DC Proposal: Graphical Models and Probabilistic Reasoning for Generating Linked Data from Tables*

Varish Mulwad

Computer Science and Electrical Engineering
University of Maryland, Baltimore County
varish1@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. We describe techniques grounded in graphical models and probabilistic reasoning to infer meaning associated with a table. Using background knowledge from 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, Tables, Entity Linking, Machine Learning, Graphical Models.

1 Introduction

Most of the information found on the Web consists of text written in a conventional style, e.g., as news stories, blogs, reports, letters, advertisements, etc. There is also a significant amount of information encoded in structured forms like tables and spreadsheets, including stand-alone spreadsheets or table as well as tables embedded Web pages or other documents. Cafarella et al. [2] estimated that the Web contains over 150 million high quality relational tables. In some

* Advisor: Tim Finin, University of Maryland, Baltimore County. 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.

L. Aroyo et al. (Eds.): ISWC 2011, Part II, LNCS 7032, pp. 317–324, 2011.

ways, this information is easier to understand because of its structure but in other ways it is more difficult because it lacks the normal organization and context of narrative text. Both integrating or searching over this information will benefit from a better understanding of its intended meaning.

A wide variety of domains that are interesting both technically and from a business perspective have tabular data. These include medicine, healthcare, finance, e-science (e.g., biotechnology), and public policy. Key information in the literature of these domains, which can be very useful for informing public policy, is often encoded in tables. As a part of Open Data and transparency initiative, fourteen nations including the United States of America share data and information on websites like www.data.gov in structured format like CSV, XML. As of May 2011, there are nearly 390,000 raw datasets available. This represents a large source of knowledge, yet we do not have systems that can understand and exploit this knowledge.

In this research, we will present techniques to evaluate our claim that "It is possible to generate high quality linked data from tables by jointly inferring the semantics of column headers, values (string and literal) in table cells, relations between columns, augmented with background knowledge from open data sources such as the Linked Open Data cloud."

2 Motivation

Ever since its inception in 2001, the Semantic Web has laid strong foundations for representing and storing knowledge in machine understandable formats such as RDF and OWL. The principles of Linked Data further strengthens the move from a web of documents to a web of data.

While the Semantic Web was able to lay strong foundations, its growth remains slow because of the lack of quality of data available on the Semantic Web. Even though data.gov has more than 390,000 datasets, only 0.071% of those are available as RDF datasets. Existing technology on the Semantic Web either rely on user's knowledge or generate "low quality" data which in some cases is as useless as raw data. Our proposed framework can easily be used to convert legacy datasets stored in tabular formats to RDF and publish it on the Semantic Web and the Linked Data Cloud. Not only does our framework generate RDF from tables, but it also produces high quality linked RDF.

Many real world problems and applications can benefit from exploiting information stored in tables including evidence based medical research [14]. 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. Most of this information is encoded in tables, which are currently beyond the

scope of regular text processing systems and search engines. By adding semantics to such tables, 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.

3 Related Work

Several systems have been implemented to generate Semantic Web data from databases [15,19,12], spreadsheets [5,7] and csv [3] . Virtually all are manual or semi-automated and none have focused on automatically generating *linked* RDF data. Current systems on the Semantic Web either require users to specify the mapping to translate relational data to RDF or systems that do it automatically focus only on a part of the table (like column header strings). These systems have mainly focused on relational databases or simple spreadsheets.

The key shortcoming in such systems is that they rely heavily on users and their knowledge of the Semantic Web. Most systems on the Semantic Web also do not automatically link classes and entities generated from their mapping to existing resources on the Semantic Web. The output of such systems turns out to be just "raw string data" represented as RDF, instead of generating high quality linked RDF.

In the web tables domain, Wang et al. [21] present a table understanding system which identifies a concept to be associated with the table based on the evidence provided by the column header and strings in the "entity column" of the table. The concepts come from their knowledge base Probase created from the text on the World Wide Web which can be noisy and "semantically poor" as compared to concepts from the Linked Open Data cloud.

Ventis et al. [20] identify concepts to be associated with the column headers in a table based on the evidence provided by strings in a given column. They also identify relations between the "subject column" and other columns in the table. However they also rely on a isA database they create from the text on the Web which can be noisy as well as "semantically poor".

Limaye et al. [8] present a probabilistic graphical model based framework that identifies concepts to be associated with the column headers, links table cell values to entities and identifies relations between columns with Yago as a background knowledge base.

None of the current table understanding systems propose or generate any form of linked data from the inferred meaning. A key missing component in current systems is tackling literal constants. The work mentioned above will work well with string based tables. To the best of our knowledge, no work has tackled the problem on interpreting literals in tables and using them as evidence in the table interpretation framework. The framework we present is complete automated interpretation of a table that focuses on all aspects of a table - column headers, row values, relations between columns. Our framework will tackle strings as well as literals.

