Applying Markov Logic for Debugging Probabilistic Temporal Knowledge Bases

Jakob Huber, Christian Meilicke, Heiner Stuckenschmidt
Research Group Data and Web Science
University Mannheim, Germany
jakob|christian|heiner@informatik.uni-mannheim.de

Abstract

A probabilistic temporal knowledge base contains facts that are annotated with a time interval and a confidence score. The interval defines the time span for which it can be assumed that the fact is true with a probability that is expressed by the confidence score. Given a probabilistic temporal knowledge base, we propose the use of Markov Logic in combination with Allen's interval calculus to select the most probable consistent subset of facts by computing the MAP state. We apply our approach on a specific domain of DBpedia, namely the domain of academics. We simulate a scenario of extending a knowledge base automatically in an open setting by adding erroneous facts to the facts stated in DBpedia. Our results indicate that we can eliminate a large fraction of these errors without removing too many correctly stated facts.

1 Introduction

Integrating web extracted facts in a consistent knowledge base that uses a well-defined taxonomy of concepts and roles and a URI schema to identify individuals is a challenging task tackled by many researchers in the previous years, e.g. [15, 16, 7]. In this paper, we are concerned with a special case: the problem of integrating temporal facts. We refer to a temporal fact as a fact that is annotated with an interval, which refers to a time span for which it can be assumed that the fact is true. Examples are facts like *Einstein studied at ETH Zurich from 1896 to 1901* or *Einstein was born on 14 March 1879* where the time span is implicitly specified as part of the given fact.

Such facts typically appear amongst those facts extracted by Open Information Extraction systems like ReVerb [10] and they can be used to extend semantic knowledge bases as DBpedia. However, many of these facts are ambiguous and first need to be mapped to the URI schema of the target knowledge base. The fact itself might have been incorrect and the mapping process might introduce an incorrect interpretation. As a result, an extended knowledge base is generated that contains probabilistic temporal and non-temporal facts. In this paper, we present an approach for detecting and removing erroneous facts from such a probabilistic temporal knowledge base. In particular, we propose to model the challenge of removing the erroneous facts as a maximum a posteriori (MAP) inference problem using the formalism of Markov Logic [6] in combination with Allen's interval calculus [1]. Moreover, we present first experiments where we apply our approach to a specific sub domain of DBpedia.[1]

In Section 2, we explain how to model the temporal debugging problem in Markov Logic. In particular, we describe how to encode temporal facts and how to model temporal constraints. Moreover, we illustrate how to solve the debugging problem by computing the MAP state. In Section 3, we

[1]Our paper is a condensed version of a technical report available at https://ub-madoc.bib.uni-mannheim.de/37100/. We refer to the report w.r.t. details missing due to the lack of space.

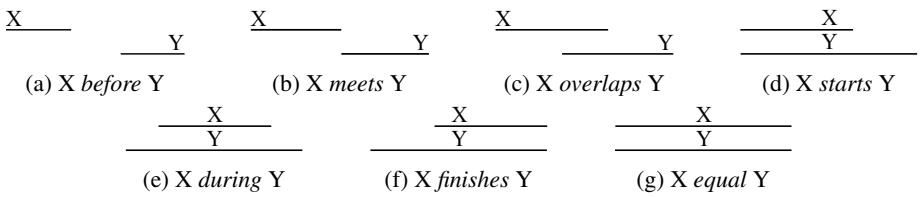

Figure 1: Allen Interval's Algebra [1]: Overview on the Relations.

describe our experiments. In these experiments we extend the sub-domain of academics in DBpedia with randomly generated facts. We apply our method and report about the results. Finally, we end with a discussion of related work and conclude in Section 4.

