NY	M.Bloomberg	8,400,000
Boston	MA	T.Menino	610,000

Fig. 3. A simple table representing information about cities in United States of America

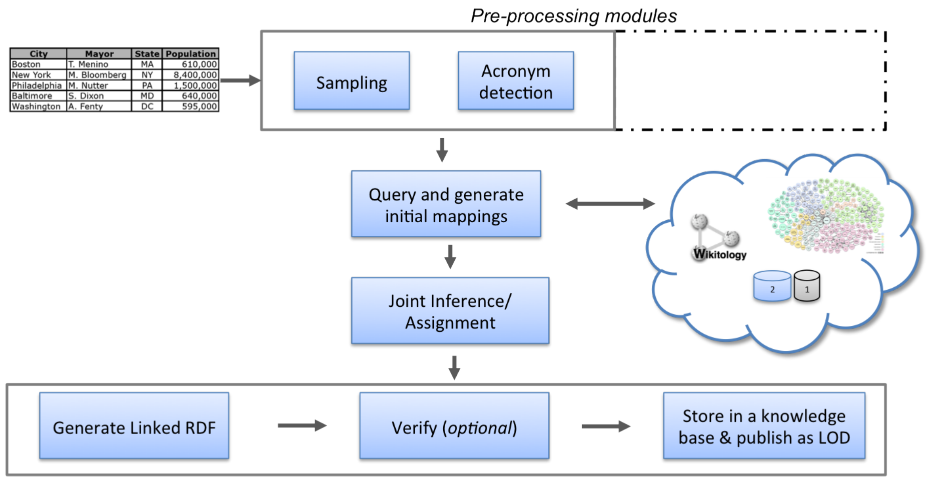
look at the table tells us that the cities in column one are the largest cities of the respective states in column three.

To extract such information from tables, it will be important to interpret the meaning of column (and row) headers, correlation between columns, and entities and literals mentioned in tables. Additional context and information can be also be obtained from caption of the table as well text surrounding the table.

The intended meaning of column headers can be extracted by analyzing the values in the columns. For example, the strings in column one in Figure 3 can be recognized as entity mentions that are instances of the *dbpedia-owl:Place class*. Additional analysis can automatically generate a narrower description such as major U.S. cities. The string in the third column match the names of people and also the narrower class of politicians. The column header provides additional evidence and better interpretation that the strings in column three are the mayors of the cities in column one. Linking the table cell values to known entities enriches the table further. Linking S.C.Rawlings-Blake to *dbpedia:Stephanie_C._Rawlings-Blake*, T.Menino to *dbpedia:Thomas_Menino* , M.Nutter to *dbpedia:Michael_Nutter* we can automatically infer the additional information that all three belong to the Democratic party. Discovering correlations between table columns also add key information. For example, in this case, correlation between columns one and two help us infer that cities in column one are *largestCities* of the respective states in column three.

The techniques above will work well when the table cell values are strings; but not necessarily when cell values are literals, for e.g. numerical values such as the ones from the table in Figure 2 or values from column four of the table in Figure 3. We discuss the challenges posed by such literals and how to tackle them later in the paper.

Producing an overall interpretation of a table is a complex task that requires developing an understanding of the intended meaning of the table as well as attention to the details of choosing the right URIs to represent both the schema as well as instances. We break down the process into following major tasks: (i) assign every column (or row header) a class label from an appropriate ontology; (ii) link table cell values to appropriate LOD entities, if possible; (iii) discover relationships between the table columns and link them to linked data properties; and (iv) generate a linked data representation of the inferred data.



City	Mayor	State	Population
Boston	T. Menino	MA	610,000
New York	M. Bloomberg	NY	8,400,000
Philadelphia	M. Nutter	PA	1,500,000
Baltimore	S. Dixon	MD	640,000
Washington	A. Fenty	DC	595,000

Pre-processing modules
Sampling
Acronym detection
Query and generate initial mappings
Joint Inference/ Assignment
Wikitology
2
1
Generate Linked RDF
Verify (*optional*)
Store in a knowledge base & publish as LOD

Fig. 4. We are developing a robust domain independent framework for table interpretation that will result in a representation of the information extracted as RDF Linked Open Data.

4 DIF-LDT : A Domain Independent Framework

We present DIF-LDT - our domain independent framework for inferring the semantics associated with tables in Figure 4. With little or no domain dependence, the framework should work equally well with tables found on web pages, in medical literature or tabular datasets from sites like data.gov. The goal of this framework is also to address a number of practical challenges, including handling with large tables containing many rows, tables with acronyms and encoded values, and literal data in the form of numbers and measurements.