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

(a)

```
@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.
```

(b)

Fig. 1. This example shows a simple table about cities in the United States and some output of the prototype system that represents the extracted information as linked data annotated with additional metadata

4 Interpreting a Table

Generating high quality linked data from a table requires understanding its intended meaning. The meaning of a table is often encoded in column headers, table cells and implicitly conveyed via structure and relations between columns in a table.

Consider the table shown in Figure 1(a). The column headers suggest the type of information in the columns: *city* and *state* might match classes in a target ontology such as DBpedia [1]; *mayor* and *population* could match properties in the same or related ontologies. Examining the data values, which are initially just strings, provides additional information that can confirm some possibilities and disambiguate between possibilities for others. For example, the strings in column one 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 cities located in the United States.

Consider the strings in column three. The string by themselves suggest that they are politicians. The column header provides additional evidence and better interpretation that the strings in column three are actually mayors. Discovering relations between columns is important as well. By identifying relation between column one and column three, we can infer that the strings in column three are mayors of cities presented in column one. Linking the table cell values to known entities enriches the table further. Linking S.Rawlings to *dbpedia:Stephanie_C._Rawlings-Blake*, T.Menino to *dbpedia:Thomas_Menino*, M.Nutter to *dbpedia:Michael_Nutter* we can automatically infer additional information that all three belong to the Democratic party, since the information will be associated with the linked entities.

Column four in this table presents literal values. The numbers in the column are values of the property *dbpedia-owl:populationTotal* and this property can be associated with the cities in column one. All the values in the column are in the range of 100,000. They provide evidence that the column may be representing the property population. Once relation between column one and column four is discovered, we can also look up on DBpedia, where the linked cities in column one will further confirm that the numbers represent population of the respective cities.

Producing an overall interpretation of a table is a complex task that requires developing an overall 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 tasks: (a) assign every column (or row header) a class label from an appropriate ontology; (b) link table cell values to appropriate linked data entities, if possible; (c) discover relationships between the table columns and link them to linked data properties; and (d) generate a linked data representation of the inferred data.

5 Approach

We first developed a baseline system to evaluate the feasibility of tackling the problem. The baseline system is a sequential multi-step framework which first maps every column header to a class from an appropriate ontology. Using the predicted class as additional evidence, the frameworks then links table cell values to entities from the Linked Data Cloud. The final step in the framework is discovering relations between table columns. Once this information is inferred, the framework generates a linked data representation of the interpretation (see figure 1(b)). The details of the baseline system and its evaluation is described in [11]. Based on the evaluation of our baseline system, we present a framework grounded in the principles of graphical models and probabilistic reasoning.

The baseline system makes local decision at each step of the framework. The disadvantage of such a system is that error can percolate from the previous phase to the next phase in the system thus leading to an overall poor interpretation of a table. To overcome this problem, we need to develop a framework that performs joint inference over the evidence available in the table and jointly assign values to the column headers, table cells and relations between columns. Probabilistic graphical models [6] provide a convenient framework for expressing a joint probability over a set of variables and perform inferencing over them.

Constructing a graphical model involves the following steps: a) Identifying variables in the system b) Identifying interactions between variables and representing it as a graph c) Parametrizing the graphical structure d) Selecting an appropriate algorithm for inferencing. In the following sections we first present our work on the first three tasks in constructing a graphical model.

Variables in the System. The column headers, the table cells and the relation between columns in a table represent the set of variables in an interpretation framework.

Graphical Representation. We choose a Markov network based graphical representation,since the interaction between the column headers, table cell values and relation between table columns are symmetrical. The interaction between a column header and cell values in the column is captured by inserting an edge between the column header and each of the values in the column in the graph. To correctly disambiguate what a table cell value is, evidence from the rest of the

values in the same row can be used. This is captured by inserting edges between every pair of cell values in a given row. Similar interaction exists between the column headers and is captured by the edges between every pair of table column headers.

Parametrizing the Network. To represent the distribution associated with the graph structure, we need to parametrize the structure. One way to parametrize a Markov network is representing the graph as a factor graph. A factor graph is an undirected graph containing two types of nodes : variable nodes and factor nodes. A factor node captures and computes the afinity between the variables interacting at that factor node. Variable nodes can also have associated "node potentials". Our parametrized model has two node potentials - ψ_1 and ψ_2 and three factor nodes - ψ_3, ψ_4 and ψ_5.

The node potential ψ_1 captures the affinity between the column header string in the table and the class to which the column header is being mapped, i.e., the affinity between *State* and *dbpedia:AdminstrativeRegion*. We define ψ_1 as the exponential of the product of a weight vector and a feature vector computed for column header. 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 class label associated with column header C_i. The feature vector f_1 is composed of the following features : the Levenshtein distance, Dice score [16], semantic similarity between the column header string and the class label and the information content of the class label. We use the semantic similarity measure defined in [4].

