Using Node Embeddings to Generate Recommendations for Semantic Model Creation

Alexander Paulus, Andreas Burgdorf, Alina Stephan, André Pomp and Tobias Meisen
Chair of Technologies and Management of Digital Transformation,
University of Wuppertal, Wuppertal, Germany

Keywords: Enterprise Data Management, Knowledge Graphs, Semantic Modeling, Recommender Systems.

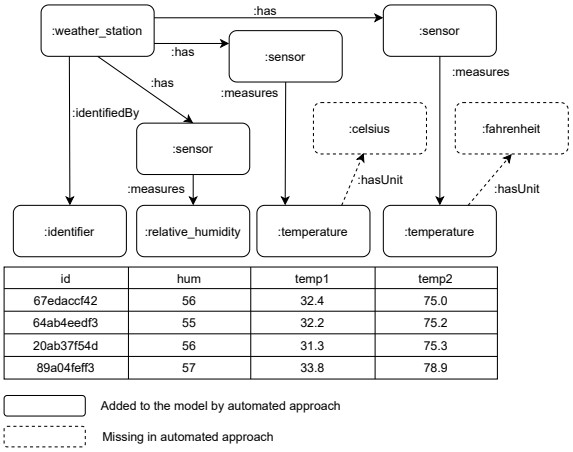
Abstract: With the ongoing digitalization and the resulting growth in digital heterogeneous data, it is becoming increasingly important for enterprises to manage and control this data. An approach that has established itself over the past years for managing heterogeneous data is the creation and use of knowledge graphs. However, creating a knowledge graph requires the generation of a semantic mapping in the form of a semantic model between datasets and a corresponding ontology. Even though the creation of semantic models can be partially automated nowadays, manual adjustments to the created models are often required, as otherwise no reliable results can be achieved in many real-world use cases. In order to support the user in the refinement of those automatically created models, we propose a content-based recommender system that, based on the present semantic model, automatically suggests concepts that reasonably complement or complete the present semantic model. The system utilizes node embeddings to extract semantic concepts from a set of existing semantic models and utilize these in the recommendation. We evaluate accuracy and usability of our approach by performing synthetic modeling steps upon selected datasets. Our results show that our recommendations are able to identify additional concepts to improve auto-generated semantic models.

1 INTRODUCTION

For many years, digital transformation has been changing a wide variety of business segments in enterprises, such as the fabrication of goods, the administration of organizational processes or even the communication with customers. One outstanding technology that is considered to have great potential for process optimization and the development of new business models is artificial intelligence, respectively data-driven machine learning. Today's AI-based systems rely on large amounts of data in order to provide meaningful and concise support. An approach for managing these heterogeneous data in a meaningful and efficient way is the use of knowledge graphs. Integrating data into a knowledge graph requires a mapping between the attributes of a dataset and an ontology that defines the entities of the knowledge graph. While the generation of mappings in the form of semantic labeling already enables a basic specification of the data, the construction of a more detailed mapping, known as semantic model, enables a supplementary description of the data, thus easing the process of finding, understanding and utilizing data (Vu et al., 2019; Futia et al., 2020; Knoblock et al., 2012; Paulus et al., 2021). Thereby, a semantic model extends the basic mapping by defining relations between mapped attributes, increasing the level of information associated with the dataset. However, creating a semantic model is often challenging as automated approaches can only cover parts of the process and mostly yield basic models. Hence, human interaction is often needed in a process called Semantic Refinement which corrects and expands the model according to the modelers needs.

In this paper, we focus on the task of providing part of a support system that proposes suitable recommendations for missing concepts or relations during the semantic refinement. For this purpose, we design a recommendation engine that serves as the core of this support system and considers implicit information expressed in historically (human-)created semantic models. On the basis of these, a prediction is made which additional concepts should be added to the current state of the semantic model. Providing suitable choices via a recommendation engine supports non-expert users by suggesting potentially unknown concepts to extend the model and expert users by pro-

Paulus, A., Burgdorf, A., Stephan, A., Pomp, A. and Meisen, T.
Using Node Embeddings to Generate Recommendations for Semantic Model Creation.
DOI: 10.5220/0011034900003179
In *Proceedings of the 24th International Conference on Enterprise Information Systems (ICEIS 2022) - Volume 1*, pages 699-708
ISBN: 978-989-758-569-2; ISSN: 2184-4992



id	hum	temp1	temp2
67edaccf42	56	32.4	75.0
64ab4eedf3	55	32.2	75.2
20ab37f54d	56	31.3	75.3
89a04feff3	57	33.8	78.9

Figure 1: Resulting semantic model for a given dataset. Element omitted by automated approaches are denoted as dashed.

viding suitable contexts faster. In order to generate recommendations, we use node embeddings to build a recommendation engine to recommend nodes that have a high proximity to the current semantic model to the modeler. We show that this approach reduces the effort and time required to manually create semantic models. To evaluate our approach, we perform synthetic modeling steps on different semantic models and estimate how many usable predictions the recommendation engine produces in this process (recall). Based on the results, we evaluate the number of recommended concepts that were added as context information.

