Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 137 (2018) 249–254

Procedia Computer Science

www.elsevier.com/locate/procedia

SEMANTiCS 2018 – 14th International Conference on Semantic Systems

A Web-based UI to Enable Semantic Modeling for Everyone

André Pomp*, Alexander Paulus, Daniel Klischies, Christian Schwier, Tobias Meisen

Institute of Information Management in Mechanical Engineering, RWTH Aachen University, Aachen, Germany

Abstract

Since companies generate and store large amounts of data daily in centralized systems such as data lakes, understanding data sets from different sources is becoming an increasingly complex task in dealing with data heterogeneity across domains. One solution for describing semantics of data sources is the use of semantic models based on an available vocabulary. However, creating detailed semantic models can be a challenging task for users who are not familiar with semantic modeling and today's available tools.

To overcome this challenge, we developed an intuitive and user-friendly interface, allowing data owners to define detailed semantic models for their data sources. The design of the user interface is based on an intensive requirement analysis gathered among several peers. It provides an intuitive mapping of semantic concepts to data attributes and the definition of relations between those concepts using drag and drop interaction. The user is given full modeling freedom as the insertion of semantic concepts and relations that are missing in the underlying vocabulary can be done on-demand and does not delay or impair the modeling process. Additionally, the refinement of the original detected data schema is supported with several operations. We built the interface into the semantic data platform ESKAPE, which already uses a flexible knowledge graph as underlying vocabulary and provides a detailed analysis of the data schema.

Keywords: Semantic Modeling; Knowledge Graph; User Interface

1. Introduction

In the age of the Internet of Production, which focuses on the challenges to guarantee the availability of real-time information at any time and place, large amounts of data are generated, collected and stored in industrial settings every second. The goal is to utilize the collected data for gaining additional insights or for developing new applications. To store all these structured and unstructured data sources at a central location, companies start to use data lakes, which follow a schema on read approach. However, data lakes have the disadvantage of accessing, finding and understanding all added data sources. To support accessibility, searchability and comprehensibility of data sources for

* Corresponding author.
E-mail address: andre.pomp@ima.rwth-aachen.de

10.1016/j.procs.2018.09.024

persons who are not familiar with the data schema and labels, different meta-data management solutions exist. One solution, derived from the efforts of the Semantic Web, is the use of semantic models. These models allow to specify the meaning of data columns and their relationships by annotating the columns with semantic concepts obtained from an underlying vocabulary, such as an ontology, which is created by domain experts and ontology engineers in advance. This annotation process defines the data set's semantics explicitly and makes them available to other users. However, defining semantic models or the underlying ontologies is a cumbersome task for domain experts as they have to deal with the formalism coming with solutions like RDF or OWL. Therefore, it is still necessary to involve ontology engineers in this process as they are familiar with this formalism and can support the domain experts by either creating their semantic models or the required underlying vocabulary.

Our research goal is to overcome the issue of involving ontology engineers in the process of semantic modeling by providing an easy to use and intuitive user interface for supporting the domain experts. This interface hides the required formalism and simplifies the modeling process. Simplified modeling is achieved by intuitive drag and drop interactions with which a user can map semantic concepts to data attributes. Although modeling semantic entities is the main focus of the interface, additional challenges have to be faced, as users are dealing with different forms of potentially inhomogeneous data sets during their work. Thus, the interface needs to be able to also process and transform the raw data values or react to violations of schemas. Examples include the cleansing (i.e., removing irrelevant parts from the actual values) or splitting of a data attribute into multiple attributes. All these actions result in the definition of more detailed semantic models.

We implemented our approach based on the semantic data platform ESKAPE [1, 2]. This platform enables users to define semantic models for their data sets based on an underlying knowledge graph. ESKAPE aims at a flexible definition of semantic models by allowing users to add new semantic concepts and relations on-demand, which then expand the underlying knowledge graph.

2. Related Work

Due to the rising number of available data sources, the research effort invested in developing methods for finding and understanding these sources have been increasing as well. We can differentiate between approaches that rely on a fixed vocabulary, such as ontologies, and approaches where new tags can be added on-demand without defining a vocabulary in advance. Both approaches might be used to annotate structured or unstructured data.