At the core of the framework are two modules - a) module that queries and generates initial set of mappings for column headers, cell values and relation between columns in a table and b) a module grounded in probabilistic graphical model, which performs joint inference. Once the table passes through initial preprocessing, the query phase generates a set of candidate classes, entities and relations between columns, for every column header and cell values in a table. The module for joint inference will jointly assign values to column headers, cell values and relations between columns in a table. The table interpretation will be useful only when we are able to generate an appropriate representation of it which can be reasoned and queried over by other systems. Thus, the next step would be generating an appropriate representation of the inferred information. Certain applications may require that the user review and if necessary change the generated interpretation. To incorporate this requirement, an additional module provides a interactive framework to allow a human to work with the system to produce the interpretation. In the following sections we describe each module in detail.

4.1 Pre-processing

The goal of the preprocessing modules at the start of the process is for dealing with special cases. For example, certain tables or datasets can be too large to be dealt by the module for joint inference. In such cases, it would better to sample the table, generate a smaller version and let that pass through the rest of workflow. While applying joint inference/joint assignment techniques to large tables is not feasible, we believe that it is also not necessary. We note that people can usually understand a large table's meaning by looking only at its initial portion. Our approach will be similar – given a large table, we will sample the rows to select a smaller number to which we will apply the graphical model. The other pre-processing module we present is for acronyms. Many tables tend to use acronyms. Replacing them with their expanded forms, will provide a more accurate context and thus help in generating a better interpretation. While, we present only two such modules, given the independent nature of the modules, more modules can be easily added without breaking the rest of the workflow.

4.2 Generate and Rank Candidates

The goal of the querying phase is to access knowledge sources and generate a initial set of mappings of classes, entities and relations for each mention in the table. The knowledge sources used in the query process will include datasets such as DBpedia [3], Yago [25] from the LOD cloud. For other specialized domains such as the medical domain or open government data, additional ontologies and knowledge resources may be needed. For general tables, like the ones found on the web, DBpedia, Yago and Wikitology [26] provide very good coverage.

Presently, we use Wikitology, a hybrid kb based on Wikipedia's structured and unstructured information augmented with information from structured sources like DBpedia, Freebase [4], WordNet [17] and Yago, to generate our initial mappings. The query module generates a set of candidate entities for each cell value in a table by querying Wikitology, using query techniques described in [19].

Each returned entity has a set of associated classes (or types). For example, a subset of classes associated with the entity *dbpedia:Baltimore* are *yago:IndependentCitiesInTheUnitedStates*, *dbpedia-owl:PopulatedPlace*, *dbpedia-owl:City*, *yago:CitiesInMaryland*. The set of candidate classes for a given column in a table can be obtained by taking a union of the set of classes associated with the candidate entities in that column. Our current focus is restricted to column headers and entities in a table.

Once the candidate sets are generated, the next step is to rank the candidates. We developed two functions ψ_1 and ψ_2 for this purpose. ψ_1 ranks the candidate classes in a given set, whereas ψ_2 ranks the candidate entities. ψ_1 will compute the 'affinity' between a column header string (e.g., *City*) and a class from the candidate set (say *dbpedia-owl:City*). We define ψ_1 as the exponential of the product of a weight vector and a feature vector computed for column header. ψ_1 will assign a score to each candidate class which can be used to rank the candidate classes. Thus,

$$\psi_1 = exp(w_1^T.f_1(C_i, L_{C_i}))$$

where w_1 is the weight vector, L_{C_i} is the candidate class label and C_i is the string in column header i. The feature vector f_1 is composed of the following features :

$$f_1 = [LevenshteinDistance(C_i, L_{C_i}), DiceScore(C_i, L_{C_i}),\\ SemanticSimilarity(C_i, L_{C_i}), InformationContent(L_{C_i})]$$

f_1 includes a set of string similarity metrics (Levenshtein distance [15], Dice score [24]) to capture string similarity between the class and column header string. To overcome cases where there is no string or content match (e.g. *dbpedia-owl:AdministrativeRegion* and *State*), we also include a metric to capture Semantic Similarity [10] between the candidate class and column header string.

Selecting 'specific' classes is more useful than selecting 'general' classes. For example it is better to infer that a column header is of type of *dbpedia-owl:City* as compared to inferring it as *dbpedia-owl:Place* or *owl:Thing*. Thus, to promote classes of the likes of *dbpedia-owl:City*, f_1 incorporates an Information content measure. Based on semantic similarity defined in [21], we define Information Content as, $I.C(L_C) = -log_2[p(L_C)]$, where $p(L_C)$ is the probability of the class L_C. We computed $I.C.$ for classes from the DBpedia ontology and noticed that specific classes will have a higher value for $I.C.$ as compared to more general classes.

