Towards Semantic Identification of Temporal Data in RDF

Lars Runge[1], Wolfgang May[2]

Institut für Informatik, Georg-August-Universität Göttingen, Germany

Abstract
Temporal data, i.e., time-annotated data in the RDF data model expressing the valid time of a Resource, Property or Relationship can be stored in various forms. However, the modeling of temporal data and how properties that describe them are named heavily depends on the creators' preferences. Thus, to utilize the temporal axis in SPARQL queries the user needs intimate knowledge of the data source and preferably of the underlying ontology. This paper covers the analysis of temporal data usage in various open RDF data sources, especially in the LOD cloud, and proposes a framework to automatically identify temporal data facts in unknown data sources by semantical and structural analysis. The temporal data is then made available through a temporal extension of the SPARQL query language which allows uniform temporal query patterns regardless of its specific underlying implementation.

Keywords
Temporal Data, RDF, Machine Learning

1. Introduction

Efficient storage and querying of temporally annotated data (subsequently referred to as "temporal data") is a highly researched topic for every emerging data model. Temporal data enables researchers and practitioners to gain valuable insights into the evolution of events, behaviors, and processes over time. The interest in temporal data is so great that special temporal databases were developed to better facilitate the analysis of time-series data like temporal-spatial data.

But, the term temporal data can mean different things in the context of databases. Depending on the purpose of the temporal information it can be classified into several distinct sub-categories. For example, one purpose is the tracking of *transaction* times, which concerns itself with the temporal logging of the different states a fact has or had inside the database management system (DBMS). These cover temporal information such as creation, modification and deletion timestamps. Another purpose is the *valid* time, which describes when a fact is considered true outside of the database. This could be for example the time interval when a person lived or the time point when an event happened. The processes and framework described in this paper are specifically concerned with temporally annotated data in the context of valid time. While transaction times can be useful information for database admins and for stream processing, semantical queries usually consider valid time information. When utilizing this type of information the user can query the database facts with constraints on the temporal axis in the context of the real world.

However, the ease of annotating data facts with temporal information is heavily dependent on the data model. Especially when it comes to storing temporal data in the RDF data model, one runs into its inherent limitations. The data model of standard RDF is limited to triples $subject, predicate, object$ (s,p,o), or equivalently, binary relations where each describes the relationship between a resource and either a literal or another resource. It it is indeed possible to put a resource itself on the temporal axis by simply describing it with its properties, and annotating these property values and relationships with temporal information. However, this requires – at least– quadrupels $s, p, o, timepoint$ or quintuples $s, p, o, from, until$. Quadrupels are basically present in RDF when named data graphs are considered to represent snapshots or intervals. Then, a temporal dataset consists of several such graphs. Nevertheless, the most common way to compensate for these limitations is to use *reification* to represent temporally annotated data in a *single* graph. Reification means, to construct auxiliary resources that help to group together related information. How these auxiliary resources are structured and referenced is up to the creator of the data set. Community-drafted extensions of the RDF 1.1 standard like *RDF-star*[1] tackle this structural problem by providing a concise way to make statements about statements through *quoted triples*. Some vendor and open-source libraries such as *GraphDB* already implement this to various extents even though it is not yet integrated in the RDF W3C recommendation.

Besides the structural challenges of temporal valid data in RDF, there exist also semantical ones. The names of properties that describe the valid time of a resource can also be very diverse, because languages developed different words to describe the beginning and end of things depending on the semantical context. For example a per-

Workshop GVDB'23: 34th workshop on basics of database systems, June 07–09, 2023, Hirsau, Baden-Württemberg, Germany
lars.runge@informatik.uni-goettingen.de (L. Runge); may@informatik.uni-goettingen.de (W. May)

son's beginning is pretty likely to be described as their "birth" and the ending as their "death", while a building has its "construction" and "demolition" respectively. Using these words as property names help the human user to deduce the semantic meaning of its property value after the discovery, but also limit the exploration of algorithms that do not comprehend this human knowledge.

