19th International Conference on Information Fusion
Heidelberg, Germany - July 5-8, 2016

Predicting the Co-Evolution of Event and Knowledge Graphs

Cristóbal Esteban[1,2], Volker Tresp[1,2], Yinchong Yang[1,2], Stephan Baier[2] and Denis Krompaß[1]
[1]Siemens AG and [2]Ludwig Maximilian University of Munich, Munich, Germany.
{cristobal.esteban, volker.tresp, denis.krompass}@siemens.com
{yinchong.yang, stephan.baier}@campus.lmu.de

***Abstract*—Knowledge graphs have evolved as flexible and powerful means for representing general world knowledge. Typical examples are DBpedia, Yago, or the Google Knowledge Graph, which all started off by representing information derived from Wikipedia and were then greatly expanded. In this paper we use the concept of a knowledge graph to present information about specific classes of entities, such as patients or users. The knowledge graph represents all that is known about the entities and their relationships and the goal is to integrate and exploit that information for prediction and decision support. In previous papers it was shown that embedding learning, a.k.a. representation learning, is capable of modelling large-scale semantic knowledge graphs, by exploiting information that describes the context of an entity in the knowledge graph. In Machine Learning we often map the knowledge graph to a tensor representation. Then we learn the latent representations of the entities that compose the tensor and use them to predict unobserved facts. However knowledge graphs represent the current status of the world and therefore they lack of a temporal dimension, which means we can only use them to predict facts about the present moment. In this paper we introduce an additional set of tensors that contain temporal information. Each of this event tensors contains all the events that occurred on a particular time step. Our goal will be to predict the events that will happen in future time steps, using for that task both dynamic information from the previous event tensors and static information that is stored in the knowledge graph. Therefore, this architecture allows us to fuse static and dynamic information to predict future events. We present experiments showing how this model performs well in multiple scenarios: medical data, a recommendation engine and sensor data.**

I. Introduction

Knowledge graphs have evolved as flexible and powerful means for representing general world knowledge. Typical examples are DBpedia, Yago, or the Google Knowledge Graph, which all started by representing information derived from Wikipedia and were then greatly expanded. In this paper we use the concept of a knowledge graph to present information about specific classes of entities, such as patients or users. The knowledge graph represents all that is known about the entities and their relationships and the goal is to integrate and exploit that information for prediction and decision support.

In previous publications it was shown how a triple knowledge graph (KG) can be represented as a multiway array (tensor) and how a statistical model can be formed by deriving latent representations of generalized entities. Successful models are, e.g., RESCAL [22], Translational Embeddings Models [9], Neural Tensor Models [31] and the multiway neural networks as used in [10].

In these publications KGs were treated as static: A KG grew more links when more facts became available but the ground truth value associated with a link was considered time invariant. In this paper we address the issue of KGs where triple states depend on time. In the simplest case this might consider simple facts like "Obama is president of the United States", but only from 2008-2016. Another example is a patient whose status changes from sick to healthy or vice versa. Most popular KGs like Yago [32], DBpedia [1] Freebase [8] and the Google Knowledge Graph [30] have means to store temporal information.

Without loss of generality, we assume that changes in the KG always arrive in form of events, in the sense that the events are the gateway to the KG. For a given time step, events are described by a typically very sparse event triple graph, which contains facts that change some of the triples in the KG, e.g., from *True* to *False* and vice versa. KG triples which do not appear in the event graph are assumed unchanged.

An example might be the statement that a patient has a new diagnosis of diabetes, which is an information that first appears as an event in the event graph but is then also transmitted to the KG. Other events might be a prescription of a medication to lower the cholesterol level, the decision to measure the cholesterol level and the measurement result of the cholesterol level; so events can be, e.g., actions, decisions and measurements. In a similar way as the KG is represented as a KG tensor, the event graphs for all time steps can be represented as an event tensor. Statistical models for both the KG tensor and the event tensor can be derived based on latent representations derived from the tensor contents.

Although the event tensor has a representation for time, it is by itself not a prediction model. Thus, we train a separate prediction model which estimates future events based on the latent representations of previous events in the event tensor and the latent representations of the involved generalized entities in the KG tensor. In this way, a prediction can, e.g., use both background information describing the status of a patient and can consider recent events. Since some future events will be absorbed into the KG, by predicting future events, we also predict likely changes in the KG and thus obtain a model for the evolution of the KG as well.

978-0-9964527-4-8

The paper is structured as follows. The next section discusses related work. In Section III we review statistical KG models based on latent representations of the involved generalized entities. In Section IV we discuss the event tensor and its latent representations. In Section V we demonstrate how future events can be estimated using a prediction model that uses latent representations of the KG model and the event model as inputs. In the applications of this paper in Section VI, we consider patients and their changing states, users and their changing movie preferences and weather stations and their changing signal statistics. In the latter we show how —in addition to event data— sensory data can be modeled. Section VII contains our conclusions and discusses extensions.