In the further course of the paper, we formulate the problem in Section 2 and discuss previous related work in Section 3. Afterwards, we explain our recommendation system for supporting users during modeling in Section 4. The evaluation of the performance of the approach is given in Section 5. We conclude and give an outlook in Section 6.

2 PROBLEM STATEMENT

Figure 1 shows an example of a semantic model, which was created on the basis of the exemplified ontology illustrated in Figure 2. Mapping the concepts *:identifier*, *:relative_humidity* and *:temperature* to the data attributes *id*, *hum*, *temp1*, and *temp2*, respectively, creates an initial linkage between the ontology and the dataset. Afterwards, additional concepts and relations from the ontology should be used to describe the dataset, represented by its columns or labels, in more detail. Such provided meta information allows a semantic model to provide context in-

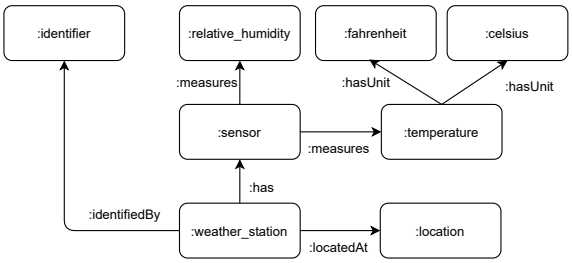


Figure 2: Example ontology to describe a set of sensors in a weather station.

formation of further specification on the relations between single labels. Following the given example, the specification of the unit of measurement *:fahrenheit* in the semantic model for the concept *:temperature* provides an essential added value that facilitates an unambiguous and correct description of the data. Additionally specifying data contained in a single column, like *:date* and *:time*, or *:latitude* and *:longitude* further improves the model's quality. Without those information, understanding and processing the data can be challenging as implicit meta information, like units of measurement or reference systems, are left unspecified. This will lead to inconsistencies when automatically merging datasets or integrating them into a knowledge graph. Context information in the semantic model can also be used to provide supplemental information, i.e., data that is not present in the original dataset. In our example ontology this could be the manufacturer of the sensors, the operator of the weather station or its location (assuming a fixed installation). While not explicitly needed, such information is valuable to people working with the data, like analysts or developers. We denote nodes in a semantic model that provide this additional information as *context nodes*.

In recent years, the focus of research has been on reducing the effort required to generate semantic models (cf. Section 3). However, there are no approaches that are able to create accurate semantic models, especially when the structure of the dataset or the target semantic model is complex (Futia et al., 2020). In addition, existing automated approaches are mostly limited to generate minimal semantic models by computing a minimal spanning tree for mapped attributes. (cf. Figure 1, solid relations). Context nodes such as *:fahrenheit*, that are present in the ontology shown in Figure 2, would not be added by such existing automated semantic modeling approaches.

It is therefore often required that a human user supervises the creation of semantic models and applies modification afterwards (Futia et al., 2020). This step, known as *Semantic Refinement*, describes user-validated modifications such as corrections as well

as extensions of the semantic model (Paulus. et al., 2021). While ontologies (like our example ontology or (Rijgersberg et al., 2013)) only provide choices for units of measurements, a modeler that is familiar with the corresponding dataset can with certainty decide which unit is fitting.

While semantic refinement does mitigate the shortcomings of automatic modeling approaches, new challenges arise once manual action is involved. First, the modeler may not know which information is meaningful and needs to be added to the semantic model. Second, the user may not know what possibilities exist in the underlying ontology to model the desired facts. Especially the second problem consumes much time when browsing potentially large ontologies for matching concepts and relations. There is also the possibility that certain facts may be modeled in different ways by different users. While automatic algorithms operate consistently, each human modeler tends to act and model differently. All of the above contribute to less consistent models that are more complex to process and understand.

3 RELATED WORK

Mapping heterogeneous datasets to conceptualizations is a research direction that has, so far, played an important role especially in the areas of Ontology-based Data Management (OBDM) and knowledge graph construction.*Semantic typing* cf. (Pham et al., 2016), (Abdelmageed and Schindler, 2020)) usually describes the process of mapping labels (e.g., table headers or leafs in a semi-structured hierarchical dataset) to an ontology. Here, different automated approaches exist that follow either a schema-driven (e.g., (Papapanagiotou et al., 2006), (Pinkel et al., 2017)) or a data-driven strategy (e.g., (Pham et al., 2016), (Rümmele et al., 2018), or (Pomp et al., 2020)).

Besides that, the creation of semantic models (also called *semantic relation inference* (Futia et al., 2020)) is also an important sub-field of semantic mapping. Apart from a direct mapping of the data attributes to concepts of the ontology, semantic models also include a specification of additional concepts as well as the relations that hold between the identified concepts. Approaches presented by (Taheriyan et al., 2016) and (Uña et al., 2018) have shown significant improvements in the area of semantic modeling during the last years. However, even the most recent approaches like (Vu et al., 2019) and (Futia et al., 2020) still lack full coverage, especially for complex modeling problems. In addition, the named approaches still only aim to connect the concepts created by semantic labeling using a minimal spanning tree (cf. Section 2.