The structural and semantic discrepancies of temporal data between various RDF data sources requires a new user to gain an intimate understanding of the inner workings of the data set before being able to query temporal data efficiently. Data integration and queries against several data sources, e.g., in LOD, even require to use different modelings in a single task. This is especially unpleasant because the RDF data model was designed to make data readily available in the Web and encourages the exploration of unknown data sets through links as is demonstrated through the Linked Open Data (LOD) initiative. To lighten the burden of data exploration from the user this paper covers a framework proposal that aims to automatically identifying temporal data in a new data set and allows uniform temporal query patterns regardless of its specific underlying implementation.

2. Preliminaries

In addition to the structural and semantic difficulties of temporal data, time values in RDF datasets can also be expressed in various forms. The standardized way is to use the RDF-compatible XSD datatypes defined in XML Schema [2]. These cover `xsd:time` and `xsd:date` values as well the their combination `xsd:dateTime`, which utilizes a subset of the ISO 8601 [3] format, and recurring and partial dates like `xsd:gYear`. Of course as is often the case with open data sets it is not guaranteed that every time value has the most specific datatype associated with it. For example, a date value might be given the generic `xsd:string` datatype and year values are often stored as simple `xsd:int` which makes the identification of time values also a parsing problem. Furthermore, time-focused ontologies were developed to extend the base functionality. The *OWL-Time OWL-2 DL ontology of temporal concepts* [4] aims to provide time values as separately addressable resources and focuses on ordering of temporal entities. The topological temporal relations are based on the algebra of binary relations on intervals developed by Allen [5, 6]. But also more general ontologies cover some time-related properties and encourage a standardized naming scheme in RDF datasets as well. The `DCMI Metadata Terms` [7] for example provide `dc:date`, which allows any type of RDFS literal as its range, but assumes a date value according to ISO 8601, and `dc:temporal` with the range of DCMIs own *PeriodOfTime* class.

As a result, entities can be connected either directly to their temporal values as is the case when a property ranges over a time-expressing datatype (XSD or otherwise) or indirectly if they are further encapsulated through a time-describing resource. This is often the case if the relevant time frame of the value is not a single time point, but a time interval. For simplicity, in the scope of this paper for all further mentions of time values, it should be assumed that a mapping is used that converts the encapsulated indirect connections to direct ones and that they are in a format that conforms to an XSD datatype.

Related Work. Implementing and querying temporal information in RDF data has been a long-time research topic [8]. In [9], Tappolet et al. introduce *τ-SPARQL* based on named graphs to store temporal data and temporal wildcards *[?s, ?e]* that bind to the respective underlying named graph(s) to bind the temporal context of triples. Grandi [10] proposed a variation of temporal SPARQL named *T-SPARQL* that aims to adapt features from the *TSQL2* [11] temporal query language for relational databases. The query language works on an underlying multi-temporal RDF database model consisting of multiple time domains. The triples in the *WHERE* clause are extended with an optional fourth position `?s ?p ?o | ?t` where *?t* binds the complex timestamps. With the temporal SPARQL implementation in [12] Robatjazi highlights the advantages of hiding the intricate n-ary relationship structures from the user by providing manually crafted higher-level predicates to ease the querying. These high-level predicates are later mapped to the underlying properties in the query process by a RDF/RDFS reasoner. Unfortunately all these approaches propose a modeling and an extension of SPARQL with temporal constructs that utilizes *their* respective new model. To use them on an existing dataset would require to transform it beforehand. A limitation that hinders the exploration of new data sources for example by following links in the LOD and using SPARQL endpoints.

For investigating the availability of temporal information in the LOD cloud, Rula et al.[13] analysed the 2011 Billion Triple Challenge (BTC) dataset. Besides a document-centric perspective on temporal data, the research also covered a fact-centric perspective, which includes the different structural representations such as reification, n-ary relationships and temporal named graphs. They found that the overall occurrence of temporal information is quite small, but if reification is used to annotate temporal information then it is much more likely that a custom n-ary relationship is used instead of the more formal *rdf:Statement.* A finding that we reproduced in our own analysis of the LOD cloud. In addition to that they analysed the occurrences of n-ary relationships with manual sampling because they are impossible to identify just by analysing the graph structure. A limitation that we want to tackle by including semantical information into the analysis that tries to deduce the meaning behind property names.