We also develop a function ψ_2 to rank and compute the affinity between the string in the table row cell (say *Baltimore*) and the candidate entity (say *dbpedia:Baltimore*). We define ψ_2 as the exponential of the product of a weight vector and a feature vector computed for a cell value. Once again ψ_2 will assign a score to each entity which can be used to rank the entities. Thus,

$$\psi_2 = exp(w_2^T.f_2(R_{i,j}, E_{i,j}))$$

where w_2 is the weight vector, $E_{i,j}$ is the candidate entity and $R_{i,j}$ is string value in column i and row j. The feature vector f_2 is composed as follows:

$$f_2 = [LevenshteinDistance(R_{i,j}, E_{i,j}), DiceScore(R_{i,j}, E_{i,j}),\\ PageRank(E_{i,j}), KBScore(E_{i,j}), PageLength(E_{i,j})]$$

f_2 is consists a set of string similarity metrics (Levenshtein distance, Dice score) and also a set of popularity metrics(Predicted Page Rank [27], Page Length and Wikitology KB score for the entity). When it is difficult to disambiguate between entities, the more popular entity is likely to be the correct answer; hence the inclusion of popularity metrics. The weight vectors w_1, w_2 can be tweaked via experiments or can be learned using standard machine learning procedures. As we continue to make progress in our work, in the future, we will develop a similar function for ranking candidate relations.

4.3 Joint Inference

Given candidate sets for column headers, cell values and relation between table columns, the joint inference module is responsible for joint assignment to mentions in the table and infer the meaning of a table as a whole. Probabilistic graphical models [13] provide a powerful and convenient framework for expressing a joint probability over a set of variables and performing inference or joint assignment of values to the variables. Probabilistic graphical models use graph based representations to encode probability distribution over a set of variables for a given system. The nodes in such a graph represent the variables of the system and the edges represent the probabilistic interaction between the variables.

Based on the graphical representation used to model the system, the graph needs to be parametrized and then an appropriate inference algorithm needs to be selected to perform inferencing over the graph. Thus constructing a graphical model involves the following steps: (i) identifying variables in the system; (ii) specifying interactions between variables and representing it as a graph; (iii) parameterizing the graphical structure; and (iv) selecting an appropriate algorithm for inferencing. Following this plan, we describe how a graphical model for inferring the semantics of tables is constructed.

Variables in the System. The column headers, cells values (strings and literals) and relation between columns in a table represent the set of variables in an interpretation framework. Each variable has a set of candidates associated, which are generated as described in section 4.2. The initial assignment to each variable will be its top ranked candidate.

Graphical Representation. There are three major representation techniques for encoding the distribution over set of variables: directed models (e.g., Bayesian networks), undirected models (e.g., Markov networks), and partially directed models. In the context of graphical models, Markov networks are undirected graphs in which nodes represent the set of variables in a system and the undirected edges represent the probabilistic interactions between the them. The edges in the graph are undirected because the interaction between the variables are symmetrical. In the case of tables, interaction between the column headers, table cell values and relation between table columns are symmetrical. Thus we choose a Markov network based graphical model for the inferring the semantics of tables.

Figure 5(a) shows the interaction between the variables in a table. In a typical well formed table, each column contains data of a single syntactic type (e.g., strings) that represent entities or values of a common semantic type (e.g., people). For example, in a column of cities, the column header *City* represents the semantic type of values in the column and *Baltimore*, *Boston* and *Philadelphia* are instances of that type. Thus knowing the type (or class) of the column header, influences the decision of the assignment to the table cells in that column and vice-versa.

To capture this influence, we insert an edge between the column header and each of the table cells in the column. Edges between the table cell themselves in

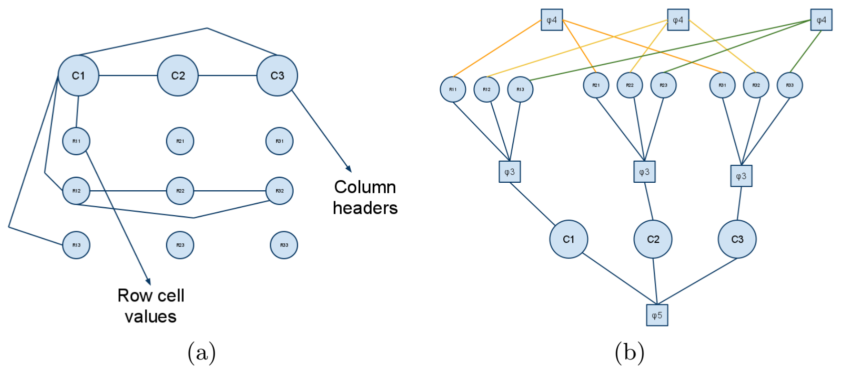


Fig. 5. (a) This graph represents the interactions between the variables in a simple table. Only some of the connections are shown to keep the figure simple. (b) This factor graph is a parameterized Markov network with nodes for variables and factors.

the same column are not needed since they become independent of each other once the column header is known. To keep the figure simple, we show interaction between column header and row values for one column only. The same interactions apply, of course, to the rest of the columns.

