SAND: A Tool for Creating Semantic Descriptions of Tabular Sources

Binh Vu(✉) and Craig A. Knoblock

USC Information Sciences Institute, Marina del Rey, CA 90292, USA
{binhvu,knoblock}@isi.edu

Abstract. Building semantic descriptions of tables is a vital step in data integration. However, this task is expensive and time-consuming as users often need to examine the table data, its metadata, and ontologies to find the most appropriate description. In this paper, we present SAND, a tool for creating semantic descriptions semi-automatically. SAND makes it easy to integrate with semantic modeling systems to predict or suggest semantic descriptions to the users, as well as to use different knowledge graphs (KGs). Besides its modeling capabilities, SAND is equipped with browsing/querying tools to enable users to explore data in the table and discover how it is often modeled in KGs.

Keywords: Semantic descriptions · Semantic models · Knowledge graph · Semantic web · Linked data · Ontology

1 Introduction

There is a large number of tables available on the Web and Open Data Portals. However, these tables come with various formats and vocabularies describing their content, thus making it difficult to use them. To address this problem, a semantic description of a table is created that encodes column types, relationships between columns, and additional context values needed to interpret the table. Given the semantic description, the table can be automatically converted to linked data or RDF triples providing the ability to quickly combine multiple tables for downstream tasks/applications to consume data using the same representation. Nevertheless, creating semantic descriptions is time-consuming. They are difficult to write manually, and during the creation process, the developers often need to switch back and forth between the ontology and the data to find the appropriate terms. Therefore, a user interface (UI) to assist and guide the developers in creating the descriptions is needed.

There are several UIs developed for creating semantic descriptions. Notable examples include Karma [2] and MantisTable [1]. Karma is a data integration tool that can ingest data from various types of sources (e.g., spreadsheets, databases, etc.), map them to an ontology that users choose, and export the normalized data to RDF or store it in a database. MantisTable can import tables

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
P. Groth et al. (Eds.): ESWC 2022 Satellite Events, LNCS 13384, pp. 63–67, 2022.
https://doi.org/10.1007/978-3-031-11609-4_12

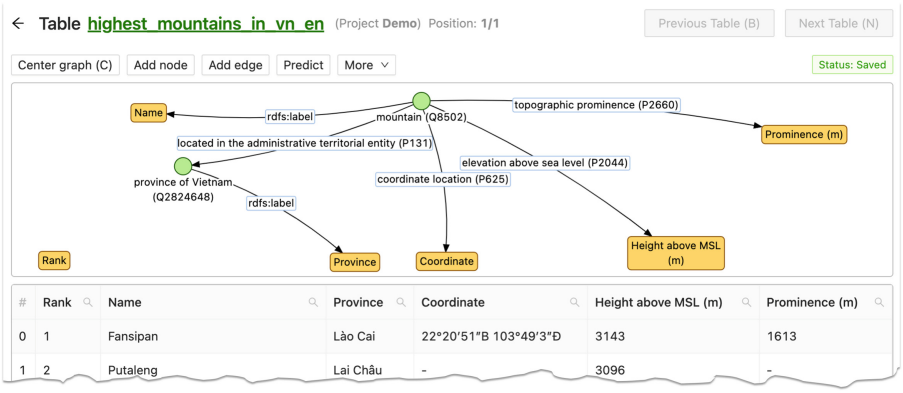

Fig. 1. The semantic description of the table about mountains shown as a graph on top of the table. Green nodes are ontology classes, yellow nodes are columns. Edges depicts the types (e.g., `rdfs:label`) and relationships of columns (e.g., `located in administrative teritorial entity (P131)`). (Color figure online)

from files and create semantic descriptions to map them to KGs such as Wikidata. In both cases, these systems are strongly integrated with their semantic modeling algorithms for suggesting candidate descriptions and hence difficult to extend with state-of-the-art algorithms. In addition, when mapping tables to KGs, we often need to explore the data through filtering and browsing entities to find rows that do not match the semantic descriptions, yet such features are lacking in the aforementioned systems.

In this paper, we introduce SAND a novel system that addresses the above limitations. The contributions of SAND are: (1) a UI for creating semantic descriptions with a full pipeline from raw table data to RDF/JSON-LD output; (2) browsing/querying features for exploring the data and how it is modeled in KGs; and (3) an open design enabling easy integration with semantic modeling algorithms for semi-automatically creating semantic descriptions while supporting different knowledge graphs (e.g., Wikidata, DBPedia). SAND is available as an open source tool under the MIT License[1].