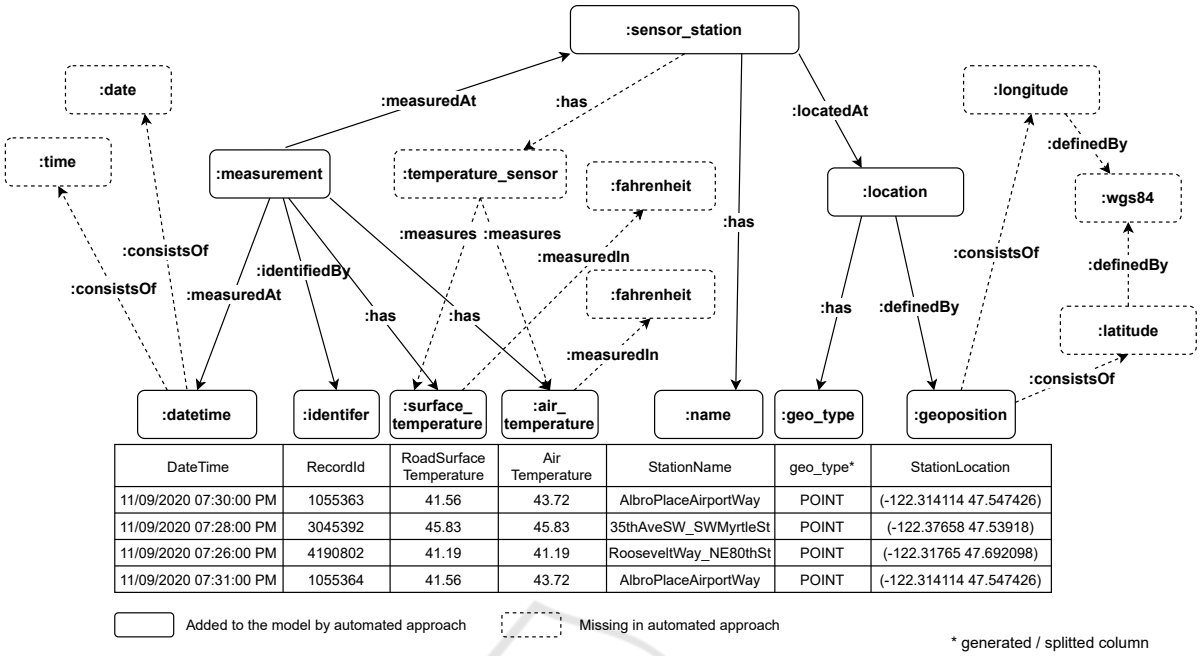
In order to overcome the shortcomings of automated approaches, it is important to rely on the integration of humans in a subsequent refinement process (cf. (Taheriyan et al., 2016), (Futia et al., 2020)). However, there are only few approaches that feature tools to inspect and modify a semantic model after the automated generation. Notable examples are Karma (Knoblock et al., 2012) and PLASMA (Paulus. et al., 2021). However, although following the human in the loop approach to finalize and refine models, all systems do not actively support the user during the modification phase.

Supporting the human user in the refinement process requires to derive additional implicitly modelled information from existing semantic models, which is often regarded as a problem of entity prediction or link prediction, i.e., derivation of facts that are not yet part of a model but could potentially increase its usefulness. (Baumgartner et al., 2021) proposed an entity prediction method based on textual descriptions. In (Futia et al., 2020), historic data is used to improve models but the problem of target node selection is solved in a brute force manner unsuited for most scenarios, especially interactive environments due to time constraints. Also, the resulting model is still minimal as it is computed via a Steiner Tree, disregarding context nodes.

While there has been an extensive usage of semantic information to build recommender systems for different domains (cf. (Codina and Ceccaroni, 2010), (Almonte et al., 2021)), using recommender systems to iteratively build semantic models is, to the best of our knowledge, a new approach. Existing approaches in this direction are suggestion generators for semantic labeling which let the user make a one-time decision which candidate to chose from a set of potential matches (cf. (Papapanagiotou et al., 2006), (Paulus et al., 2018)) or full ontologies and knowledge graphs (Saeedi et al., 2020).

4 MODEL EXTENSION RECOMMENDATION

While there are several approaches that improve the generation of initial (and minimal) models, the introduction of human modelers generates a need for different types of systems, namely those that improve the human modeling process. Recommendations help users to find suitable items of interest, as often observed in search engines or online warehouses. For example, recommender systems for online ware-



DateTime	RecordId	RoadSurface Temperature	Air Temperature	StationName	geo_type*	StationLocation
11/09/2020 07:30:00 PM	1055363	41.56	43.72	AlbroPlaceAirportWay	POINT	(-122.314114 47.547426)
11/09/2020 07:28:00 PM	3045392	45.83	45.83	35thAveSW_SWMyrtleSt	POINT	(-122.37658 47.53918)
11/09/2020 07:26:00 PM	4190802	41.19	41.19	RooseveltWay_NE80thSt	POINT	(-122.31765 47.692098)
11/09/2020 07:31:00 PM	1055364	41.56	43.72	AlbroPlaceAirportWay	POINT	(-122.314114 47.547426)

Figure 3: Semantic model for a dataset from the VC-SLAM corpus. Element omitted by automated approaches are denoted as dashed. 'geo_type' is a generated column which has been split from 'StationLocation'.

houses use information observed in previous shopping histories to generate advertising emails.