Table cells across a given row are also related. Consider a table cell with a value *Beetle*. It might be referring to an insect or a car. The next table cell has a value *red* which is a color. The value in the last table cell is *Gasoline*, a type of fuel source. All the values considered together, indicate that the row is representing values of a car rather than an insect. This can be further confirmed by the evidence from type of the column header. Thus to disambiguate a table cell correctly, the context from the rest of table cells in the row should be used. To capture this context, we insert edges between all the table cells in a given row. Again for simplicity, we show interaction between the table cells of only one row.

Similar interaction also exist between the column headers. The column header *City* might suggest that the strings in the columns are cities. However if *City* appears in the table with other columns which are Basketball players, Coach and Division, we can infer that the column of cities is referring to a team itself – an example of metonymy in which the team is referenced by one of its significant properties, the location of it's base. Thus to correctly disambiguate what the column header means, we should use evidence from rest of the column headers as well. To capture this interaction, we insert edges between all the column headers as well.

The model we have presented captures interactions between the column headers and table cell values. Our initial plan is to experiment with the current model and then further extend it to incorporate the variable to capture relations between table columns in the graph as well.

Parametrizing the network. To represent the distribution associated with the graph structure, we need to parameterize the structure. Since Markov networks are undirected, the goal of parameterization is to capture the *affinity* between the interacting variables. That is if variable A is assigned a value a_1 and variable B is assigned b_1, the question we ask is whether A and B are likely to agree or not. The function that computes this *affinity* is known as a *factor*.

One way to parameterize a Markov network is to represent it as a *factor graph*. A factor graph is an undirected graph containing two types of nodes: variables and factors. The graph has edges only between the factor nodes and variable nodes. The factor nodes are responsible for computing the *affinity* between the variable nodes to which it is connected. Figure 5(b) presents the parametrized network. The column headers (all C_i) and cell values (all R_{ij}) are the variable nodes and ψ_3, ψ_4 and ψ_5 are factor nodes in the graph.

ψ_3 is a factor node that captures the affinity between the class label assigned to a column header and the entities linked to the cell values in the same column; i.e., its goal is to check whether *dbpedia-owl:City, dbpedia:Baltimore, dbpedia:-Philadelphia* and *dbpedia:Boston* are 'compatible' or not. ψ_3 has two goals - to detect if any of cell values have been linked to incorrect entities or if the class label assigned to the column header is incorrect.

Each entity is associated with a set of classes (or types). For example the classes associated with *dbpedia:Baltimore* include *dbpedia:City*, *yago:IndependentCitiesInTheUnitedStates*, and *yago:CitiesInMaryland*. ψ_3 will assign a score to the class mapped to the column header as follows. Each entity in the column will contribute a score between zero and one to the class. The score contributed will be 0.0 if the class assigned to the column header is not in the set of classes associated with the entity and 1.0 if its the best match. An entity will contribute a lower score if the class assigned to the column header has a more "descriptive" or specific class in its set. For example, the class *dbpedia:City* is more descriptive and informative as compared to *dbpedia:Place*. The score assigned to the class label will be the average of the sum of scores assigned by each of the individual entities in the column.

What inferences can be drawn from the score? A score of 0.0 indicates either that the class label assigned is incorrect or all the entities linked to the cell values are incorrect. A score of 1.0 strongly suggests that class and entity assignments are correct. Scores tending towards 1.0 will indicate higher level of agreement whereas scores closer to 0.0 indicate less agreement. We will discuss how this information is used in the section on inferencing over the graph.

ψ_4 is a factor node that captures the affinity between the entities linked to the values in the table cells in a given row in the table, i.e., the affinity between *dbpedia:Baltimore*, *dbpedia:Maryland*, and *dbpedia:Stephanie_Rawlings--Blake*. Entities across a given row are likely to be related to each other. We use Pointwise mutual information (PMI) as a measure to capture the relatedness between entities. The PMI between totally unrelated entities will be zero, whereas the value will be non zero for related entities. For example, the PMI between *dbpedia:Baltimore* and *dbpedia:Seattle_Seahawks* will be 0.0, since they are unrelated. We have computed PMI values for Wikitology entities based on statistics derived from Wikipedia and DBpedia.

