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 a 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 it's 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 is 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 sentance 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 Conclusion

A proposal for generation of knowledge graph from tabular data is proposed along with methods for matching. The proposal can not have a definitive conclusion to application as there were no experiments conducted in this case.