In the area of annotating unstructured data, different approaches exist for defining tags and adding those to data sources, such as images or audio files. For instance, Semdrops [3] enables users to tag websites. Each user defines his own set of available tags which decreases the necessity to keep tags consistent between users. By contrast, the Semantic Knowledge Management Tool (SKMT) [4] tags documents based on an underlying ontology. Both solutions are easy to use since they do not constrict the user with additional formalism.

For annotating structured data, there exists a research project named KARMA [5], which allows users to define semantic models for their data sets based on a pre-defined ontology. The UI supports the user in creating semantic models by suggesting semantic types and relations. However, the user's semantic models are limited to the vocabulary provided by the underlying ontology. Another disadvantage of this approach is that the user, who is creating the semantic models, must be qualified to deal with the syntax and properties coming with OWL and RDF.

This disadvantage also holds for all the tools that are required for creating the necessary ontologies that may be used during the process of creating semantic tags or models. Tools like Protégé/WebProtégé [6], TurtleEditor [7] and many others enable the design and development of fully-fledged ontologies. The authors of [8] give an overview of the multitude of existing ontology modeling tools. Other solutions that are not listed there, like the Aspect-Oriented Visual Ontology Editor [9], try to simplify the modeling process with the help of aspect-oriented programming and an editable visual language. Their user interface already hides a lot of formalism from the user, but basic knowledge in OWL and RDF syntax is still required.

We notice that modeling processes based on simple tagging systems enable users to create semantic annotations without dealing with the underlying formalism. However, with an increasing community and a lacking moderation, these systems may become unstable. In contrast to these approaches, semantic modeling based on pre-defined ontologies is more powerful and does not suffer from the issue of reaching an inconsistent state. Here, it is especially valuable if users do not build a model from scratch but start from a given set. However, the presented approaches

Listing 1: Example input data set containing a single data point holding information about a flight.

```
{
  "id": "AL0025-20200123-1456-1634",
  "o": {"name": "Cologne", "shorthand": "CGN"},
  "d": {"name": "Berlin", "shorthand": "BER"},
  "staff": [
    {"name": "Pilot 1", "seat": "P1"},
    {"name": "Pilot 2", "seat": "P2"},
    {"name": "Flight attendant 1", "seat": "FA1"},
    {"name": "Flight attendant 2", "seat": "FA2"}
  ]
}
```

still require users to deal with the formalism associated with OWL and RDF. This is especially important for users when they have to create the required ontology beforehand. A system which tackles these issues is the semantic data platform ESKAPE [1, 2]. ESKAPE allows data stewards to semantically integrate, process and query data sources. For that, users define semantic models upon their data sets. The created semantic models are developed on the basis of a continuously evolving knowledge graph, which is maintained by ESKAPE. If a user lacks concepts during the semantic model creation, they can be added on-demand, resulting in a higher flexibility during the modeling process.

3. The Visual Modeling Interface

Our goal is to combine the benefits of already existing ontology engineering tools with the simplicity used by current tagging approaches presented in Section 2. As the semantic data platform ESKAPE already provides its users the flexibility to introduce new concepts, our goals are to develop a user interface which supports the user during the modeling process, to simplify the modeling process as much as possible and to stick to usage patterns commonly known even to non-technical users. Keeping this in mind, we build a web based modeling GUI to provide the user with a convenient modeling environment. Since our work is based on the semantic data platform ESKAPE [1, 2], we stick to the terms and definitions that are used by ESKAPE. For more information about the process of building and storing models, we refer to [1]. In the following, we describe how we visualize the data schema that is detected by ESKAPE and how we enable users to create their semantic model for the detected schema. As semantic models may become more accurate when the data schema is modified, we describe how such modifications are supported with the help of our UI. A demonstration of creating a semantic model based on our developed UI is available online[1].