The identification of the semantical meanings/senses of a word is the core topic in the field of word-sense induction. The general consensus in linguistics is that the context of a word is tremendously important to understand its meaning [14, 15]). Arora et al. [16] have shown that word embeddings produced even from simple models like *word2vec* [17] capture the meanings of words as long as the context was used to train them. Further research results [18, 19] indicate that the inclusion of additional sense information for example from *BERT* [20] sense embeddings or semantic networks like *WordNet* [21] improve disambiguation tasks.

3. Modeling of Temporal Data in Open RDF Datasets

To verify the results of [13] and to check the current status of temporal data usage in open RDF sources, we conducted our own analysis. Given that the structure of triples is not sufficient to clearly identify them as temporal data, we look for further characterizing features. The obvious solution would be to search and employ the specialized properties and classes in standardized ontologies that denote some type of temporal annotation. Thus we focus on the adoption rate of these standardized notions to determine whether they are sufficiently utilized to be reliable in the identification process. Again, we were first of all interested how valid times are described and stored in a wide variety of topics.

For that the analysis utilizes first and foremost the data sources listed in the LOD cloud [22], which cover different disciplines from geographical and governmental data to life science and music. In addition to that a selection of smaller and bigger open RDF data sources found by search engines were also included. To focus on more recent and active sources and in compliance with the requirements of high quality open RDF data, we specifically searched for data sources that feature a functioning SPARQL service. At the time of the analysis the LOD-cloud featured 1300 RDF datasets with over 450 registered SPARQL endpoints. Unfortunately of those only just over 100 were able to answer a simple query. This prompted the inclusion of further RDF dataset repositories for example *Govdata.de* [23] and *data.europa.eu* [24] for governmental data.

We tested the datasets on the occurrence of *XSD* time-related datatypes, *OWL-Time* ontology classes and *DCMI* time-related properties. Additionally we tested the usage of the *rdf:Statement* class to check if reifications were done strictly formally. Of course it would also be interesting to analyse how often custom n-ary relationships are constructed as a form of reification for the purpose of temporal annotation. Unfortunately, to the best of our knowledge no process for this was proposed so far except manual identification. A circumstance we aim to address with this work. Table 1 contains our findings.

Table 1
Occurrences of time related datatypes, classes and properties in open RDF datasets

XSD datatypes	Frequency in %
dateTime	39
date	28
gYear	4.6
gYearMonth	3.1
gDay / gMonth	1.6

TIME ontology classes	Frequency in %
Instant	8.4
Interval	3.7
TemporalEntity	3.7
DateTimeDescription	3.7
DateTimeInterval	1.9

DCMI time properties	Frequency in %
dcterms:date	55
dc:date	22
dcterms:temporal	7.4
dcterms:valid	6.5

Reification Method	Frequency in %
rdf:Statement	8.3

From the results of the analysis it can be projected that a significant portion of open RDF datasets utilize at least the more general time properties of DCMI and standard XSD datatypes. Furthermore the OWL-Time ontology was adopted at least partially in more datasets as one might suspect given its niche application. In contrast, the more specialized datatypes of singular time units and properties to explicitly describe valid time are rarely seen. One needs to keep in mind that not all disciplines have a use for valid times or even time values in general, but it seems likely that most datasets containing valid time facts apply their own custom property names to describe them. In addition to that we found that only 8.3% of the datasets utilize the *rdf:Statement*, which matches the findings of [13] and the conclusion that reifications are done more likely informally as n-ary relationships.

As a result we determined that we can not rely on these standardized notions for a comprehensive identification of temporal data. The temporal annotation of data in RDF is still a highly varied procedure that heavily depends on the personal preferences of its creator. Therefore a suitable automatic identification framework must cater to these custom characteristics as well as possible.