The factor node ψ_4 will compute pairwise PMI for a entity with the rest of entities in the row. Thus, in this case, for *dbpedia:Baltimore*, ψ_4 will compute PMI between *dbpedia:Baltimore* , *dbpedia:Maryland* and *dbpedia:Baltimore* , *dbpedia:Stephanie_Rawlings-Blake*. If a cell value in the row is mapped to a incorrect entity, the entity will have zero PMI with every other entity present in the row. This can be used as a indicator to detect incorrect mapping for a given cell value.

Similarly, ψ_5 is a factor node that captures the affinity between classes that have been assigned to all the column headers in the table, i.e., the affinity between *dbpedia-owl:City*, *dbpedia-owl:AdministrativeRegion*, and *dbpedia-owl:Mayor*. We again rely on the PMI data to capture the association between the class labels assigned to column headers. For every class label, ψ_5 will compute PMI between the class label and each of the other class labels across the column header. A unrelated class label will have zero PMI with every other class label, which can be used as indicator of incorrect mapping.

The factor nodes are dependent on measures such as *PMI* and the *score* assigned by entities to a class. These measures can be computed from the domain knowledge source. We see these functions as first iteration and expect them to evolve as continue to experiment. Similarly as the graph is extended to incorporate relations between table columns, more factor nodes are likely to be added.

Inference. The objective of the inference process is to determine the best possible assignments to the column headers and table cell values and eventually relations between table columns. We will use a variation of message-passing/belief propagation [13] for this process. Our inference algorithm will work as follows. The process will start with the variable nodes sending a message to all its connected factor nodes. The message will be composed of the current value assigned to the variable node.

Once a factor node receives messages from all the connected variable nodes, it will compute if the values assigned are in agreement or not. If the values are not in agreement, it will identify the variable node(s) that may have a wrong assignment. For all the nodes with possible wrong assignments, the factor node will send a message requesting the node to change its assignment to a different value. It will also send a confidence measure; i.e., how confident is the factor node about the assertion that the variable node has a wrong assignment. For the variable nodes with correct assignment, the factor node will send a message of ‘no change’.

The factor nodes will use the functions defined above to determine agreement. For example, the factor node ψ_4 can conclude that all the values are in agreement, if every entity assigned in a row is connected (i.e., has non-zero PMI) with at least one other entity in the same row. ψ_4 will send a message of change assignment to a variable, if the entity assigned has zero PMI will all other entities in the row. Similarly if the score assigned to a class mapped to the column header is one, ψ_3 will conclude that the class and the entities in the column are in agreement.

Once a variable node receives messages from all the factor nodes to which it is connected, the variable nodes determines whether it needs to change its assigned value or not. For example, if a node receives a 'no change' message from all factor nodes, the variable node will not change its value. If the node receives a change message from all the factor nodes, the node will change its value and select a new assignment. In the long run, our goal will be to develop a generic function that nodes will use to decide whether to change values or not. The function will take into account factors such as the 'change' and 'no change' messages received, confidence associated with each message and overall 'global temperature'. This last measure will capture whether most (or all) variable nodes are satisfied with their assignment or not. If there is high level of agreement/satisfaction across all nodes, the variable told to change its assignment with low confidence, may not change. We expect that this mechanism will ensure or at least promote convergence to a solution, much like the temperature concept of simulated annealing.

All variable nodes which change their assigned value to a new one send a message to all their connected factor nodes, announcing the update as well the updated value. Every factor node which receives such a message, will redo the computation with the new value and the above process is repeated. This continues until all the variables do not change their values. To ensure that this process converges, variations like penalizing a change in assignment as the number of iterations increase, will be included.

4.4 Generate Linked Data

The table interpretation will be useful only when we are able to generate an appropriate representation of it which can be reasoned and queried over by other systems. Figure 6 presents our preliminary template for representing the inferred information as Semantic Linked Data.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix dbpedia: <http://dbpedia.org/resource/>.
@prefix dbpedia-owl: <http://dbpedia.org/ontology/>.
@prefix dbpprop: <http://dbpedia.org/property/>.

