Tabular Data to Knowledge Graph Matching

Kushagra Singh BISEN[1][0000−0003−0950−6043]

Université de Lyon, CNRS UMR 5516 Laboratoire Hubert Curien, F-42023, Saint-Etienne, France
kushagra.bisen@etu.univ-st-etienne.fr

Abstract. The document presents a proposal for conceptualizing the Tabular Data to Knowledge Graph Matching challenge

Keywords: Semantic Web · Knowledge Graph · Tabular Data · RDF · Matching

1 Introduction

Tabular Data is an input format normally employed for generating a Knowledge graph. It is easy to browse through and friendly for people who are not computer scientists as well. However, it is not easy to generate a Knowledge graph from Tabular Data. The problem mostly arises from deciding the structure of the table and deciding which columns signify which value. We wish to create a Knowledge Graph to make better sense of the data, and poor semantic understanding of the tabular data will defeat its purpose. In this document, the proposal is to generate a method to match the tabular data to the Knowledge Graph based on the SemTab challenge. Improvements to the existing KG generation method given in the SemTab 2019 are also proposed.

1.1 Document's Organisation

The document is divided into two parts, one for the introduction and one for the proposal to solve the issue of Tabular Data to Knowledge Graph Matching. The conclusion is then written, which is not a definitive conclusion as there were no experiments conducted over any dataset to validate the proposal

2 Proposal

2.1 On tabular data to Knowledge Graph generation

When a tabular data is presented, we face multiple problems such as the following:

– The relationship between the columns.
– Lack of semantic meaning of the content.
– Lack of Metadata.

As we can have heterogenous data from different sources, one improvement can be find the type of the column and arrange columns with similar types together. This will result in a generalized structure useful when we will be coding the KG. To counter the problem of lack of metadata while processing the data from tables. One approach could be to search for the the item on wikidata, DBpedia and if it exists, using the data entered there to add into the KG. To counter the problem of noise in the data, we can use levenshtein distance to find the closest word, if there is a misspelling. We can also use a word similarity index calculator with preexisting language models like BERT. If the KG to generate is a in a specific domain, and we have a masked language model for that domain, we can use the masked language model to find the closest word (but it is important to show a relation that this entity was not originally present but is generated, for example can use a vocabulary containing 'was previously' relation or anything semantically similar). Domain experts in the knowledge graph can identify the mistakes and correct them. We as computer scientists, can not correct. We can make a relation to make it easily queryable with SPARQL by adding a blank node []. (e.g ex:Item1 ex:needsReview []) There can be entries with short labels (like abbreviations), which can be corrected either with domain experts/maintainer of the KG or by voting among the users of the KG or by checking if there is an alias similar on WikiData or DBpedia.

Data Generation The paper uses only English DBpedia, instead of which one can use as many KGs we need to extract and generate data. We can add references to the data generated from a specific KG. We won't have to process as many RDF data dumps as before and more data will be generated as the goal is to have more meaningful data.

For generation of raw table data, we will write much more SPARQL queries for each dataset being used (e.g DBpedia and Wikidata). I am an advocate for decentralized systems and if knowledge on a specific dataset is compromised by let's say an acquisation by BigTech companies, the Knowledge should be free. Thus references with multiple datasets will be beneficial.

Instead of using a randomized function to choose the columns, we can run a query to check if there is a blank node. We can process the one with the least blank nodes first, because there can be a case where a specific entry in the column is the 'subject' for other columns. This will generate better meaningful data. If there are two similar values don't skip one, instead add the other value too in the KG. It can be useful for a question-answering system to check for the amount of data is there related to a property, providing more meaning to the data. The method to ensure diversity in the data is great in my opinion if we wish to get smaller dataset for the experiments.

RDF Data The table and rows will be an instance of OWL class. We have to make sure that we take in consideration that we might have data from multiple sources, which will help us scaling the KG later. We can employ the RML

Mapping Language for this, as it defines the source of a value injected and will provide more meaning to the data.

Regarding evaluation, I am not what is the best way to move forward because we will need a ground truth. This gets complicated when we inject information from multiple heterogenous data sources. Maybe, if we get multiple values from different datasets, the users decide and vote on the options. This data is then stored and used to train a machine learning model to decide which particular information source is more reliable/preferred by the users.

2.2 On CEA, CTA and CPA

Column Entity Association I think there are two ways one can iteratre over to find the column entity, one can be querying the specific label or second one can be to run a loop by using the row and column of each entity. Although some preprocessing is required, which as decribed in the previous section, could be similarity of words. Although there can be a problem when the specific entity is, let's say a sentence instead of a word.

If we go by rows and columns, we will have to make sure of the type of columns we are interacting with. For example, a column with property can not be in the end of our triple and likewise. The values we will get will be then added along with the subject they are meant to be with. *If we just go by the subject's name*, then we will not be sure if the column is of the same/required type. This will hence add a filtering process after the values extracted.