4. Proposed Framework

Employing the advances in programmatic processing of word senses, the foundation of the proposed framework for identifying temporal data in unknown sources is a two-step analysis. First a structural analysis on graph patterns is done to create a general set of potential temporal data candidates. Then a semantical analysis of the property names being used in the candidates to perform the actual classification follows. For this crucial second step, a neural network is employed that is trained on a corpus of English words that appear in the context of temporal values. The goal of the supervised learning approach is to create a classifier that can differentiate words that are typically used to describe temporal data. In the beginning we focus on property names that denote the temporal sense of *valid* times, but the process can be extended to also include other temporal senses, for example *active* times. Regardless of their specific temporal sense, properties that are thus classified by their name are in the following generally referred to as time properties.

After running the identification process on a dataset, it is planned to use the classified temporal data fragments to create a dictionary which stores the relevant time properties for each resource. Through further heuristical examination of the findings and its surrounding graph patterns, it is aimed to generalize the dictionary from specific resources to general classes. Supplementing an OWL ontology for this step can help with the generalization effort. Ultimately this dictionary can then in turn be deployed as the backbone for a temporal SPARQL implementation that allows the user to query the dataset without knowing its specific time properties by providing a lookup opportunity for the returned variable bindings. A typical sketch of the consolidation of the SPARQL query is shown in Figure 1.

To illustrate the details of each process, they will be explained on the basis of one of the most prominent open RDF data sources, *Wikidata* [25]; more specifically on the resource that describes the Entity "Germany"[1]. To maintain readability the common Wikidata prefixes[2] will be used. Notably the *wdt:* prefix for properties with direct $resource \to value$ connections and the *p:*, *ps:* & *pq:* prefix used for reification. Where *p:prop* is the connection from the resource to the auxiliary reified resource, *ps:prop* the simple key value and *pq:prop* any number of qualifier properties that further describe the [*resource*, *prop*, *value*] statement as shown in figure 2. These could be for example the temporal annotations *P585* (point in time), *P580* (start time) and *P582* (end time). It should be noted that in addition to the reification there exists almost always a direct property *wdt:prop* to the same value as the path *p:prop / ps:prop*.

[1]https://www.wikidata.org/wiki/Q183
[2]https://www.wikidata.org/wiki/EntitySchema:E49

4.1. Structural analysis

The main purpose of the structural analysis is to reduce the search space for temporal facts before applying the semantical analysis step. In addition to that, it allows to gain a higher-level understanding of the dataset in case an OWL ontology is not provided and may even uncover relations between properties that are not covered by it.

The common consensus is that there are four different ways to annotate triples with temporal information in standard RDF. They range from more straightforward approaches like singleton properties (construct new properties) to more abstract approaches like standard reification (construct new auxiliary resources), n-ary relationships (introduce new resources, properties and classes) and even named graphs (construct new RDF graphs).

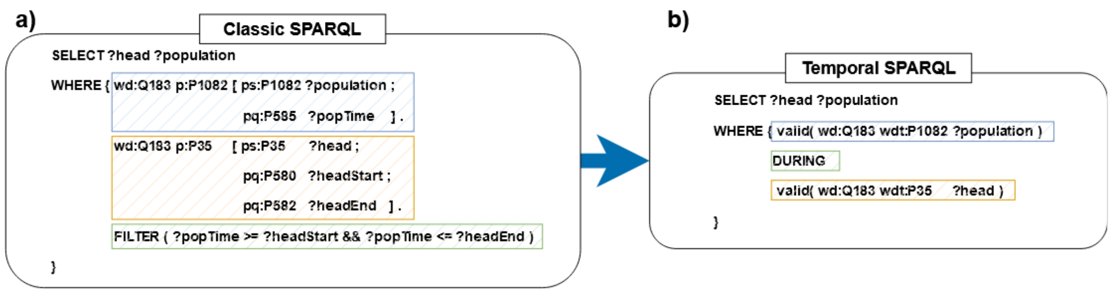
However, we want to utilize a higher level view on the structure of temporal data that focuses more on the purpose of the temporal annotation, especially *what* is being annotated. This view centers around what a user would want to express in queries and illustrates the detachment from the specific underlying structural implementation. Additionally, it introduces the necessary flexibility to accommodate new structural extensions to RDF like RDF-star later on by adding appropriate transformations to the unifying view. In the confines of this paper it should be assumed that there already exist suitable transformations for every mentioned standard structure for temporal annotations even though only the ones found in Wikidata will be represented in the examples. Thus we determined the following three different kinds of temporal data.