3.1. Data Schema Visualization

When designing an interface that is supposed to behave intuitively and also to be usable by untrained users, developers usually have to stick to usage patterns which are already known to the user base. For starters, our interface therefore displays the detected data model schema and indicates for each attribute whether it is a composite, such as a list, or constitutes a conflict as it contains different types of data throughout the data set (cf. Figure 1). The original data structure, as it was detected during the prior analysis phase, is displayed as a hierarchical or flat model, depending on the data's input format. Each node is labeled by its original identifier taken from the data set's header or structure. List and object nodes have special symbols and hold references to their respective child elements. In case of lists, where the children are unnamed, a placeholder ('W:') is used. Leaf nodes, i.e., the nodes that have actual values assigned to them, are called *primitive values* as they do not contain any sorts of sub-elements or hierarchy but plain content. These nodes are labeled with a symbol that represents their detected data type or a question mark if the system could not detect one. In both cases, the data type of this node can be adjusted manually. The data type indicates how the values represented by this label are supposed to be interpreted (e.g., as a number, Boolean, binary, etc.).

[1] https://youtu.be/tmeFWiFZgUs

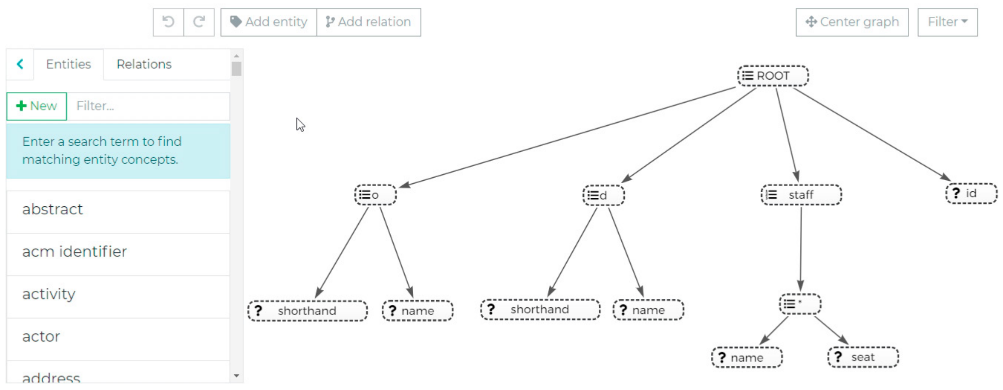


Fig. 1: Modeling GUI at the start of the modeling process. The analyzed data schema is displayed to the user. In this case, it is a hierarchical structure extracted from the JSON file shown in Listing 1.

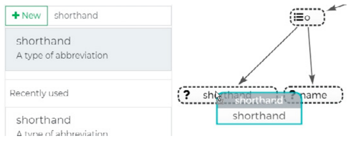
3.2. Building the Semantic Model

To add an entity to the semantic model, a semantic concept has to be selected by the user. Using the left sidebar, filtering, creation and selection of available Entity Concepts is possible. Once a suitable concept has been found, the user can drag and drop the concept on the canvas. Placing the concept on a leaf node combines both and creates an Entity Type to hold the customized name, data type and Entity Concept. If an Entity Concept is not placed on a data attribute, the resulting Entity Type will just represent that concept (without any binding to a data node) and extend the semantic model with additional concepts resembling meta information. Figure 2a shows the process of dragging and dropping a concept to place it onto a leaf node or into free space. In case it has been assigned to a leaf node, Figure 2b shows the resulting view after the combination has been performed. The Entity Type replaces the primitive leaf node from that point on.

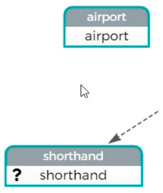
This Entity Type can then be modified, e.g., renamed, and in case it was bound to a data attribute, have a data type specified. Our visualization of Entity Types provides all immediately relevant information during the mapping process at a glance. Additional information can be obtained from the context menu, such as example data values for leaf nodes and original data labels (if overshadowed by an Entity Type, see Figure 2b). We chose to combine Entity Type and Entity Concept to a single visual element for the modeling process to increase readability and overview of the whole semantic model. In our view, the Entity Concept is always shown as an additional field above the original Entity Type label. While the Entity Concept, which is assigned to a leaf node via an Entity Type, can be exchanged throughout the modeling process, it is not possible to modify the label or description of any Entity Concept. This is because Entity Concepts represent a global understanding of subjects, rather than the specific incarnation of these subjects in the current data set, and can therefore not be modified on a per data set basis.