2 Approach

We debug temporal probabilistic knowledge bases by computing the MAP state of a Markov Logic Network [6]. Therefore, we define first-order predicate symbols for the different types of statements contained in a knowledge base. Thus, we use the predicate `triple` to express facts of the form *subject predicate object* that hold time independent (non-temporal facts and terminological knowledge). Additionally, we use the predicate `quad` to state facts that are annotated with temporal information, given as an interval, which indicates the validity time of a fact. This leads to the following typed predicates, where $\mathtt{r}_1$, $\mathtt{r}_2$ and $\mathtt{r}_3$ refer to a resource or a literal and $\mathtt{t}$ refers to an interval.

$$\mathtt{triple}(\mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_3) \qquad\qquad \mathtt{quad}(\mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_3, \mathtt{t})$$

The predicate symbol `quad` is used to encode temporal facts, whereby the temporal information is expressed in the last argument. We assign an identifier $\mathtt{t}_1, \ldots, \mathtt{t}_n$ to each interval $\mathtt{t}_i$ occurring in the knowledge base. In order to express the temporal relations of the intervals that are annotated to facts, we define additional predicate symbols that allow stating the temporal relation of intervals. In particular, we use the following predicates of Allen's interval algebra (see Figure 1):

$$\mathtt{before}(\mathtt{t}_i, \mathtt{t}_j) \quad \mathtt{meets}(\mathtt{t}_i, \mathtt{t}_j) \quad \mathtt{overlaps}(\mathtt{t}_i, \mathtt{t}_j) \quad \mathtt{starts}(\mathtt{t}_i, \mathtt{t}_j)$$
$$\mathtt{during}(\mathtt{t}_i, \mathtt{t}_j) \quad \mathtt{finishes}(\mathtt{t}_i, \mathtt{t}_j) \quad \mathtt{equal}(\mathtt{t}_i, \mathtt{t}_j)$$

We calculate all relations among the intervals occurring in the knowledge base and ground the respective predicates using interval identifiers. As the set of relations is jointly exhaustive and pairwise disjoint, i.e., exactly one relation holds between a pair of intervals [12], we are able to define the temporal relations as observed predicates. Due to the pairwise disjointness, groundings of observed predicates that are not explicitly stated are considered to be false. Hence, it is legit to compute all temporal relations before calculating the MAP state and it is also not necessary to define additional constraints that ensure the characteristics (e.g. pairwise disjointness) of the predicates expressing the interval relations. Due to the fact that many statements are only valid at a specific point in time (e.g. birth dates), we model time points as intervals with the same lower and upper bound.

Based on those predicate symbols, we define domain-specific constraints as universally quantified first-order formulas that allow detecting inconsistencies in a knowledge base. It is possible to define the constraints as weighted (soft) or unweighted (hard) formulas. A Markov Logic solver [14] resolves the existing conflicts by computing the MAP state of the Markov Logic Network. As an example, consider a knowledge base that contains the following weighted facts.

```
(F1) 0.8 quad(Einstein, birthYear, 1879, id1)    id1 := [1879,1879]
(F2) 0.5 quad(Einstein, birthYear, 1955, id2)    id2 := [1955,1955]
(F3) 1.0 quad(Einstein, deathYear, 1955, id2)
```

You may note that the third and fourth argument of `F1`, `F2`, and `F3` represent redundant information. However, we have chosen this modeling style, because there are also temporal facts whose validity time cannot be extracted from the third argument, e.g., *Einstein studied at ETH Zurich [1896,1901]*.

We compute the temporal relation between all intervals, in our example `id1` and `id2`, and express it with Allen's interval algebra. This leads to the ground predicate `before(id1,id2)`. Moreover,

we define a constraint that ensures that the birth date of an entity occurs before its death date.

$$\texttt{quad(x,"birthYear",t1,i1)} \wedge \texttt{quad(x,"deathYear",t2,i2)} \Rightarrow \texttt{before(i1,i2)}$$

This leads to a clash between the facts `F2` and `F3` as the constraint infers `before(id2,id2)`. In order to resolve the conflict, one of the facts has to be removed from the knowledge base as all possible groundings of the temporal predicates are considered as false if they are not explicitly stated. The Markov Logic solver will discard `F2` as this leads to a higher objective. So, only the facts `F1` and `F2` remain in the knowledge base. Hence, the MAP state corresponds to a consistent subset of facts extracted from an inconsistent knowledge base.