Column Type Association I think it is better to find the values for the entities related to column type association which are found through the column entity association. I think just finding a value of a datatype "Item" (just an example) is not enough. It does not provide a meaningful relation. If we search from the entity list we generated with CEA, we will be able to make a meaningful relation. If we search for the entities who have the datatype "Item" (for example) I think it would be better. (Not very sure, please correct me if I am wrong). Otherwise, we will have to make it filtered in the next stage, to check out the values with no entites related. To make the system more efficient, we can make the final list by either voting or counting how often the item exists in the table (to give a higher priority of selection (if we need let's say less amount of data))

Column Property Association As we have tried to make the data of same datatype in nearby columns, I think it will make the extraction a tad bit easier. We can select the type of property we wish to extract, for example if you wish to get date then we select the property related to date, and likewise for other datatype. This might help us to filter out the data which don't have a the required value linked with them. One can discard all the other values or maybe store for other usecases to check the accuracy of this approach.

3 Existing Approaches

The sections describe the different papers entered in the competition. Some general modifications which I would propose in every apporoach will be,

- Adding data from multiple sources, and being generalized enough to introduce a new source at any point of time.
- Querying the original KG after a specific time, to check and infer a new knowledge or a change in knowledge, if any.
- Providing a reasoning logic in generation, if the same label provides different values when we go through different KGs.
- If we are using a KG or several KGs, reasoning logic to define the ground truth is important if we wish to take feedback and enhance with machine learning and deep learning models.

Specific merits and demerits I found in the papers are disucssed in the subsections,

3.1 ADOG : Anotating Data with Ontologies and Graphs

Merits

- Unfortunately, I would not add anything to the system I would make if I had to make, from this paper.

Demerits

- I don't think this is a good approach, because we would like the knowledge graph to be generated from the CSV
- I would like to find a method to find relations between the columns automatically than predefining the schema by myself.
- They also said to make the KG not dependant on one single source, but they are only using DBPedia now.
- It would be interesting if we could see a multi-source system.
- Very minimal preprocessing of the data.
- No discussion if the property/new column type entered or present in the table is not present already in the defined and loaded ontology file.

3.2 CSV2KG : Transforming Tabular Data into Semantic Knowledge

Merits

- The column inference heuristic logic with majority voting employing the hierarchy of DBPedia ontology
- Choosing voting as well as counting/preferring the majority of the results as the final decision.
- Employing reasoning to find equivalent classes and column types.

Demerits

- The column type annotation is not considering irregular data, for example if in a column (of 20 entries) you have 15 with one context and 5 irregular, or out of context data. You need to set a threshold, by finding the datatype of every entity, and with threshold compute the actual type of the column. If possible, remove the outliers to a different column and add them with a different type into the KG.
- As everything happens step by step, and the steps are dependant on one another, if one fails the KG generated will not be good. Thus, it can be improved by making a check that if step 2 is executed fully and correctly we move towards step 3.

3.3 Entity Linking to Knowledge Graphs to Infer Column Types and Properties

Merits

- The candidate generation and selection method approach was very good.
- They have employed a neural network to learn weights and relatinships from labeled data to rank the entities. (which will be interesting to see how we can work with neural networks in a context-free table to KG generation, as in most of the tables we are not sure if there will be entites on wikidata or if there will be a metadata to work with. If a neural network is trained, it can classify the entities and columns with properties and classes. I am not sure if this approach has been implemented already.)
- The model ranking system in the entity is also employed for deciding the column property and column type.
- They have targeted the two major KGs, DBPedia and Wikidata.

Demerits

- The ground truth (one for each KG i.e. one for wikidata and one for DBPedia and beyond) for training the ranking algorithm is not implemented.
- The entities who are discrarded in the candidate generation process are not considered beyond, maybe they were domain specific and not in WikiData/DBPedia.

A note that, if I had to build a system for KG generation from context-free tables, I will build upon this approach. Although, I am not very sure because there could be better papers, I did not do enough literature review!

3.4 MantisTable: An automatic approach for the Semantic Table Interpretation

The paper identifies the limit of the state of the art of being reliant on the metadata. The paper follows with, "MantisTable takes a well-formed and a normalized relational table" which is a bit misleading as if the current limitation is reliance on metadata, and then taking a regular normalized table to solve the problem is not very beneficial. Metadata is important to understand the structure. RDF is a very powerful tool for metadata, and it is very important to understand the structure of the metadata. I am not sure how can we say that reliance on metadata is wrong, if we have a context-free table metadata can actually help us to infer the structure of the table.