All Entity Types that have been added to the model can be connected using Relations, which leads to a higher amount of modeled information. Relations form an integral part of the semantic model as they allow to provide information on how Entity Types, and therefore data attributes, are related to each other. Following Relations, analysts can find connected data attributes and later process them accordingly. Adding Relations to the semantic model is done similarly to the creation of Entity Types. After an appropriate Relation has been selected in the left-side menu, dragging a connection from one Entity Type to another (Figure 3a) will create a Relation of the selected type between them (Figure 3b).

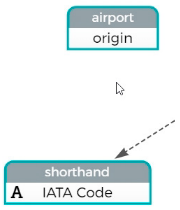
Should the user decide to use a concept or relation that is not already defined in the knowledge base, the GUI supports the addition of new concepts and relations to the model. Users just have to provide a matching name and a short description for the entity to be available in their semantic model. In order to check the models for consistency before they are processed by the backend during the data integration, a validation feature has been added to the GUI. It performs a series of checks during the modeling time, e.g., validates that all leaf nodes have been properly modeled. Using the described simple interactions enables users, with or without technical background, to annotate their data by building a baseline semantic model.



(a) Semantic concepts are placed using a drag and drop principle.

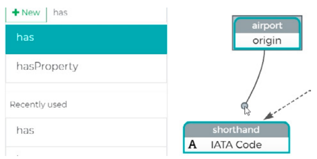


(b) Two Entity Types of which one is attached to a data attribute.

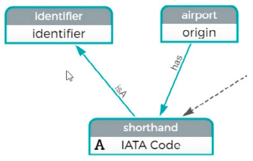


(c) Users can set the data type and rename the Entity Type.

Fig. 2: Assigning and configuring Entity Types.



(a) Example of a Relation. For adding Relations, the user just marks the Relation and drags it between two Entity Types.



(b) A relation's meaning is defined by its label, description and properties.

Fig. 3: Relating Entity Types in the semantic model.

3.3. Modifying the Data Schema

When creating semantic models, the usual goal is to cover all leaf nodes of the corresponding data schema as ESKAPE will integrate all these modeled entities later on. However, there might occur situations where users do not want to integrate specific information from the original data set. In this case, we offer users the possibility to not add a representing Entity Type to the data schema's leaf nodes. This forces ESKAPE to skip those during the integration.

In addition, ESKAPE offers the functionality to detect conflicts in the underlying data schema during a schema analysis step (cf. [1, 2]). These conflicts refer to inconsistencies among the same attribute. In such cases, it might not be possible to assign a uniformly matching semantic concept to this single attribute as it contains different syntactic elements among different data points. For instance, the value for the same key within a JSON document may switch between a JSON primitive value and a JSON object. A similar problem manifests when a list, e.g., in a JSON-coded data set, contains multiple elements of different types, but needs to be homogeneous to be annotated with a specific semantic concept. Before the data can be semantically integrated, the user has to resolve these conflicts during modeling process. This ensures that the model matches the stored data. To deal with inconsistencies in the data, we add data schema manipulation operations to the UI. For JSON keys with different JSON elements (e.g., primitives or objects that can be found under the same key), the user has to decide whether he wants to model the primitive values or the object values. For lists, we support splitting based on a repeating pattern. This enables us to filter out unwanted and conflicting elements. The splitting process is based on a pattern entered by the user when selecting a conflicting data point. Example values, on which the system determined the conflict, are shown and the user can enter operational parameters.

Besides the requirement of resolving conflicts, many primitive data attributes may contain multiple semantic information. An example data value is *(52.32131;4.83135)* where the first part represents a *latitude* and the last part a *longitude*. The combined value makes a fitting semantic annotation difficult, as usually those concepts should be described separately. Hence, we offer the splitting of primitive attributes into multiple independent ones. This enables the user to create more detailed semantic models. Splitting is, similar to schema conflicts, done via Regular Expressions. The user can define multiple of those, where each pattern is applied exactly once to cut of one token from the start of the value. Thus, n patterns split the value into $n + 1$ tokens which become separate primitive nodes. For instance, a data attribute *flight id*, which contains values like 'AL0025-20200123-1456-1634', can be split into four parts by applying the hyphen slitting pattern three times. A single node $'flightid'$ is then split into four separate ones, which can now independently be annotated in the semantic model.