2 Creating Semantic Descriptions in SAND

In this section, we will show how a user can build the semantic description of a table of mountains using the Wikidata ontology and then export the data as RDF. First, we discuss the overall process as if the user performs all steps manually. Then, we demonstrate how the user can do it semi-automatically.[2]

Manual Process. The process begins by uploading tables to SAND. SAND supports reading tables from various formats such as `.json`, `.csv`, `.xlsx`, and

[1] https://github.com/usc-isi-i2/sand.
[2] Demo: https://github.com/usc-isi-i2/sand/wiki/Demo.

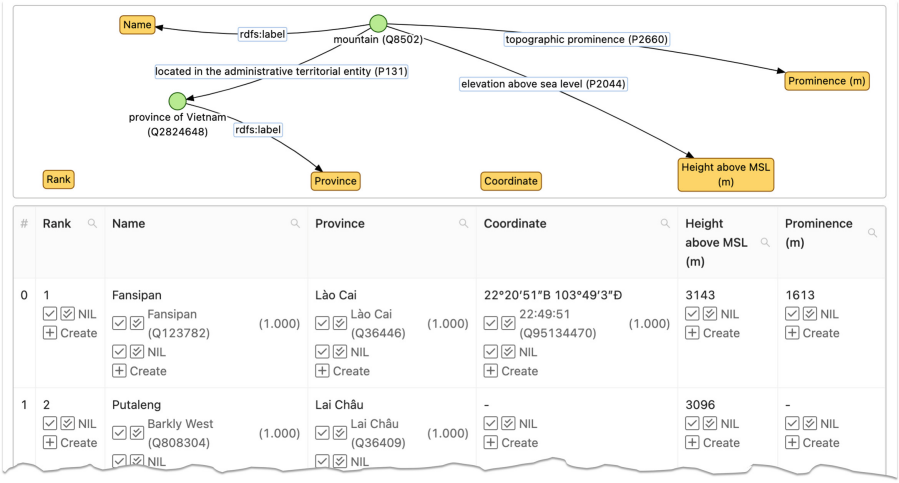

Fig. 2. The auto-generated semantic description and linked entities of the table. If a cell is linked to an entity, it is shown as a hyperlink and its candidate entities are displayed below it. The user can update the linked entity of a cell by clicking on a single-check or a double-check button. Differing from the single-check button, the double-check button will apply the operation to all cells (same column) that have the same value as the current cell.

allows the user to refine the parsing options for different formats. The next step is to create the semantic description of the uploaded table. In particular, the user assigns semantic types to columns containing entities (called entity columns) and relationships (ontology predicates) between the entity columns and the remaining columns. Figure 1 shows the final outcome of this step. For example, the 2nd column *Name* is assigned to type `mountain (Q8502)` (via the link `mountain...` → `rdfs:label`) and the 3rd column *Province* is assigned to type `province of Vietnam (Q2824648)`. The description also tells us that these mountains are located in the provinces by the relationship `located in ... (P131)` between the two nodes: `mountain...` and `province...`. The third step is to link cells with existing entities in KGs (e.g., Wikidata). This step is optional, but we recommend doing it in SAND as during exporting, any cell in the entity columns that does not link to any entity is treated as a new entity. In this table, many entities are already in Wikidata so by doing entity linking, we can avoid creating duplicated data. Finally, once the user is satisfied with the semantic description, they can export and download the RDF of the table. Note that in the RDF data, raw values of the coordinates and heights are converted to the correct types thanks to the value constraints provided by the semantic description. Exporting the data via the UI may not be ideal if the user has a huge table or many tables of the same schema. Therefore, SAND also allows exporting an internal D-REPR specification of the table, which is used to transform the table to RDF by the D-REPR engine [5] programmatically.