Recommendations for semantic concepts use a similar approach to generate suitable suggestions for the modeling user by exploiting previously observed models of different users. The engine's objective is to recommend concepts that are not added by fully-automated approaches but provide significant additional information to the model or selected parts of it. Figure 3 gives an example of a larger semantic model built for one of 100 datasets from the evaluation corpus (cf. Section 5.1). The models in this corpus was built using human modelers and does provide semantic models that are not limited to the minimal spanning tree but contain additional nodes to provide context. In the figure, solid lines indicate the minimal model, which has been generated automatically by a Steiner tree algorithm. The remaining items of interest are those denoted with dashed lines. We refer to the dashed nodes as *target concepts* as those are the model extensions our recommender should predict, as they are not covered by existing approaches.

Following the recommender system approach for products and items, our engine generates specific recommendations based on observed concept combinations. For example, imagine we have a set of concepts that comprise a semantic model to which the user adds the concept *:temperature*. If the concept *:temperature* has been observed in combination (not necessarily directly related) with the concept *:fahrenheit* in a majority and with *:celsius* in a minority of historic models, we expect the system to recommend adding *:fahrenheit* to the model and provide *:celsius* as an alternative.

We suggest a multi-step algorithm to identify and select target concepts. Figure 4 gives an overview of the recommendation process. Step 0 comprises the preparation and training and includes the extraction of relevant information from the historic data. Step 1-4 are the actual recommendation generation where a subset of nodes is selected to be used as a basis to generate the recommendations. In step 1, it is decided which concepts to use for recommendation generation, step 2 converts those to be fed into the node embedding. The results are then curated in step 3 before being filtered (for unwanted recommendations) in step 4. Afterwards, the resulting recommendation list is presented in an UI to be made available to the user (step 5). Last, elements are added to the model (step 6). As we focus on the recommendation generation during steps 0-4, we do not cover steps 5 and 6 in this paper.

In particular, we focus on two key aspects of support during manual semantic model creation: (1) reduction of modeling time and (2) provision of useful context information. We achieve a reduction of modeling time by eliminating the need for manual searches in ontologies. By offering a pre-selection of likely concepts, the user saves browsing time, thus speeding up the modeling process. Ideally, this also

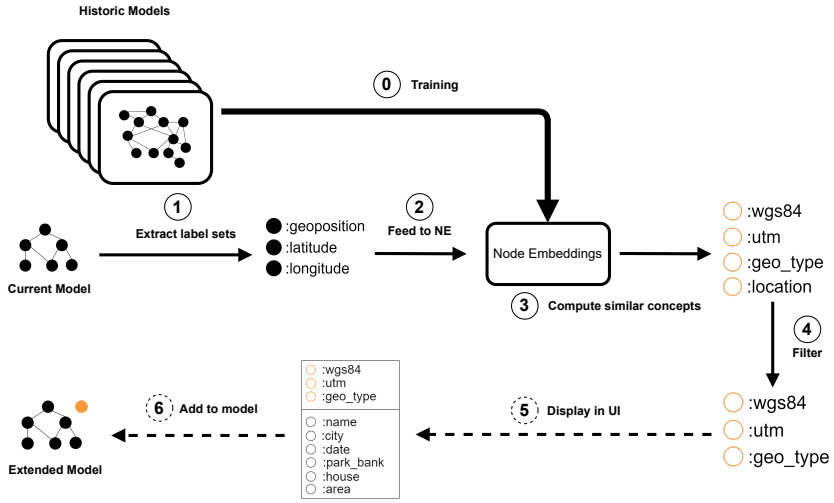


Figure 4: Overview of the model extension process. The proposed engine covers steps 1-4. Steps 5+6 are not realized yet.

saves the user the time to identify the next fact to add, e.g., *This geoposition consists of latitude and longitude*, but also, and more significantly, the time to identify proper concept(s) and/or relation(s), e.g., *:geoposition :consists_of :latitude*.

4.1 Embedding Generation

In order to generate model extensions, the engine needs to select a small number of concepts from all available concepts to recommend. Those target concepts are identified by means of node similarity computed using node embeddings. As training data, we use RDF triples contained in previously created semantic models. Those triples express facts in a (subject, predicate, object) form and are used to built ontologies, semantic models and knowledge graphs. We generate a directed weighted graph by incorporating those triples (s, p, o) and combine them to one graph where duplicate triples are merged to one edge with a higher weight. The weights are calculated as $\omega_{s,o} = \frac{|(s,?,o)|}{|(s,?,?)|}$ where $|(s,?,o)|$ denotes the number of times a matching triple $(s,?,o)$ was found in the training data ($|(s,?,?)|$ respectively). The final embeddings are generated using Node2Vec (Grover and Leskovec, 2016) with the weighted graph as input. As Node2Vec only works on homogeneous graphs, do not differentiate between relations p_1 and p_2 when building the weighted graph, i.e. we drop the edge label (URI) and treat all relations uniformly. In contrast to ontology based embeddings, this weighted graph approach exploits previously observed models to compute the most probable matching node combinations. Matching nodes are nodes similar to the given input node and are identified by computing the cosine distance (dot product) using the Node2Vec's underlying Word2Vec embeddings. Thus, in our approach, recommendations can only be generated for concepts observed in the training data. With additional available training data, the number of domains covered could be increased.