"City"@en is rdfs:label of dbpedia-owl:City.
"State"@en is rdfs:label of dbpedia-owl:AdminstrativeRegion.
"Baltimore"@en is rdfs:label of dbpedia:Baltimore.
dbpedia:Baltimore a dbpedia-owl:City.
"MD"@en is rdfs:label of dbpedia:Maryland.
dbpedia:Maryland a dbpedia-owl:AdministrativeRegion.
```

Fig. 6. A preliminary represented of inferred knowledge from a table

In the future, we will extend our preliminary template and develop an richer ontology for annotating and representing tabular data as linked data. The ontology will provide terms for expressing the provenance of the interpretation and annotating some mappings certainty information. With the use of a probabilistic graphical model, it is possible to generate certainty information. Since we are dealing with tables on the web, different tables are likely to generate contradictory information. Thus provenance and source of tables will be important to applications reasoning over our data. Our representation will also capture table meta data such as number of columns and rows as well as the table in its raw form. We wish to allow applications to be able to reconstruct the original table from the linked data representation.

4.5 Human in the loop

Since achieving perfect accuracy in automatically translating a table into linked data is infeasible, we will develop a interactive framework to allow a human to work with the system to produce the interpretation. Our approach will have two phases: interpretation exploration and providing feedback. The first phase will allow a person to inspect and explore the interpretation, determining the entities, concepts and relations to which table components are mapped. The human user will also be able to see the ranked list of alternatives for each mapping along with any associated scores or confidence measures. The second phase will permit the user to identify one of the mappings an incorrect and optionally select and "lock down" one of the alternate interpretations. The system will then rerun the graphical model, producing a new overall interpretation. This sequence of inspection and correction can be repeated until an acceptable interpretation is produced.

4.6 Challenges – Literals

Literals pose unique challenges. Unlike strings in table cells, literals are not entities that can be linked to existing entities from a knowledge base; but rather they represent values of properties. The properties themselves can be associated with other entities in the table. Thus techniques that will work with string based values will not necessarily work for literals.

So how do we treat literals such as numerical data ? We can take the intuition that humans use to understand columns of numerical data as a starting point. To begin with, the range of numbers in a given column can start providing evidence about what the column is about. If a person looks at a column (without a column header) that contains numbers in the range of 0–100, the person is likely to infer that the column could be percentages or ages. The row (or column) header may have additional clues. For example, in the case of percentages, the % sign maybe associated with the numbers in the table cell or it may be present in the row (or column) header in the table.

Successfully understanding numerical data, will require understanding what properties do values from a column map to and extracting unit associated with numbers or unit symbols (like %).

5 Related Work

Our work is closely related to two threads of research. The first focuses on generating RDF and linked data from sources such as databases, spreadsheets and CSV files. The second, and more recent one, addresses understanding and inferring the implicit semantics of tables.

Several systems have been implemented to generate semantic web data from databases [23, 28, 20], spreadsheets [11, 14] and CSV files [8]. All are manual or at best partially automated and none have focused on automatically generating *linked* RDF data for the entire table. These systems have mainly focused on relational databases where the schema is available or on simple spreadsheets. In the domain of open government data, [8] presents techniques to convert raw data (CSV, spreadsheets) to RDF. However the generated RDF data does not use existing classes or properties for column headers, nor does it link cell values to entities from the LOD cloud. To generate a richer, enhanced mappings, users will need to manually specify a configuration file. Their focus has been on generating massive quantity linked government data rather quality linked government data.

The key shortcoming in such systems is that they rely heavily on users and their knowledge of the Semantic Web. Most systems do not automatically link classes and entities generated from their mapping to existing Semantic Web resources – their output turns out to be just *raw string data* represented as RDF, instead of fully linked RDF. Figure 7 shows a part of RDF representation of dataset 1425 from *data.gov* [7]. The property names in the representation are column headers from the raw dataset and the values of the properties represent row values for the respective columns.