4.1.1. Temporal Resources

The first and simplest form of temporal data are temporal resources. These express the validity of the resource itself, so for example the birth and death dates of a person or the time point an event happened. As a standard graph pattern they are not different from any other property that is being used to describe the resource. Thus any pattern

```
<resource> ?time-property ?time
```

where *?time* is a temporal value, is a potential candidate. There should be only a single property for each resource which denotes its valid time if it is a time point, or a pair of properties for intervals, denoting their begin and end.



```
SELECT ?head ?population
WHERE { wd:Q183 p:P1082 [ ps:P1082 ?population ;
                          pq:P585  ?popTime    ] .
        wd:Q183 p:P35   [ ps:P35   ?head ;
                          pq:P580  ?headStart ;
                          pq:P582  ?headEnd    ] .
        FILTER ( ?popTime >= ?headStart && ?popTime <= ?headEnd )
}
```

```
SELECT ?head ?population
WHERE { valid( wd:Q183 wdt:P1082 ?population )
        DURING
        valid( wd:Q183 wdt:P35    ?head )
}
```

Figure 1: SPARQL queries on the Wikidata dataset to select the head of state of Germany and the results of population census done during their mandate. a) in standard SPARQL b) in our pursued temporal SPARQL notation (syntax not final)

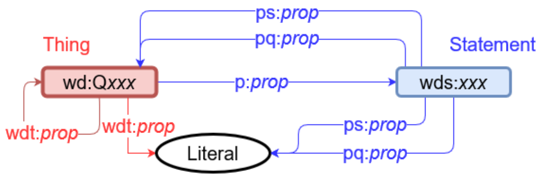


Figure 2: Usage of selected Wikidata prefix for direct properties (wdt:) and reification (p:, ps:, pq:)

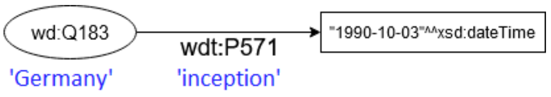


Figure 3: Representation of the triple describing the temporal resource "Germany" in Wikidata

In the case of Wikidata's Germany this would match the properties *schema:dateModified*, *wdt:P571* and *wdt:P1249* as temporal data candidates. The subsequent semantic algorithm would then classify *wdt:P571* as the correct time property. A representation of the triple is shown in Figure 3. For some entities it can be possible that multiple entries of its time property exist. This can make sense for more conceptual entities that can stop existing for a while or change their nature, which is the case for the entity Germany. This forces the introduction of reified resources representing each period of its valid time. Luckily Wikidata uses the same property names for the reified resources, but with different prefixes. The property path *p:P571/ps:571* then points to the relevant time values. *wdt:P571* can be seen more as a direct shortcut to the most recent value for the property P571 (inception). In most datasets this shortcut to the most recent valid time will not exist, but nevertheless an interesting consideration must be made when querying the valid time of resources with multiple valid intervals. Does the user want to consider every valid interval or only the most recent one?

4.1.2. Temporal Literals

Annotating properties with literal values is the most common form of temporal data. There are two different ways to achieve this in the triple format. One way is to create a second property which points to the relevant time value. The name of this time property should be in some form semantically connected to the property it augments. So for example annotating the value of a *population* property could be done with a property describing when the population census was done like *populationTime*. This variant is so unorthodox and only works if there can only be one instance of the property for each resource, that it is really rare to encounter in real-world datasets. The overwhelmingly more common strategy is to achieve the annotation through reification. The base graph pattern that matches temporal literal candidates then looks like this:

```
<resource> ?property [
      ?time-property  ?time  .
      ?value-property  ?value  ]
```

with *?time* a temporal value and *?value* the literal value of the original property before annotation. The names of both *?property* or *?value-property* are likely to stand in relation to the orignal property name, but are not guaranteed to be. In addition the reification could be done as a blank node or as an auxiliary resource with or without a proper class type that may give further semantical hints for its purpose.