For this to work, the pattern has to be consistent in the specific attribute across all processed data points. Otherwise, the integration process, which will perform the actual split when the data is integrated into the system, will rule out mismatching elements and render them unusable later on. In case of batch data being integrated into the platform, this can be validated as all data points are already available. In case of streaming data, the algorithm must be capable of handling such occurring exceptions during the data integration. Adaption of splitting patterns, e.g., in case of a change in the data schema sent by the data emitter, is currently not possible but could be in the future. Combination of multiple primitive values to one object is not needed as the data is only represented by its semantic model, rendering the original data schema invalid after the integration took place.

4. Conclusion and Future Work

We presented a GUI to build semantic models for data sets to be integrated into the ESKAPE platform. The interface focuses on simple interactions with a minimum of formalities, e.g., by performing most of the operations using mouse interactions. Models are built using drag and drop principles, which is especially convenient for non-technical users. Furthermore, no syntax or modeling language has to be learned to use the modeling tool as all elements are visually represented. Due to the flexibility of the ESKAPE platform to support previously unknown concepts, the user has no constraints when modeling, as new concepts and relations can be added on-demand. To allow the creation of more detailed semantic models, we offer users the possibility to manipulate the data schema.

For future work, we will focus on improving the automation and the support that the interface can provide to users during the modeling process. More precise real-time checks are supposed to be in place in combination with suggestions on which semantic concept from the pool of available concepts to use, based on the label, data values, concept clusters in the evolving knowledge graph as well as already used concepts. This mitigates keyboard input when searching for matching concepts and allows users to simply select one of multiple suggested concepts which will be added to the semantic model. From the technical side, we would like to improve the support for bigger data models by providing selectional modeling with adaptive zooming, hiding elements not relevant to the current task and a stable auto alignment of nodes and relations. Improvements on, e.g., the splitting dialogs, are also planned to make those components more user-friendly. Beside these technical improvements, we focus on performing an exhaustive user study. The evaluation will compare different existing modeling tools, such as Protégé or Karma, with our developed user interface. Utilizing this evaluation and the feedback we get from people who are continuously using the platform, we hope to provide a tool that will allow every user to quickly create accurate semantic models.

References

[1] A. Pomp, A. Paulus, S. Jeschke, T. Meisen, ESKAPE: Platform for Enabling Semantics in the Continuously Evolving Internet of Things, in: [2017 IEEE 11th International Conference on Semantic Computing, ICSC, San Diego, Calif., USA], IEEE, Piscataway, NJ, 2017, pp. 262–263.

[2] A. Pomp, A. Paulus, S. Jeschke, T. Meisen, ESKAPE: Information Platform for Enabling Semantic Data Processing, in: Proceedings of the 19th International Conference on Enterprise Information, SCITEPRESS - Science and Technology Publications, 2017.

[3] D. Torres, A. Diaz, H. Skaf-Molli, P. Molli, Semdrops: A social semantic tagging approach for emerging semantic data, in: Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01, IEEE Computer Society, 2011, pp. 340–347.

[4] M. Kalender, J. Dang, SKMT: A Semantic Knowledge Management Tool for Content Tagging, Search and Management, in: 2012 Eighth International Conference on Semantics, Knowledge and Grids (SKG), IEEE, 2012, pp. 112–119.

[5] S. Gupta, P. Szekely, C. A. Knoblock, A. Goel, M. Taheriyan, M. Muslea, Karma: A system for mapping structured sources into the Semantic Web, in: Extended Semantic Web Conference, Springer, 2012, pp. 430–434.

[6] Stanford University, Protégé - open-source ontology editor (2018).
URL `https://protege.stanford.edu/`

[7] N. Petersen, A. Similea, C. Lange, S. Lohmann, TurtleEditor: A Web-Based RDF Editor to Support Distributed Ontology Development on Repository Hosting Platforms, International Journal of Semantic Computing 11 (03) (2017) 311–323.

[8] MKBergman, Sweet Compendium of Ontology Building Tools (2018).
URL `http://www.mkbergman.com/862/the-sweet-compendium-of-ontology-building-tools/`

[9] F. Hallay, S. Hartmann, N. Kewitz, R. Mertens, An aspect-oriented visual ontology editor with edit-time consistency checking, in: Semantic Computing (ICSC), 2017 IEEE 11th International Conference on, IEEE, 2017, pp. 297–304.