```
<rdf:Description rdf:about="#entry1">
<value>6444</value>
<label>Number of Farms</label>
<group>Farms with women principal operators</group>
<county_fips>000</county_fips>
<state_fips>01</state_fips>
<state>Alabama</state>
<rdf:type rdf:resource="http://data-gov.tw.rpi.edu/2009
/data-gov-twc.rdf#DataEntry"/>
</rdf:Description>
```

Fig. 7. A portion of the RDF representation from dataset 1425 - Census of Agriculture Race, Ethnicity and Gender Profile Data from data.gov.

The representation fails to use existing vocabulary terms to annotate the raw data and most of the column headers are mapped to properties local to the RDF file. Mapping column headers to classes and properties from the LOD cloud, provides richer description as compared to the local properties. Such a representation often uses string identifiers for table cell values instead of linking them to existing entities in the LOD cloud. Linking the string cell values can further enrich the semantic representation of the data. Our framework will link and reuse existing classes, properties and entities with dereferenceable URIs from the LOD cloud. Our goal is to generate linked data in a form which is identified as "five star" by Tim Berners-Lee [1].

Early work in table understanding focused on extracting tables from documents and web pages [12, 9] with more recent research attempting to understand their semantics. Wang et al. [30] began by identifying a single 'entity column' in a table and, based on its values and rest of the column headers, associates a concept from the Probase [31] knowledge base with the table. Their work does not attempt to link the table cell values or identify relations between columns. Ventis et al. [29] associate multiple class labels (or concepts) with columns in a table and identify relations between the 'subject' column and the rest of the columns in the table. Both the concept identification for columns and relation identification is based on maximum likelihood hypothesis, i.e., the best class label (or relation) is one that maximizes the probability of the values given the class label (or relation) for the column. Their work also does not attempt to link the table cell values. Limaye et al. [16] use a graphical model which maps every column header to a class from a known ontology, links table cell values to entities from a knowledge-base and identifies relations between columns. They rely on Yago for background knowledge.

The core of our framework is a probabilistic graphical model that captures a much richer semantics, including relation between column headers as well relation between entities across a given row. Our model has a single 'factor' node to capture relation between column header and strings in the column, which makes it possible to deal with missing values (e.g., absent column header).

Current systems for interpreting tables rely on semantically poor and possibly noisy knowledge-bases and do not attempt to produce a complete interpretation of a table. None of the current systems propose or generate any form of linked data from the inferred meaning. The work mentioned above will work well with string based tables but we know of no systems that interpret columns with numeric values and use the results as evidence in the table interpretation. Doing so is essential for many domains, including medical research.

6 Discussion and Conclusion

We built a baseline system [19] to evaluate the feasibility in tacking the problem. The baseline system was a sequential system that did three things : i) predict class labels for column headers ii) link table cell values to entities and iii) discover correlation between column headers. We evaluated our baseline system using a dataset of 15 tables obtained from the web, Google Squared and tables from Wikipedia articles. Excluding the columns with numbers, the 15 tables have 52 columns and 611 entities for evaluation of our algorithms. We used a subset of 23 columns for evaluation of relation identification between columns.

In the first evaluation of the algorithm for assigning class labels to columns, we compared the ranked list of possible class labels generated by the system against the list of possible class labels ranked by the evaluators. For 80.76% of the columns the average precision between the system and evaluators list was greater than 0 which indicates that there was at least one relevant label in the top three of the system ranked list. The mean average precision for 52 columns was 0.411.For 75% of the columns, the recall of the algorithm was greater than or equal to 0.6. We also assessed whether our predicted class labels were reasonable based on the judgement of human subjects. 76.92% of the class labels predicted were considered correct by the evaluators. 66.12% of the table cell strings were correctly linked by our algorithm for linking table cells. Our dataset had 24 new entities and our algorithm was able to correctly predict for all the 24 entities as new entities not present in the KB. We did not get encouraging results for relationship identification with an accuracy of 25%.

Analysis of our evaluation provided useful lessons. First, we noticed, with a sequential system, the error percolated from stage one to stage three, thus leading to an overall poor interpretation of the semantics of tables. This lead us to developing a framework based around probabilistic graphical model for joint inference over a table. Secondly, our baseline system was giving preference to 'general classes' over 'specific classes', which we address by introducing measures like Information Content of a class. Our framework also goes beyond web-tables and aims to deal with tables across multiple domains – from medical literature to open government data. We are in the process of implementing the graphical model. Once the model is implemented, we will evaluate it against a dataset of tables shared by Limaye et al. [16].