Naturally this simple graph pattern would match a huge number of data fragments that are no temporal data. Further heuristical filtering must be applied to reduce the number of generated candidates. For example a very effective condition would be that no other resources reference the reification. An obstacle so far is the identification of the correct value-property if the reification features multiple properties for even more detailed descriptions. It could be argued that every property grouped by the reification is temporally annotated by the time-property, but the user is likely to be more interested in the value of the original property.

In Wikidata the "population" property is called *P1082*. Each occurrence of *p:P1082* points to a reified resource with an internal name, which in turn groups among others the property *ps:P1082* describing the value of the population and the property for "point in time" *pq:P585* describing the valid time of the value. Figure 4 shows a representation of these data facts for Germany.

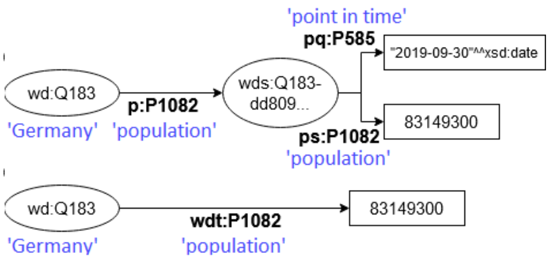


Figure 4: Representation of the triples describing a temporal literal "population" and its shortcut triple in Wikidata

Furthermore if there are multiple time-annotated instances of the original property, for example multiple population values for each year a population census occurred, then the dataset might also feature a shortcut property to the most recent value. Most datasets do not include these direct properties like *wdt:P1082*, but if they exist, they can be beneficial in various ways. First of all identifying them would allow to create a mapping so the user can utilize either in the queries. A heuristical approach for this could not rely on the thoroughly planned naming scheme of Wikidata, but would need to check for property pairs that most of the time appear together with one pointing to a (set of) temporal reification(s) and the other pointing to a literal that is also the value occurring in the newest instance of those reifications. This would also help to indicate which property is the correct value-property inside the temporal reification.

4.1.3. Temporal Relationships

Temporal relationships are quite like temporal properties, but allowing the value of the temporally annotated property to be a resource. This opens up new possibilities how the temporal reification is modeled. A fully connected temporal reification would feature properties both from and to the resources which the relationship describes. This is rarely the case so that only one of the directions is present for each resource. The most common structure is similar to temporal literals with one resource pointing to the temporal reification which itself points towards the related resource as its value. An example for this is the "member of" *P463* property in Wikidata relating countries to organizations. Germany is a member of the European Union *wd:Q458* with "start time" *P580* '1957-03-25'. While there exist temporal relationships from the EU to Germany ("has part(s)" and "contains the administrative territorial entity"), they do not share the same temporal reification. Nevertheless the possible existence of inverse properties means that the condition outlined to reduce the number of false temporal candidates needs to be adjusted.

Again, just like with temporal literals, there might exist a shortcut property which connects the two resources without the temporal information, which in this example is *wdt:P463* as shown in Figure 5. The same benefits apply for temporal relationships too.

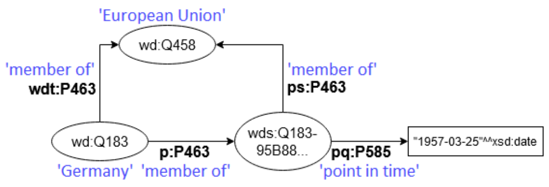


Figure 5: Representation of the triples describing a temporal relationship "member of" and its shortcut triple in Wikidata

4.2. Semantic temporal words

The structural analysis of the dataset through graph patterns can only generate temporal data *candidates*. Any optimization done on the patterns and further heuristic analysis on structural information may reduce the number of generated false positives, but does not remove them entirely. There will always be data fragments that match the pattern, but are no true temporal data. A human on the other hand can discern which properties are used to describe valid time by interpreting the meaning of their name in the context of the resources' real-world nature.

The main building block of the proposed framework aims to harness this semantical knowledge to obtain an accurate classifier for the produced candidates. To do this, a machine learning approach has been chosen that is trained with manually labelled property names extracted from a wide variety of open RDF datasets. The goal is that the network will be able to label unknown words even from topics it was not trained upon.