II. Related Work

There is a wide range of papers on the application of data mining and machine learning to KGs. Data mining attempts to find interesting KG patterns [5], [27], [24]. Some machine learning approaches attempt to extract close-to deterministic dependencies and ontological constructs [19], [13], [16]. The paper here focuses on *statistical* machine learning in KG where representation learning has been proven to be very successful.

There is considerable prior work on the application of tensor models to temporal data, e.g., EEG data, and overviews can be found in [14] and [20]. In that work, prediction is typically not in focus, but instead one attempts to understand essential underlying temporal processes by analysing the derived latent representations.

Some models consider a temporal parameter drift. Examples are the BPTF [36], and [11]. Our model has a more expressive dynamic by explicitly considering recent histories. Markov properties in tensor models were considered in [26], [25]. In that work quadratic interactions between latent representations were considered. The approach described here is more general and also considers multiway neural networks as flexible function approximators.

Our approach can also be related to the neural probabilistic language model [4], which coined the term *representation learning*. It can be considered an event model where the occurrence of a word is predicted based on most recent observed words using a neural network model with word representations as inputs. In our approach we consider that several events might be observed at a time instance and we consider a richer family of latent factor representations.

There is considerable recent work on dynamic graphs [17], [23], [33], [28] with a strong focus on the Web graph and social graphs. That work is not immediately applicable to KGs but we plan to explore potential links as part of our future work.

III. The Knowledge Graph Model

With the advent of the Semantic Web [7], Linked Open Data [6], Knowledge Graphs (KGs) [32], [1], [8], [30], triple-oriented knowledge representations have gained in popularity. Here we consider a slight extension to the subject-predicate-object triple form by adding the value $(e_s, e_p, e_o;$ *Value*$)$ where *Value* is a function of s, p, o and can be the truth value of the triple or it can be a measurement. Thus *(Jack, likes, Mary; True)* states that Jack likes Mary, and *(Jack, hasBloodTest, Cholesterol; 160)* would indicate a particular blood cholesterol level for Jack. Note that e_s and e_o represent the entities for subject index s and object index o. To simplify notation we also consider e_p to be a generalized entity associated with predicate type with index p.

A machine learning approach to inductive inference in KGs is based on the factor analysis of its adjacency tensor $\underline{X}$ where the tensor element $x_{s,p,o}$ is the associated *Value* of the triple (e_s, e_p, e_o). Here $s = 1, \ldots, S$, $p = 1, \ldots, P$, and $o = 1, \ldots, O$. One can also define a second tensor $\underline{\Theta}^{KG}$ with the same dimensions as $\underline{X}$. It contains the natural parameters of the model and the connection to $\underline{X}$. In the binary case one can use a Bernoulli likelihood with $P(x_{s,p,o}|\theta^{KG}_{s,p,o}) \sim \text{sig}(\theta^{KG}_{s,p,o})$, where $\text{sig}(arg) = 1/(1 + \exp(-arg))$ is the logistic function. If $x_{s,p,o}$ is a real number than we can use a Gaussian distribution with $P(x_{s,p,o}|\theta^{KG}_{s,p,o}) \sim \mathcal{N}(\theta^{KG}_{s,p,o}, \sigma^2)$.

In representation learning, one assigns an r-dimensional latent vector to the entity e denoted by $\mathbf{a}_e = (a_{e,0}, a_{e,1}, \ldots, a_{e,r})^T$. We then model using one function

$$\theta^{KG}_{s,p,o} = f^{KG}(\mathbf{a}_{e_s}, \mathbf{a}_{e_p}, \mathbf{a}_{e_o})$$

or, using one function for each predicate,

$$\theta^{KG}_{s,p,o} = f^{KG}_p(\mathbf{a}_{e_s}, \mathbf{a}_{e_o}).$$

For example, the RESCAL model [22] is

$$\theta^{KG}_{s,p,o} = \sum_{k=1}^{r}\sum_{l=1}^{r} R_{p,k,l} a_{e_s,k} a_{e_o,l},$$

where $R \in \mathbb{R}^{P \times r \times r}$ is the core tensor. In the multiway neural network model [10] one uses

$$\theta^{KG}_{s,p,o} = \text{NN}(\mathbf{a}_{e_s}, \mathbf{a}_{e_p}, \mathbf{a}_{e_o})$$

where NN stands for a neural network and where the inputs are concatenated. These approaches have been used very successfully to model large KGs, such as the Yago KG, the DBpedia KG and parts of the Google KG. It has been shown experimentally that models using latent factors perform well in these high-dimensional and highly sparse domains. For a recent review, please consult [21].

We also consider an alternative representation. The idea is that the latent vector stands for the tensor entries associated with the corresponding entity. As an example, $\mathbf{a}_{e_s}$ is the latent representation for all values associated with entity e_s, i.e., $x_{s,:,:}$. [1] It is then convenient to assume that one can calculate a so-called M-map of the form

$$\mathbf{a}_{e_s} = M^{\textit{subject}} x_{s,:,:}. \tag{1}$$

[1] If an entity can also appear as an object ($o : e_o = e_s$), we need to include $x_{:,:,o}$.