4.2 Model Extension Generation

In order to provide useful model extensions, the engine has to predict a concept and / or relations a user likely needs to add to increase usefulness and consistency of the model. We differentiate between two modes. In each mode, the selection of concepts to use as an input as well as the generation inside the engine differs.

The first mode is the extension of an existing model with the aim of adding useful supplementary information. This is similar to the customer visiting a warehouse site and being shown recommendations for general products the customer may want to purchase. Those recommendations can be based on the personal shopping history in general or items that are being bought by multiple customers frequently in the past. In the semantic modeling scenario, the goal is to make the user aware of missing meta information that could be added to the model. While it is challenging to find improvements that apply for the whole model, the direction of this mode is to first identify certain areas that could be extended with context information, and second to find suitable concepts to do so. As a basis for those extension recommendations the model-wide *general context* is used. The engine generates model extensions using all nodes of the whole semantic model as input. In Figure 5, the general context is indicated by a green border.

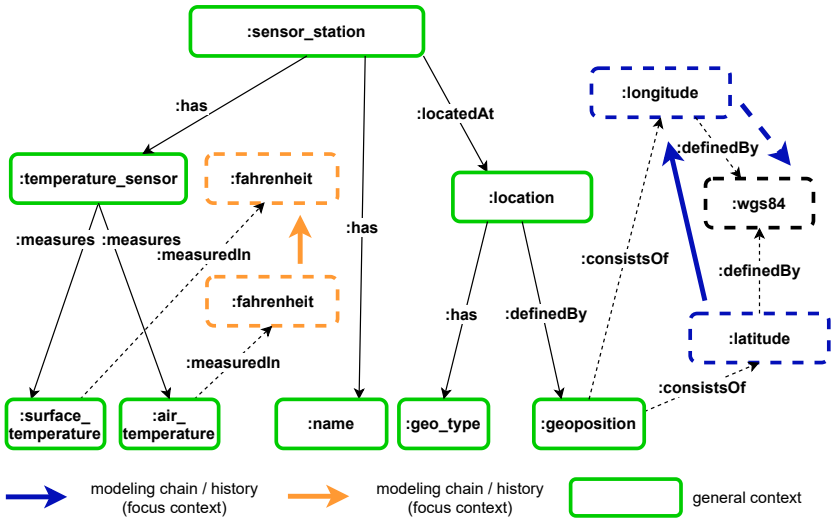


Figure 5: Different foci during semantic model creation on a reduced semantic model from Figure 3. Arrows indicate the order in which concepts could be added by the user and form different focus contexts. Elements in green indicate the general context.

Once the user begins to model, i.e., manually adds concepts to the graph, the mode changes. Now, the engine is able to recommend concepts based on a specific concept that has recently been added to the model, similar to recommendation engines that suggest accessories when a single item is bought online. In this second mode, the local *focus context* is used which consists of the recent modeling history, e,g,, the last three concepts added to the model, as an input to the engine, resulting in extensions that are more focused on the current part of the graph. The recommendations from the focus context can be used to quickly advance the model creation by recommending matching concepts in the current domain or subgraph, effectively working as an adaptive filter. The main aim in this mode is to reduce modeling time. Different focus contexts are shown in Figure 5 indicated by arrows. For the orange focus context, the user added *:fahrenheit* two times, resulting in a history of just [:fahrenheit]. For the blue context, the modeling order leads to a focus context of [:latitude, :longitude], with *:wgs84* being a probable output of the engine. Once the user would add *:wgs84* to the model, the history contains [:latitude, :longitude, :wgs84].

For the actual recommendation, the trained embeddings in the Node2Vec model are used as background data. For each set of input concepts, Node2Vec outputs a set of matching concepts and a score, indicating how certain the model is for each recommendation. In general context mode, the engine is set to generate five recommendations for each node of the input. For those nodes, the engine additionally considers the undirected neighborhood of the node (if it exists) to obtain more context information. Using Figure 5 as an example, for node *:location*, the input given to the embedding would be *:location*, *:sensor_station*, *:geo_type*, *:geoposition*. For each of the resulting sets of matching nodes, the average score is computed and the best three sets are selected. We build the union of those sets, while adding up the scores if the node occurred in two or more sets and use the resulting set as the recommendation output. In focus context mode, no additional information is added before generating the model's suggestions, as the history already defines a context. The resulting set of recommendations is directly used as output.

Once a set of recommendations for either of the two modes has been computed, the engine returns the whole set. The embedding system then has the choice of selecting a proper sample from those elements, e.g. top *n* elements based on the certainty score.

5 EVALUATION

In order to evaluate our approach, we tested the model extension generation engine on predicting new concepts for a given combination of concepts.

5.1 Data Selection

The main challenge of recommendation training is to find a suitable data corpus that contains models with meta concepts like units of measurement or reference systems. As most evaluation data corpora for semantic model creation aim for minimal models (Paulus

Table 1: Parameters used for the node embedding generation.

p	1
q	1
walk length	30
walks per node	500
window size	10
dimensions	100
negative sampling size	5
negative samling exponent	1.0

et al., 2021) which do not include the desired meta concepts, those are not usable in this context.