4.2.1. Network development

The learning approach is planned in multiple stages. First, the construction of a simple network that should just classify whether an English word appears in the context of valid time. This network is then expanded and tweaked until a satisfactory accuracy is achieved to serve as a proof of concept.

Then, the class of "valid time" property names is split into more refined categories that denote the occurrence of a time point, the beginning of a time interval, or the ending of a time interval. Formally it could be argued that any time point is a time interval of minimum duration and thus a time point property could also be the beginning (or end) of an interval. *Semantically*, and for the actual meaning of the words to be detected, the difference between *events* and time intervals is crucial and helpful, e.g., the events of the signature and the notice of termination of a rental contract, and the actual duration. Also, for the execution of temporal queries it is of vital importance to estimate the boundaries of the resources' valid time, including intervals that have a beginning, but no end (so far). Semantically, finding out which time point and/or time frame properties relate to each other should help in the data exploration. In case multiple time values are present it is not always prudent to take the minimum and maximum values as the start and end time respectively (cf. the rental contract case).

Because these words rely so heavily on the context they are used in, the ultimate goal would be that the network can be expanded so far that it not only takes the words, but also for example the classes of the resource into consideration. It is suspected that training relations like 'birth for a person is like construction for a building' would boost the performance significantly. More intricate is the case of a rental contract, which is a temporal object whose existence starts with the signature, and it describes the property "being rented" for another (maybe ongoing) interval, and its existence may bind the contractors even longer. For this, we extracted so far over 6000 property names with classes from the associated context that were found to be related to time values. We manually labelled them accordingly for [valid timepoint] [valid time begin] [valid time end] to perform the initial training of the neural network.

4.2.2. Semantic word embedding

Technically, to utilize words in machine learning they are be first converted into numeric vector representations. To realistically achieve any of the previously mentioned goals it is paramount that the words are embedded to vectors in such a way that the semantical meaning of them is conserved. We are particularly interested in an embedding model that keeps the relationship between words intact, so that for example the vector pointing from "human" to "birth" might relate to the vector from "building" to "construction", and from "contract" to "signature".

Much research has been done on this topic, which is often covered under the generic term "Word2Vec". Some well known approaches include skip-grams [26] and CBOWs (continuous bag of words). One characteristic of real-world property names is that they are often a combination of multiple words for example *startTime* or *productionDate*. Therefore it was crucial to choose a model that is able to produce vector embeddings for concatenated words like *fastText* [27], developed from the Facebook AI Research (FAIR) lab. It provides models for either skipgram or cbow and can produce embeddings even for unknown words through the use of subword information. Utilizing subword information should enable fastText to produce meaningful embeddings especially for concatenated words by factoring in every single component word. However, for getting realistic results the property names need to be words that at least resemble correct english dictionary words. As in the Wikidata case, this is sometimes not directly given. There, to create a unifying model across all languages, every resource and property is tokenised into an alphanumerical code, like the above P1082 for the "population" property. The proper property names are then stored as an *rdfs:label* for each language. For cases like these, the learning process uses the corresponding label, preferrably in the English language (@en), instead of the tokenised property name.

5. Conclusion

In this paper we have discussed issues of modeling temporal data in RDF. We have analyzed how temporal data is represented structurally in real-world RDF data, and we have provided an approach for classifying properties describing the valid time of data facts by the semantical meaning of their names. With this, temporal data can be detected in unknown data sources.

The second part of the project will exploit the lifting of temporal data to a conceptual view on its modeling for designing a temporal SPARQL query extension which provides generic temporal constructs instead of querying the exact (temporal) RDF triples. With this approach the user will be able to query the specific purpose of the temporal annotation, beginning with valid times but extensible for further purposes like active times.

References

[1] O. Hartig, B. Thompson, Foundations of an alternative approach to reification in RDF, CoRR abs/1406.3399 (2014). URL: http://arxiv.org/abs/1406.3399. arXiv:1406.3399.

[2] XML Schema Datatypes, W3C: The xml schema built-in datatypes, 2014. Www.w3.org/TR/rdf11-concepts/#dfn-rdf-compatible-xsd-types.