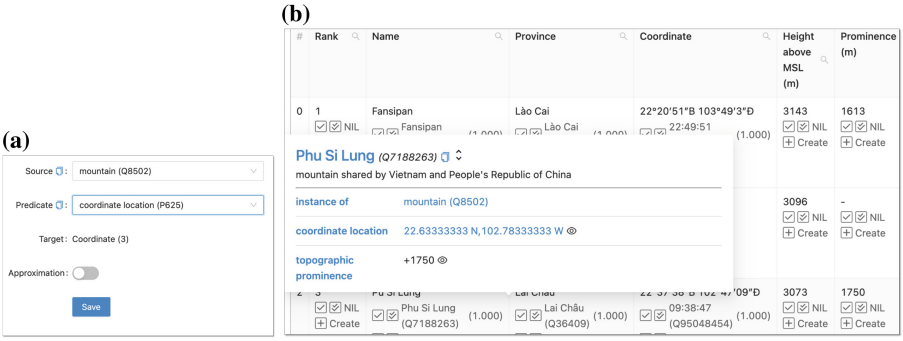

Fig. 3. (a) Add/Update relationship dialog box (b) Entity browsing

Semi-automatic Process. As the whole process is quite lengthy, hence at the beginning, the user can tell SAND to generate the semantic description and link entities for them by clicking the predict button. Under the hood, SAND will invoke semantic modeling and entity linking algorithms that are integrated to SAND via its plugin systems. Currently, SAND provides plugins for MTab [3] and GRAMS [4]. Figure 2 shows the result after prediction. Notice that it cannot predict the correct relationship of the 3rd column and some of the linked entities are incorrect. The user can fix the missing relationship by clicking on the yellow node representing the 3rd column, and selecting the source node and the relationship as shown in Fig. 3a. Another way to add the missing relationship is to click on the `add edge` button. To fix the incorrectly linked entities, the user can deselect them or right-click on the column and choose a filter to deselect cells containing entities that do not belong to the current semantic type (`mountain`) all at once.

Exploring the Data. Assuming that the last column in the table is ambiguous and the user is unsure if the predicted relationship `topographic prominence (P2660)` is correct, they can choose to filter rows that have entities having the property P2660 and use SAND's entity browsing capability to check if any values of the properties are different from the values in the table (Fig. 3b).

3 System Design and Configurations

To integrate with KGs, we define common schemas for entities, ontology classes, and ontology predicates. Then, we create their data access objects (DAOs), each of which has two functions: (1) querying an object by its ID or URI; and (2) searching the objects by their label. The DAOs also convert objects from their original schema in the KG to the common schema in SAND. This approach makes it easy to set up SAND since DAOs can be implemented using KG's public APIs. In other words, we do not require to build a local database containing KG's entities and ontology.

To integrate with a semantic modeling algorithm, we declare an abstract class that predicts the semantic description of a given table and links table cells to entities in KGs.

The DAOs and semantic modeling algorithms are specified in SAND's configuration file by their paths (e.g., `plugins.wikidata.get_entity_dao`) and imported dynamically when the server starts.

4 Discussions and Future Work

With SAND, users can quickly model and convert their tables to RDF or JSON-LD for integrating with their dataset or knowledge graph. SAND reduces the amount of work that the users need to do via its semi-automatic semantic modeling, entity linking, and data cleaning features. For the researchers or developers, customizing SAND's components is straightforward via its plugin system. They can implement and test their semantic modeling algorithm on SAND or adapt SAND to their use cases without having to develop a new UI from scratch.

For future work, we plan to add more data source types such as databases and APIs. We also plan to support writing custom data cleaning scripts and incorporating data cleaning algorithms into the system.

References

1. Cremaschi, M., Rula, A., Siano, A., De Paoli, F.: MantisTable: a tool for creating semantic annotations on tabular data. In: Hitzler, P., et al. (eds.) ESWC 2019. LNCS, vol. 11762, pp. 18–23. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32327-1_4
2. Gupta, S., Szekely, P., Knoblock, C.A., Goel, A., Taheriyan, M., Muslea, M.: Karma: a system for mapping structured sources into the semantic web. In: Simperl, E., et al. (eds.) ESWC 2012. LNCS, vol. 7540, pp. 430–434. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46641-4_40
3. Nguyen, P., Yamada, I., Kertkeidkachorn, N., Ichise, R., Takeda, H.: Demonstration of MTab: tabular data annotation with knowledge graphs (2021)
4. Vu, B., Knoblock, C.A., Szekely, P., Pham, M., Pujara, J.: A graph-based approach for inferring semantic descriptions of Wikipedia tables. In: Hotho, A., et al. (eds.) ISWC 2021. LNCS, vol. 12922, pp. 304–320. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88361-4_18
5. Vu, B., Pujara, J., Knoblock, C.A.: D-REPR: a language for describing and mapping diversely-structured data sources to RDF. In: Proceedings of the 10th International Conference on Knowledge Capture, pp. 189–196 (2019)