Thus, for the test setup, we chose the VC-SLAM corpus (Burgdorf et al., 2022) with more than 100 rich semantic models, i.e., models that contain more than the minimal set of concepts and relations. It represents a heterogeneous collection of datasets from the smart city domain obtained from open data platforms. The data corpus itself is handcrafted and thus also contains modeling variations for the same fact. The underlying ontology consists of 483 different concepts and 117 relations. Each model consists on average of $28.1(\pm 11.34)$ relations and $23.74(\pm 8.4)$ concepts with 14.81 ± 6.58 types, i.e., concepts connected to data, and 8.93 ± 3.78 context nodes. Although, heterogeneity is one of the advantages of the data corpus, it poses a problem to the training approach that relies on single facts appearing multiple times. The only shared domain in a majority of semantic models is geoinformation (cf. Figure 6). We therefore expect the engine to perform particularly well in this domain.

The node embedding training parameters are given in Table 1. All 101 models in the VC-SLAM data corpus are used during training and the evaluation is done using a subset of 10 randomly selected models. We do not treat training and evaluation datasets separately as we expect to deal with triples that have already been seen before.

5.2 Focus Context Evaluation

In order to evaluate the focus context extension generation, we measure the ability of the model to generate coherent chains of concepts. In an optimal case, subsequently recommended concepts are able to build a whole subgraph, e.g., representing a specific part of a domain, of the semantic model. The starting point is an empty model. As target, we define the model as it is defined in VC-SLAM. In order to generate the chains, we use an algorithm that takes into account the output of the recommendation engine and selects the next concept to add to the semantic model

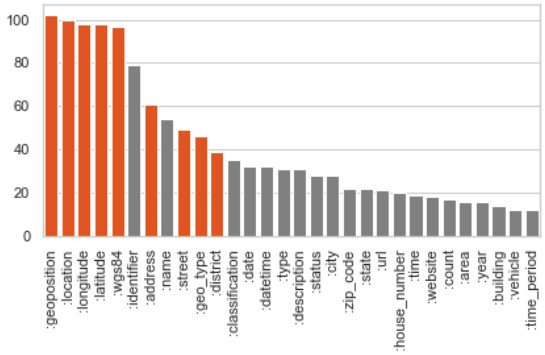


Figure 6: Distribution of nodes over all 101 semantic models in VC-SLAM. Red bars indicate a concept from the geoinformation domain. Only the top 30 are shown. Multiple occurrences of a concept in one model are counted as one.

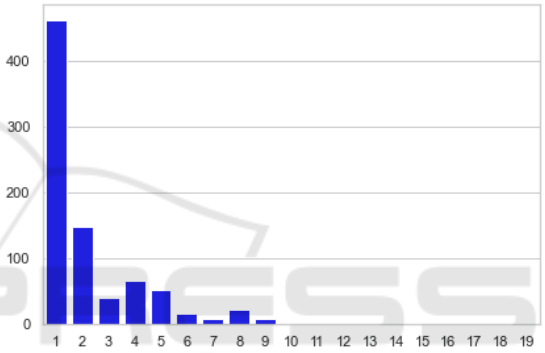


Figure 7: Chain lengths for all 101 models of the VC-SLAM dataset obtained by Algorithm 1.

based on a set of rules. Algorithm 1 illustrated in the Appendix shows the algorithm that, from a random starting node, tracks the current modeling state (model_nodes, history, current chain) and computes an intersection of the concepts in the target model and those from the recommendation for the specific history. A *chain* denotes the sequence of subsequent recommendations that could be found using the recommendation engine. The collection of all chains is called the *trail*. For each resulting node, the algorithm then recursively computes what would happen if this recommendation was accepted, expanding the chain until no suitable concept can be found in the recommendations. After all recursive calls have finished, the state that generated the longest chain is selected and added to the trail. Afterwards, a concept is randomly chosen from all concepts in the target nodes that are not yet present in the model. The process is then repeated until all target concepts have been added to the model and the trail is returned.

Figure 7 shows the length of all chains obtained when running Algorithm 1 once for each model in the data corpus. Chains of length one indicate that

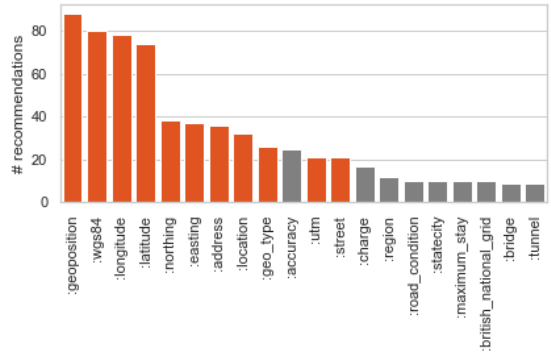


Figure 8: Recommendations of single concepts during the chain evaluation. Highlighted bars indicate a concept from the geoinformation domain.