[3] ISO 8601-1:2019, ISO: Date and time — representations for information interchange, 2019. Www.iso.org/standard/70907.html.

[4] TIME Ontology, W3C: Time ontology in owl, 2022. Www.w3.org/TR/2022/CRD-owl-time-20221115/.

[5] J. F. Allen, G. Ferguson, Actions and Events in Interval Temporal Logic, Springer Netherlands, Dordrecht, 1997, pp. 205–245. URL: https://doi.org/10.1007/978-0-585-28322-7_7. doi:10.1007/978-0-585-28322-7_7.

[6] J. F. Allen, Towards a general theory of action and time, Artificial Intelligence 23 (1984) 123–154. URL: https://www.sciencedirect.com/science/article/pii/0004370284900080. doi:https://doi.org/10.1016/0004-3702(84)90008-0.

[7] Dublin Core DCMI, Dublin Core: Dcmi metadata terms, 2020. Www.dublincore.org/specifications/dublin-core/dcmi-terms.

[8] C. Gutierrez, C. A. Hurtado, A. Vaisman, Introducing time into rdf, IEEE Transactions on Knowledge and Data Engineering 19 (2007) 207–218. doi:10.1109/TKDE.2007.34.

[9] J. Tappolet, A. Bernstein, Applied temporal rdf: Efficient temporal querying of rdf data with sparql, in: L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, E. Simperl (Eds.), The Semantic Web: Research and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 308–322.

[10] F. Grandi, T-sparql: A tsql2-like temporal query language for rdf., in: ADBIS (local proceedings), 2010, pp. 21–30.

[11] R. T. Snodgrass, The TSQL2 temporal query language, volume 330, Springer Science & Business Media, 2012.

[12] M. RobatJazi, M. Z. Reformat, W. Pedrycz, P. Musilek, Lori: Linguistically oriented rdf interface for querying fuzzy temporal data, in: Flexible Query Answering Systems 2015: Proceedings of the 11th International Conference FQAS 2015, Cracow, Poland, October 26-28, 2015, Springer, 2015, pp. 337–352.

[13] A. Rula, M. Palmonari, A. Harth, S. Stadtmüller, A. Maurino, On the diversity and availability of temporal information in linked open data, in: P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, E. Blomqvist (Eds.), The Semantic Web – ISWC 2012, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 492–507.

[14] R. C. Anderson, Z. Shifrin, The meaning of words in context, Theoretical issues in reading comprehension (2017) 331–348.

[15] N. S. Dash, Context and contextual word meaning., SKASE Journal of Theoretical Linguistics (2008).

[16] S. Arora, Y. Li, Y. Liang, T. Ma, A. Risteski, Linear algebraic structure of word senses, with applications to polysemy, Transactions of the Association for Computational Linguistics 6 (2018) 483–495.

[17] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013. arXiv:1301.3781.

[18] S. Mizuki, N. Okazaki, Semantic specialization for knowledge-based word sense disambiguation, arXiv preprint arXiv:2304.11340 (2023).

[19] A. Saha, A. Gittens, B. Yener, Word sense induction with knowledge distillation from bert, arXiv preprint arXiv:2304.10642 (2023).

[20] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[21] WordNet, Princeton university "about wordnet." wordnet. princeton university., 2010. Wordnet.princeton.edu.

[22] LOD-cloud Website, LOD cloud: The linked open data cloud monitors and provides links to datasets available as open linked data, 2019. Lod-cloud.net.

[23] Das Datenportal für Deutschland, Open Government : Verwaltungsdaten transparent, offen und frei nutzbar, 2023. Www.govdata.de.

[24] data.europe.eu Website, data.europa.eu : The official portal for european data, 2023. Data.europa.eu.

[25] Wikidata Website, Wikidata: multilingual knowledge graph hosted by the wikimedia foundation, 2023. Www.wikidata.org.

[26] D. Guthrie, B. Allison, W. Liu, L. Guthrie, Y. Wilks, A closer look at skip-gram modelling, in: LREC, volume 6, 2006, pp. 1222–1225.

[27] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin, Advances in pre-training distributed word representations, in: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.