Generating an explicit representation of the meaning implicit in tabular data will support automatic integration and more accurate search. We described general techniques grounded in graphical models and probabilistic reasoning to infer a tables meaning relative to a knowledge base of general and domain-specific knowledge expressed in the Semantic Web language OWL. We represent a table's meaning as a graph of OWL triples where the columns have been mapped to classes, cell values to literals, measurements, or knowledge-base entities and relations to triples. We believe that knowledge recovered from tables can enable and assist various application and lead us towards a web of semantics, concepts and entities.

7 Acknowledgements

This research was supported in part by NSF awards 0326460 and 0910838, MURI award FA9550-08-1-0265 from AFOSR, and a gift from Microsoft Research.

References

1. Berners-Lee, T.: Linked data. http://www.w3.org/DesignIssues/LinkedData.html (July 2006)
2. Bizer, C.: The emerging web of linked data. IEEE Intelligent Systems 24(5), 87–92 (2009)
3. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia - a crystallization point for the web of data. Journal of Web Semantics 7(3), 154–165 (2009)
4. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proc. ACM Int. Conf. on Management of Data. pp. 1247–1250. ACM (2008)
5. Cafarella, M.J., Halevy, A.Y., Wang, Z.D., Wu, E., Zhang, Y.: Webtables: exploring the power of tables on the web. PVLDB 1(1), 538–549 (2008)
6. Cohen, A., Adams, C., Davis, J., Yu, C., Yu, P., Meng, W., Duggan, L., McDonagh, M., Smalheiser, N.: Evidence-based medicine, the essential role of systematic reviews, and the need for automated text mining tools. In: Proc. 1st ACM Int. Health Informatics Symposium. pp. 376–380. ACM (2010)

7. Dataset 1425 - Census of Agriculture Race, Ethnicity and Gender Profile Data. http://explore.data.gov/Agriculture/Census-of-Agriculture-Race-Ethnicity-and-Gender-Pr/yd4n-fk45, 2009
8. Ding, L., DiFranzo, D., Graves, A., Michaelis, J.R., Li, X., McGuinness, D.L., Hendler, J.A.: Twc data-gov corpus: incrementally generating linked government data from data.gov. In: Proc 19th Int. Conf. on the World Wide Web. pp. 1383–1386. ACM, New York, NY, USA (2010)
9. Embley, D.W., Lopresti, D.P., Nagy, G.: Notes on contemporary table recognition. In: Document Analysis Systems. pp. 164–175 (2006)
10. Han, L., Finin, T., McNamee, P., Joshi, A., Yesha, Y.: Improving word similarity by augmenting pmi with estimates of word polysemy. IEEE Transactions on Knowledge and Data Engineering (2012)
11. Han, L., Finin, T., Parr, C., Sachs, J., Joshi, A.: RDF123: from Spreadsheets to RDF. In: Proc. 7th Int. Semantic Web Conf. Springer (October 2008)
12. Hurst, M.: Towards a theory of tables. IJDAR 8(2-3), 123–131 (2006)
13. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
14. Langegger, A., Wob, W.: Xlwrap - querying and integrating arbitrary spreadsheets with SPARQL. In: Proc. 8th Int. Semantic Web Conf. (October 2009)
15. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Tech. Rep. 8, Soviet Physics Doklady (1966)
16. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. In: Proc. 36th Int. Conf. on Very Large Databases (2010)
17. Miller, G.A.: Wordnet: a lexical database for english. CACM 38, 39–41 (Nov 1995)
18. Mulwad, V.: T2LD - An automatic framework for extracting, interpreting and representing tables as Linked Data. Master's thesis, U. of Maryalnd, Baltimore County (August 2010)
19. Mulwad, V., Finin, T., Syed, Z., Joshi, A.: Using linked data to interpret tables. In: Proc. 1st Int. Workshop on Consuming Linked Data. Shanghai (2010)
20. Polfliet, S., Ichise, R.: Automated mapping generation for converting databases into linked data. In: Proc. 9th Int. Semantic Web Conf. (November 2010)
21. Resnik, P.: Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. Journal of Artificial Intelligence Research 11(1), 95–130 (1999)
22. Sackett, D., Rosenberg, W., Gray, J., Haynes, R., Richardson, W.: Evidence based medicine: what it is and what it isn't. Bmj 312(7023), 71 (1996)
23. Sahoo, S.S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr, T., Auer, S., Sequeda, J., Ezzat, A.: A survey of current approaches for mapping of relational databases to rdf. Tech. rep., W3C (2009)
24. Salton, G., Mcgill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York, NY, USA (1986)
25. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: 16th Int. World Wide Web Conf. ACM Press, New York (2007)
26. Syed, Z., Finin, T.: Creating and Exploiting a Hybrid Knowledge Base for Linked Data. Springer (April 2011)
27. Syed, Z., Finin, T., Mulwad, V., Joshi, A.: Exploiting a Web of Semantic Data for Interpreting Tables. In: Proc. 2nd Web Science Conf. (April 2010)
28. Vavliakis, K.N., Grollios, T.K., Mitkas, P.A.: Rdote - transforming relational databases into semantic web data. In: Proc. 9th Int. Semantic Web Conf. (November 2010)

29. Venetis, P., Halevy, A., Madhavan, J., Pasca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering semantics of tables on the web. In: Proc. 37th Int. Conf. on Very Large Databases (2011)
30. Wang, J., Shao, B., Wang, H., Zhu, K.Q.: Understanding tables on the web. Tech. rep., Microsoft Research Asia (2011)
31. Wu, W., Li, H., Wang, H., Zhu, K.: Towards a probabilistic taxonomy of many concepts. Tech. rep., Microsoft Research Asia (2011)
32. Zagari, R., Bianchi-Porro, G., Fiocca, R., Gasbarrini, G., Roda, E., Bazzoli, F.: Comparison of 1 and 2 weeks of omeprazole, amoxicillin and clarithromycin treatment for helicobacter pylori eradication: the hyper study. Gut 56(4), 475 (2007)