Moreover, we have incorporated the RDF(S) entailment rules [11, 5], comparable to the modeling style presented in [13], where a similar approach has been proposed to support reasoning in $\mathcal{EL}^{++}$ [3] which is a description logic dialect with a complete set of completion rules. In particular, features like domain and range restrictions are important to detect inconsistencies in many scenarios. Moreover, we extend the RDF(S) standard with disjointness and temporal functionality. For instance, we are able to declare `birthYear` as temporal functional property in order to ensure that at most one birth year is assigned to a person. With respect to the example, the facts `F1` and `F2` would interfere with each other. During the calculation of the MAP state, the reasoner would remove `F2` as long as its weight is smaller than the sum of the weights of `F1` and `F3`.

3 Experiments

We apply our method to a specific subdomain of DBpedia. In particular, we are interested in researchers, their alma mater, their birth and death date, and the influence relationships (e.g. influenced, academic advisor, notable student) between researchers. The following facts are, for example, stored in DBpedia.

```
db:Gottlob_Frege dbo:influenced db:Ludwig_Wittgenstein
db:Gottlob_Frege dbo:birthDate 1848-11-11
```

Suppose that the following web-extracted fact, which has been mapped to the vocabulary of DBpedia, is added to the knowledge base.

```
db:Ludwig_Wittgenstein dbo:deathDate 1751-04-29
```

Obviously, one of these facts is incorrect. In the following, we first describe how we generated a dataset that contains such kind of errors. Then we apply our method to that dataset and report about its capability to detect such errors.

We first extracted the relevant subset of DBpedia by selecting all facts using such properties as `dbo:academicAdvisor`, `dbo:influenced`, `dbo:birthDate`, `dbo:almaMater`, `dbp:established`. The complete list contains twelve properties. Based on these properties, we derive $150k$ facts (RDF triples) describing entities contained in the domain of academics. Given this set of probabilistic temporal and non temporal facts, we define a set of domain specific temporal constraints based on a common sense understanding of the chosen domain. Typical examples of such constraints are the following ones.

$$\begin{aligned}\texttt{triple(x, dbo:academicAdvisor, y)} \wedge \texttt{quad(x, dbo:birthDate, t1, i1)} \wedge \\ \texttt{quad(x, dbo:deathDate, t2, i2)} \Rightarrow \texttt{before(i1,i2)} \\ \texttt{triple(x, dbo:almaMater, y)} \wedge \texttt{quad(x, dbp:established, t1, i1)} \wedge \\ \texttt{quad(x, dbo:deathDate, t2, i2)} \Rightarrow \texttt{before(i1,i2)}\end{aligned}$$

We injected several types of erroneous facts to the extracted dataset. For a stated temporal fact, we added an incorrect fact by mixing or swapping digits of the year that has originally been specified. In some cases, we added or subtracted a small number in the range of 1 to 20 to the given year. Moreover, we randomly assigned birth or death years to persons for which this information was not yet given. With respect to the non-temporal statements, we randomly picked two instances and connected them with one of the given properties. It can be assumed that this results in nearly all cases in an incorrect fact. Thus, we can compare the outcome of applying our approach against a gold standard, by assuming that each originally stated fact is correct, and each added fact is incorrect.

Input	Debugging			Repaired Dataset			ΔF
P	P	R	F	P	R	F	
0.99	0.80	0.63	0.70	1.00	1.00	1.00	0.002
0.91	0.80	0.64	0.71	0.97	0.98	0.97	0.022
0.80	0.81	0.65	0.72	0.92	0.96	0.94	0.050
0.67	0.82	0.65	0.73	0.84	0.93	0.88	0.084
0.57	0.83	0.65	0.73	0.78	0.90	0.83	0.106
0.50	0.84	0.65	0.73	0.72	0.87	0.79	0.119

Table 1: Precision (P), Recall (R) and F-measure (F) for debugging process and repaired dataset.

We randomly assigned weights in the range $(0.0, 1.0]$ to the injected erroneous facts as well as to each of the originally stated facts. This makes the task significantly harder, compared to a setting where we treat the originally stated facts as hard facts or where we give higher weights to these facts.