We also compute the Information content for a given class in an ontology. Based on the semantic similarity and information content measures defined in [13], the information content for a class in given ontology is defined as follows: $I.C(L_C) = -log_2[p(L_C)]$ where $p(L_C)$ is the probability of the class L_C. We compute the probability by counting the number of instances that belong to the class L_C and divide it by the total number of instances. More specific classes in an ontology present more information as compared to more general classes. For example it is better to infer that a column header is of type of *dbpedia-owl:City* as compared to inferring that as *dbpedia-owl:Place* or *owl:Thing*. The information content measure precisely captures that.

ψ_2 is the node potential that captures affinity between the string in the table cell and the entity that the cell is being mapped to, for example, affinity between *Baltimore* and *dbpedia:Baltimore_Maryland*. We define ψ_2 as the exponential of the product of a weight vector and a feature vector computed for a cell value. 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 entity associated with the value in table cell $R_{i,j}$. The feature vector f_2 is composed of the following features : Levenshtein distance, Dice score, Page Rank, Page Length and Wikitology[17] index score. Along with a set of similarity metrics, we choose a set of popularity metrics, since when it is difficult to disambiguate the more popular entity is more likely the correct answer. Presently, the popularity metrics are Wikipedia based metrics, but these can be easily changed and adapted to a more general sense of popularity. The weight vectors w_1, w_2 can be learned using standard machine learning procedures.

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. The behavior (function) of the factor node is still to be defined. 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_Maryland, dbpedia:Maryland dbpedia:Stephanie_Rawlings-Blake*. We define ψ_4 as the product of Point wise mutual information between each pair of entities in the given row. Thus, $\psi_4 = \Pi_{i,k,i \neq k} PMI(E_{i,j}, E_{k,j})$, where $PMI(E_{i,j}, E_{k,j})$ computes the point wise mutual information between the entity $E_{i,j}$ in column i, row j and $E_{k,j}$ in column k, row j. Point wise mutual information between any two entities will provide us the association between the two – in some sense it is the probability of their co-occurrence. If the entities are associated (which will be the case in the context of table rows), PMI will be high. If the entities are not associated, PMI will be low.

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 dbpedia-owl:Mayor*. We again rely on point wise mutual information to capture the association between the class labels assigned to column headers. We define ψ_5 as the product of Point wise mutual information between each pair of column headers in the table.

6 Evaluation Plan

We will use two different types of evaluation to measure the effectiveness of our proposed techniques. In the first evaluation, we will compute accuracy for correctly predicted class labels for column headers, entities linked to table cells and relation between columns using the ground truth from the dataset of over 6000 web tables from [8]. While we have proposed a automatic framework for interpreting and representing tabular data as linked data, it may be helpful to develop a framework with human in the loop to make the linked data more useful and customized for certain applications. For such cases, our algorithms can generate a ranked list of candidates for each of the column headers, table cells and relation between columns. We will use Mean Average Precision [9] to compare the ranked list generated by our algorithms against a ranked list produced by human evaluators.

References

1. 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)
2. 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)
3. 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 (2010)

4. Han, L., Finin, T., McNamee, P., Joshi, A., Yesha, Y.: Improved pmi utility on word similarity using estimates of word polysemy. TKDE (2011) (under review)
5. Han, L., Finin, T., Parr, C., Sachs, J., Joshi, A.: RDF123: From Spreadsheets to RDF. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 451–466. Springer, Heidelberg (2008)
6. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
7. Langegger, A., Wöß, W.: XLWrap – Querying and Integrating Arbitrary Spreadsheets with SPARQL. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 359–374. Springer, Heidelberg (2009)
8. 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)
9. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval, 1st edn. Cambridge University Press (July 2008)
10. Mulwad, V., Finin, T., Syed, Z., Joshi, A.: T2LD: Interpreting and Representing Tables as Linked Data. In: Proc. Poster and Demonstration Session at the 9th Int. Semantic Web Conf. (November 2010)
11. 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)
12. Polfliet, S., Ichise, R.: Automated mapping generation for converting databases into linked data. In: Proc. 9th Int. Semantic Web Conf. (November 2010)
13. 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)
14. 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)
15. 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)
16. Salton, G., Mcgill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York (1986)
17. Syed, Z., Finin, T.: Creating and Exploiting a Hybrid Knowledge Base for Linked Data. Springer, Heidelberg (April 2011)
18. 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)
19. Vavliakis, K.N., Grollios, T.K., Mitkas, P.A.: Rdote - transforming relational databases into semantic web data. In: Proc. 9th Int. Semantic Web Conf. (2010)
20. 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)
21. Wang, J., Shao, B., Wang, H., Zhu, K.Q.: Understanding tables on the web. Tech. rep., Microsoft Research Asia (2011)