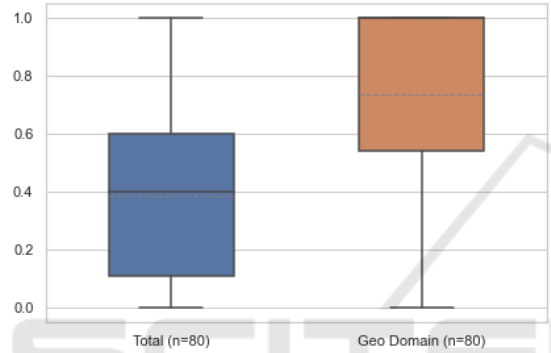


Figure 9: Recall of the general focus recommender over all concepts and limited to the geoinformation domain. Average is denoted by dashed line. The median of geoinformation domain is 1.0.

no matching follow up concepts could be identified by the recommendation engine, thus the chain contains only the randomly selected concept. 358 chains of length greater than one could be found, with the longest chains containing 9 concepts. The high number of chains of length one indicate that the recommendation does not provide sufficiently certain concepts for large parts of the models. Figure 8 shows that the majority of recommended concepts are indeed from the geoinformation domain, showing that, as expected in those subgraphs, focus context recommendation does work. While this was expected (cf. Section 5.1), it shows that there is future potential in the approach once enough data becomes available to sufficiently generate recommendations for other parts of the ontology.

5.3 General Context Evaluation

We evaluated the performance of the general context by extracting the target nodes from all models of the VC-SLAM data corpus. Target nodes were computed by first generating an approximate Steiner Tree (i.e. minimal spanning tree that contains at least all mapped nodes)(Hwang and Richards, 1992) of each model and second, declare all concepts of a model which are not part of this Steiner Tree as a target node (cf. Figure 3, dashed elements). We then used the Steiner Tree model as an input to the general context recommender (cf. Figure 3, solid elements) and compared the resulting recommendations with the set of target nodes. If one or more matches are found, those are added to the model and the next iteration started. If no matching concepts were recommended, we select one of the remaining target concepts and add it to the model before starting the next iteration. We obtained an average recall of 0.42 over all recommended concepts on 80 models, which is similar to the results obtained in the focus context evaluation. For the remaining 21 models, we did not identify any target concepts in addition to the the minimal model identified by the Steiner Tree, eliminating those models from the general context evaluation. Limited to the geoinformation domain, recall increased to 0.72 (Figure 9). This shows that, given more available training data, the engine provides improved results.

5.4 Discussion

The achieved results in both focus context recommendation as well as the general context show the potential benefit of the approach. In both modes, recommendations were generated that improve the modeling process once implemented into a semantic refinement platform such as Karma or PLASMA (cf. Section 3). However, as a content-based recommender system, the approach can only be used for domains where existing semantic models are available. The effective modeling performance of a recommendation system is still dependent on an external modeling system to measure the acceptance rate using a comparative user study. Based on our observations, the achieved results indicate that recommendations will in most cases result in a modeling time reduction compared to manual search of concepts. However, while the approach yielded promising results in the domain of geoinformation, the general performance of the approach could not be estimated due to the lack of training data. Nevertheless, we can assume that if additional training data for other domains becomes available, the approach will show similar results for those domains.

6 CONCLUSION AND OUTLOOK

In this paper, we presented an recommendation engine to generate model extension recommendations for semantic models. The focus of this engine is to enrich auto-generated semantic models with additional information to increase the informativeness and usability of the model. Recommendations are generated using a node embedding trained on existing semantic models using Node2Vec. We have shown that the engine finds suitable recommendations for both operation modes (focus and general context) when used on the VC-SLAM data corpus. However, the most promising results were limited to the domain of geoinformation due to the characteristics of the single suitable data corpus that was used as background data. Once more data sets like VC-SLAM become available, we will evaluate our approach on those.

For our future work, we would like to improve the engine to be able to recommend whole triples, i.e., including the relation, instead of single concepts in order to add more context information that has been observed in the training data in one step. The current state of the engine does not take into account which relation type could potentially be used to link an existing concept to a recommended one and therefore does not exploit all available information. Triple recommendation would further reduce the modeling time. Furthermore, we would also like to recommend replacements to existing elements, targeting elements that were potentially falsely added by fully automated approaches even before the refinement began. Once capable of recommending triples, we plan to integrate the approach into a modeling framework and test the usefulness of the generated recommendations in a comprehensive user study.

REFERENCES

Abdelmageed, N. and Schindler, S. (2020). JenTab: Matching Tabular Data to Knowledge Graphs. The 19th International Semantic Web Conference.

Almonte, L., Guerra, E., Cantador, I., and Lara, J. (2021). Recommender systems in model-driven engineering: A systematic mapping review. *Software and Systems Modeling*.

Baumgartner, M., Dell'Aglio, D., and Bernstein, A. (2021). Entity Prediction in Knowledge Graphs with Joint Embeddings. In *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, pages 22–31, Mexico City, Mexico. Association for Computational Linguistics.

Burgdorf, A., Paulus, A., Pomp, A., and Meisen, T. (2022). VC-SLAM - A Handcrafted Data Corpus for the Construction of Semantic Models. *Data*, 7(2).

Codina, V. and Ceccaroni, L. (2010). Taking advantage of semantics in recommendation systems. volume 220, pages 163–172.

Futia, G., Vetrò, A., and de Martin, J. C. (2020). SeMi: A SEmantic Modeling machIne to build Knowledge Graphs with graph neural networks. *SoftwareX*, 12:100516.