The results of our experiments are depicted in Table 1. The first column informs about the precision of the generated dataset. We have injected a fraction of 1%, 10%, 25%, 50%, 75%, and finally 100% incorrect statements to the dataset. Thus, the precision (P) of the generated dataset varies from 0.99 to 0.5. For example, the last row of the table refers to a dataset where every second fact is incorrect. The recall of this dataset is always 1.0, because we never removed a fact of the extracted dataset. For all datasets, we have computed the MAP state, which corresponds to the repaired dataset. In doing so we computed precision (P), recall (R) and f-measure (F) for the debugging process and for the final outcome. The precision of the debugging process refers to the fraction of removed axioms that were indeed incorrect; recall refers in this context to the fraction of incorrect axioms that have been removed. The precision and recall of the repaired dataset is computed by comparing it to the originally extracted dataset. The rightmost column (ΔF) informs about the gain in F-measure that we computed by comparing the F-measure of the input dataset with the F-measure of the repaired dataset.

The results show that we are able to improve the data quality of an erroneous dataset. This is depicted in the increase of F-measure in the ΔF column for all test cases. With respect to the test case where every second fact is incorrect, we are able to increase the precision by 0.22 (from 0.5 up to 0.72), while recall is only reduced to 0.87 (from 1.0). Most of the removed facts are indeed incorrect (a repair precision of > 0.8 means that at least for 4 of 5 removals are proper removals), while we are able to detect more than half of the incorrect facts (debugging recall of the repair is > 0.5). Note that a chance classifier, that removes randomly half of all facts, is expected to achieve a debugging precision and recall of 0.5 for the last test case. Remember that we randomly assigned confidence scores to both correct and incorrect statements. Thus, the positive outcome of our experiments, compared to a chance classifier, is not self-evident, but can be explained by the interaction of several constraints and the fact that we compute an optimal solution. The theoretically optimal solution in the context of the Markov Logic Network, turns out to be (at least) a good solution with respect to our application scenario.

At first sight it is not evident why our approach improves the precision of the debugging process when we increase the share of wrong facts. Therefore, we provide an example that illustrates one possible scenario related to this artifact. It also indicates that multiple constraints need to be taken into account in order to resolve inconsistencies. Considering the following set of statements that contains one wrong statement (`C` = correct, `W` = randomly generated wrong statement):

```
(C1) 0.42 quad(Ferdinand_Cohn, dbo:birthYear, 1828, [1828,1828])
(C2) 0.05 quad(Ferdinand_Cohn, dbo:deathYear, 1898, [1898,1898])
(C3) 0.02 triple(Georg_Lunge, dbo:academicAdvisor, Ferdinand_Cohn)
(W1) 0.56 quad(Ferdinand_Cohn, dbo:birthYear, 1952, [1952,1952])
```

With respect to this input, our approach keeps only the generated birth year of `Ferdinand_Cohn` (`W1`) as this leads to the consistent dataset with the highest weight. The correct birth year of `Ferdinand_Cohn` (`C1`) gets removed as a person has at most one birth date. The correct death date (`C2`) gets removed as it has to be after the birth date. `Georg_Lunge` lived from 1828 to 1898. Hence, `Ferdinand_Cohn`, whose in the knowledge base stated birth year is now 1952, cannot be

his doctoral advisor (C3). So, only the wrong statement W1 remains. However, adding an additional wrong statement W2 to this conflict set improves the result:

```
(W2) 0.58 quadW(Ferdinand_Cohn, dbo:birthYear, generated, [1540,1540])
```

While this statement is also wrong, it makes nevertheless more sense in the context of the specific constraints. Due to the additional erroneous statement W2, which has a relatively high weight, it is sufficient to only remove the correct statement that describes the birth year of Ferdinand_Cohn (C1) as well as statement W1. So, he can be the doctoral advisor of Georg_Lunge and also his correct death date causes no inconsistencies. In summary, the application removed three correct statements from the dataset in the first case. By adding an additional statement to the dataset, the application removes only one correct statement and also one wrong statement which leads to a better F-measure compared to the first scenario.

It takes $\approx$7 minutes for the initial dataset ($150k$ facts, precision = 1.0) and $\approx$18 minutes for the largest extended dataset ($300k$ facts, precision = 0.5) to determine the consistent subset. We executed our experiments on a virtual machine running Ubuntu 12.04 that has access to two threads of the CPU (2.4 GHz) and 16 GB RAM. We used the Markov Logic solver rockIt [14] to compute the MAP state of the Markov Logic Network.