Merits

– The data preparation approach is good as it considers different units a value can have, rather than classifying them into the same table-type as 'units/values'. We will get more meaning out of the data.
– The column analysis is assigning the columns into named-entity, literal and subject column which is a good approach, but what if we have multiple subject columns with the same property and different object values. I am not sure how the existing approach will handle it.
– For the concept and datatype annotation is good, it will also be nice using the ranking based model for it would be even better as instead of counting the number of rows in the table it uses the ontology present at Wikidata/DBPedia. I am not sure which one would perform better.

Demerits

– The detection of a subject cell is dependant on the fraction of empty cells, average number of words in the cells, which is oddly specific and will not be scalable for tables out of the scope of the challenge.
– The approach in the first rounds was to consider the entities for which, within the values of *rdf:type*.
– They only identify one class per column.

3.5 MAGIC: Mining an Augmented Graph using INK, starting from a CSV

Merits

– The approach employing INK is very good, as it employs existing KGs to do entity, type and property annotation at the same time. I like that it concatenates the relations on a path from all the other nodes on the path. You can also use the relations from each path to do node classification if you wished to, I think this approach would help us if we find let's say information about the entities, and then with that information define a threshold to classify the column type or as we are able to identify the node, we can extract similar nodes and place them in a different column.
– It can also provide the additional information available with the KG it is infering from.

- INK technique can also query locally stored Knowledge Graphs.
- It takes notice that there could be a case where the 'main column' is not defined, and thus the method works accordingly.
- They are employing HDT to save space (my labmate is working with HDT in his PhD thesis!)
- MAGIC approach can be adapted to any other KG as well. (something that I would like it to have.)
-

Demerits

- Very few demerits, although if there was some reasoning logic employed in candidate selection, it would've been better.
- They did not declare what will happen for cases where the entity finds no match in either WikiData or DBPedia.
- Implementing WordNet and/or levenshtein distance will be great and improve the approach.

I would chose this paper too, to build upon a Tablular Data to KG matching platform.

3.6 DAGOBAH : An End-to-End Context-Free Tabular Data Semantic Annotation System

Merits

- Implements a method based upon primitive cell typing system to find out if the table is horizontal or vertical.
- They have implemented querying from 5 different services, than just one or two in other approaches. Although, they are changing the QIDs to DBPedia ones, which they prefer.
- The elasticsearch server implemented, is good to add the aliases, but they could have been added by running a SPARQL query with *skos:altLabel* as well.
- They have implemented K-means algorithm which is good to classify the wikidata embeddings, but I am not sure if it is a good/bad approach as they only tested with one dataset, the results could change with other datasets.

Demerits

- Only dependant on Wikidata, thus not scalable.
- The header extracter, assumes that the header will not share the type of the column involved, but I think it is bad for tables who just have string values embedded into them. The preprocessing method is rather specific to the data presented, and not very generalized.

3.7 MTab : Matching Tabular Data to Knowledge Graph using Probability Methods

Merits

- They implemented lookup services for multilingual data, and used a fasttext NLP model to predict the language of each entity.
- Approach to solve the problem of preference, when there are multiple KGs to lookup from is implemented, with entity candidate estimation which is good.
- It is considering both entity-entity and entity-non entity column relation.
-

Demerits

- The assumptions laid out before explaining the approach makes the system too case specific and not generalized as we would prefer in an automatic KG generation from tabular data format.
- They are not matching the entity, but due to one assumption, using reasoning to find a relation, which in my opinion is not great.
- Assumption to find the type of column, and then doing majority voting on it, which is good when there are different types of data.
- They claim that it is not solvable for a realtime application (which is mostly the usecase) which can be improved and adapted.

3.8 JenTab: A Toolkit for Semantic Table Annotations

Merits

- It is a very good thing to remove the cell entities which do not belong to the shared selected column type. It is a strong point of the paper that it is doing the CEA/CTA/CPA simultaneously and infering the knowledge from a process to improve another annotation.
- The cells which do not have a return value from a lookup into an existing KG are not left un-annotated or discarded but the are assigned the knowledge one infers from row and column contexts.
- System architectural benefits such as caching and different nodes to execute different services in a distributed fashion.
- They have a ground truth they build up the model upon.
- They employ SearX to lookup, which does lookup more than 80 engines.
- The preprocessing employed is good to eliminate noise, if there is introduced in the tabular data. The lexographic challenges are also handled when looking up entities in a preexisting KG.
- They are employing majority voting in Column Property and Type annotation.
- The classification of object, date, string and quantity as shown in the confusion matrix can be employed to train a model to improve the classification and annotation's accuracy further.

Demerits

– The suppport different date-time format, which is good but can be troublesome while querying a generated KG from the tabular data. In my opinion, all time should be preprocessed and converted to Unix timestamp and after that one can convert in the postlookup by a SPARQL query.
– String values which have an overlap of atleast 50 % are considered a match, but I think employing levenshtein distance is a better approach. They have employed levenshtein distance in the selection of an annotation, but I think it would be better in the creation of the entities.