Grover, A. and Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 855–864, New York, NY, USA. Association for Computing Machinery.

Hwang, F. K. and Richards, D. S. (1992). Steiner tree problems. *Networks*, 22(1):55–89.

Knoblock, C. A., Szekely, P., et al. (2012). Semi-Automatically Mapping Structured Sources Into the Semantic Web. In *Extended Semantic Web Conference*, pages 375–390.

Papapanagiotou, P., Katsiouli, P., et al. (2006). RONTO: Relational to Ontology Schema Matching. *AIS Sigsemis Bulletin*, 3(3-4):32–36.

Paulus, A., Burgdorf, A., Pomp, A., and Meisen, T. (2021). Recent advances and future challenges of semantic modeling. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, pages 70–75.

Paulus., A., Burgdorf., A., Puleikis., L., Langer., T., Pomp., A., and Meisen., T. (2021). PLASMA: Platform for Auxiliary Semantic Modeling Approaches. In *Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 2: ICEIS,*, pages 403–412. INSTICC, SciTePress.

Paulus, A., Pomp, A., et al. (2018). Gathering and Combining Semantic Concepts from Multiple Knowledge Bases. In *ICEIS 2018*, pages 69–80, Setúbal, Portugal.

Pham, M., Alse, S., et al. (2016). Semantic Labeling: A Domain-Independent Approach. In *The Semantic Web – ISWC 2016*, pages 446–462, Cham. Springer International Publishing.

Pinkel, C., Binnig, C., et al. (2017). IncMap: a Journey Towards Ontology-based Data Integration. *Datenbanksysteme für Business, Technologie und Web (BTW 2017)*.

Pomp, A., Kraus, V., et al. (2020). Semantic Concept Recommendation for Continuously Evolving Knowledge Graphs. In *Enterprise Information Systems*, Lecture Notes in Business Information Processing. Springer.

Rijgersberg, H., Assem, M., and Top, J. (2013). Ontology of units of measure and related concepts. *Semantic Web*, 4:3–13.

Rümmele, N., Tyshetskiy, Y., and Collins, A. (2018). Evaluating Approaches for Supervised Semantic Labeling. *CoRR*, abs/1801.09788.

Saeedi, A., Peukert, E., and Rahm, E. (2020). Incremental multi-source entity resolution for knowledge graph completion. In *The Semantic Web*, pages 393–408, Cham. Springer International Publishing.

Taheriyan, M., Knoblock, C. A., et al. (2016). Learning the Semantics of Structured Data Sources. *Journal of Web Semantics*, 37-38:152–169.

Uña, D. D., Rümmele, N., et al. (2018). Machine Learning and Constraint Programming for Relational-To-Ontology Schema Mapping. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*.

Vu, B., Knoblock, C., and Pujara, J. (2019). Learning Semantic Models of Data Sources Using Probabilistic Graphical Models. In *The World Wide Web Conference*, WWW '19, pages 1944–1953, New York, NY, USA. ACM.

APPENDIX

Algorithm 1: Algorithm to simulate a modeling process in order to evaluate the generation of subsequent recommendations in the focus context. For each proper recommendation, the current modeling chain is recursively expanded until no further matching recommendations can be found. All computed chains then form the trail which is returned for evaluation.

function BUILD_CHAIN(*chain*, *model_nodes*, *current_node*, *history*)
  history ← *history* + *current_node* # remove first element in list if list longer than n
  history ← TRUNCATE(*history*, *n*)
  chain ← *chain* + *current_node*
  model_nodes ← *model_nodes* + *current_node*
  # generate recommendations for the current history
  focus_recommended ← ENGINE.RECOMMEND(*history*)
  # remove nodes that are already present
  focus_recommended ← REMOVE_EXISTING_NODES(*focus_recommended*, *model_nodes*)
  matches ← GET_MATCHES_IN_TARGET_MODEL(*focus_recommended*, *target_nodes*)
  if len(matches) == 0 **then**
    return *chain*, *model_nodes*, *current_node*, *history* # end the chain
  end if
  results ← [ ] # recursively evaluate recommendations for each possible added concept
  for all match in matches **do**
    new_chain ← BUILD_CHAIN(*chain*, *model_nodes*, *match*, *history*)
    results ← *results* + (*new_chain*, *model_nodes*, *match*, *history*)
  end for
  results ← SORT_BY_CHAIN_LENGTH(*results*) # find the longest chain
  return *results*[0] # return the longest chain and modeling context
end function

function BUILD_TRAIL(*target_nodes*)
  model_nodes ← [ ] # start with an empty model
  random_node ← GET_UNMAPPED_RANDOM_NODE(*model_nodes*, *target_nodes*)
  current ← *random_node*
  trail ← [ ], *chain* ← [ ], *history* ← [ ]
  while |*model_nodes*| < |*target_nodes*| **do**
    chain, *model_nodes*, *current*, *history* ← BUILD_CHAIN(*chain*, *model_nodes*, *current*, *history*)
    trail ← *trail* + *chain*
    history ← [ ], *chain* ← [ ]
    current = GET_UNMAPPED_RANDOM_NODE(*model_nodes*, *target_model*)
  end while
  return *trail*
end function