4 Related Work and Conclusion

The following research is closely related to the approach and application scenario presented in this paper. In [4, 2] OWL 2.0 is extended in order to enable temporal reasoning for supporting temporal queries. The authors define SWRL rules that are compatible with a reasoner that supports DL-safe rules in order to detect inconsistencies. However, their system can only detect if a knowledge base is consistent but cannot resolve the existing conflicts. [17, 9, 8] proposed different approaches to resolve temporal conflicts at query time. In particular, they define temporal constraint as Datalog rules and integrate a subset of Allen's interval algebra. [17] follow a histogram-based approach to determine the consistent subset of facts having the highest probability. [9] model the optimization problem as a scheduling task and introduce an approximation of a scheduling algorithm. However, these approaches do not explicitly incorporate terminological knowledge while resolving the conflicts and do also not support weighted constraints.

We have presented an approach for debugging probabilistic temporal knowledge bases by computing the MAP state of a Markov Logic Network. The approach is generic in the sense that it can easily be applied to different domains by extending the basic calculus with domain specific temporal constraints. Moreover, our approach can also leverage additional terminological knowledge due to the integration of the RDF(S) completion rules and further rules related to disjointness and property functionality. While we have shown the benefits of our approach in the context of a synthetic dataset, we are currently extending our experiments for integrating facts extracted by ReVerb with DBpedia.

References

[1] James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] Eleftherios Anagnostopoulos, Sotiris Batsakis, and Euripides GM Petrakis. Chronos: A reasoning engine for qualitative temporal information in owl. *Procedia Computer Science*, 22:70–77, 2013.

[3] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the el envelope further. In *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*. Citeseer, 2008.

[4] Sotiris Batsakis, Kostas Stravoskoufos, and Euripides GM Petrakis. Temporal reasoning for supporting temporal queries in owl 2.0. In *Knowledge-Based and Intelligent Information and Engineering Systems*, pages 558–567. Springer, 2011.

[5] Dan Brickley and Ramanathan Guha. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[6] Pedro Domingos and Daniel Lowd. Markov logic: An interface layer for artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–155, 2009.

[7] Arnab Dutta, Christian Meilicke, and Heiner Stuckenschmidt. Semantifying triples from open information extraction systems. In *Frontiers in Artificial Intelligence and Applications*, volume 264. IOS Press, 2014.

[8] Maximilian Dylla, Iris Miliaraki, and Martin Theobald. A temporal-probabilistic database model for information extraction. *Proceedings of the VLDB Endowment*, 6(14):1810–1821, 2013.

[9] Maximilian Dylla, Mauro Sozio, and Martin Theobald. Resolving temporal conflicts in inconsistent rdf knowledge bases. In *14. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)*, pages 474–493, 2011.

[10] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10, 2011.

[11] Patrick Hayes. RDF semantics. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

[12] Gérard Ligozat and Jochen Renz. What is a qualitative calculus? a general framework. In *PRICAI 2004: Trends in Artificial Intelligence*, volume 3157 of *Lecture Notes in Computer Science*, pages 53–64. Springer, 2004.

[13] Mathias Niepert, Jan Noessner, and Heiner Stuckenschmidt. Log-linear description logics. In *IJCAI*, pages 2153–2158, 2011.

[14] Jan Noessner, Mathias Niepert, and Heiner Stuckenschmidt. Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2013.

[15] Stephen Soderland and Bhushan Mandhani. Moving from textual relations to ontologized relations. In *AAAI Spring Symposium: Machine Reading*, pages 85–90. AAAI, 2007.

[16] Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu, Mausam, and Oren Etzioni. Adapting open information extraction to domain-specific relations. *AI Magazine*, 31(3):93–102, 2010.

[17] Yafang Wang, Mohamed Yahya, and Martin Theobald. Time-aware reasoning in uncertain knowledge bases. In *Proceedings of the Fourth International VLDB workshop on Management of Uncertain Data (MUD 2010)*, pages 51–65, 2010.