Here $M^{subject} \in \mathbb{R}^{r \times (PO)}$ is a mapping matrix to be learned and $x_{s,:,:}$ is a column vector of size PO.[2] For multilinear models it can be shown that such a representation is always possible; for other models this is a constraint on the latent factor representation. The advantage now is that the latent representations of an entity can be calculated in one simple vector matrix product, even for new entities not considered in training. We can define similar maps for all latent factors. For a given latent representation we can either learn the latent factors directly, or we learn an M-matrix.

The latent factors, the M-matrices, and the parameters in the functions can be trained with penalized log-likelihood cost functions described in the Appendix.

IV. The Event Model

Without loss of generality, we assume that changes in the KG always arrive in form of events, in the sense that the events are the gateway to the KG. For a given time step, events are described by a typically very sparse event triple graph, which contains facts that change some of the triples in the KG, e.g., from *True* to *False* and vice versa. KG triples which do not appear in the event graph are assumed unchanged.

Events might be, e.g., *do a cholesterol measurement*, the event *cholesterol measurement*, which specifies the value or the order *take cholesterol lowering medicine*, which determines that a particular medication is prescribed followed by dosage information.

At each time step events form triples which form a sparse triple graph and which specifies which facts become available. The event tensor is a four-way tensor $\underline{Z}$ with $(e_s, e_p, e_o, e_t; \textit{Value})$ and tensor elements $z_{s,p,o,t}$. We have introduced the generalized entity e_t to represent time. Note that the characteristics of the KG tensor and the event tensor are quite different. $\underline{X}$ is sparse and entries rarely change with time. $\underline{Z}$ is even sparser and nonzero entries typically "appear" more random. We model

$$\theta^{event}_{s,p,o} = f^{event}(\mathbf{a}_{e_s}, \mathbf{a}_{e_p}, \mathbf{a}_{e_o}, \mathbf{a}_{e_t}).$$

Here, $\mathbf{a}_{e_t}$ is the latent representation of the generalized entity e_t.

Alternatively, we consider a personalized representation of the form

$$\theta^{\textit{pers-event}, s=i}_{p,o} = f^{\textit{pers-event}}(\mathbf{a}_{e_p}, \mathbf{a}_{e_o}, \mathbf{a}_{e_s=i,t}).$$

Here, we have introduced the generalized entity $e_{s,t}$ for a subject $s = i$ at time t which stands for all events of entity $s = i$ at time t.

Since representations involving time need to be calculated online, we use M-maps of the form

$$\mathbf{a}_{e_{s,t}} = M^{\textit{subject, time}} z_{s,:,:,t}$$

The cost functions are again described in the Appendix.

[2]The M matrices are dense but one dimension is small (r), so in our settings we did not run into storage problems. Initial experiments indicate that random projections can be used in case that computer memory becomes a limitation.

V. The Prediction Model

A. Predicting Events

Note that both the KG-tensor and the event tensor can only model information that was observed until time t but it would not be easy to derive predictions for future events, which would be of interest, e.g., for decision support. The key idea of the paper is that events are predicted using both latent representations of the KG and latent representations describing recently observed events.

In the prediction model we estimate future entries in the event tensor $\underline{Z}$. The general form is

$$\theta^{\textit{predict}}_{s,p,o,t} = f^{\textit{predict}}(\textit{args}) \quad \text{or} \quad \theta^{\textit{predict}}_{s,p,o,t} = f^{\textit{predict}}_{p,o}(\textit{args})$$

where the first version uses a single function and the latter uses a different function for each (p, o)-pair.[3] Here, *args* is from the sets of latent representations from the KG tensor and the event tensor.

An example of a prediction model is

$$\theta^{\textit{predict}}_{s,p,o,t} = f^{\textit{predict}}_{p,o}(\mathbf{a}_{e_s}, \mathbf{a}_{e_{s,t}}, \mathbf{a}_{e_{s,t-1}}, \ldots, \mathbf{a}_{e_{s,t-T}}).$$

where the prediction is based on the latent representations of subject, object and predicate from the KG-tensor and of the time-specific representations from the event tensor.

Let's consider an example. Let $(e_s, e_p, e_o, e_t; \textit{Value})$ stand for (*Patient, prescription, CholesterolMedication, Time; True*). Here, $\mathbf{a}_{e_s}$ is the profile of the patient, calculated from the KG model. Being constant, $\mathbf{a}_{e_s}$ assumes the role of parameters in the prediction model. $\mathbf{a}_{e_{s,t}}$ describes all that *so far* has happened to the patient at the same instance in time t (e.g., on the same day). $\mathbf{a}_{e_{s,t-1}}$ describes all that happened to the patient at the last instance in time and so on.

We model the functions by a multiway neural network with weight parameters W exploiting the great modeling flexibility of neural networks. The cost function for the prediction model is

$$\text{cost}^{\textit{predict}} = -\sum_{z_{s,p,o,t} \in \underline{Z}} \log P(z_{s,p,o,t} | \theta^{\textit{predict}}_{s,p,o,t}(A, M, W)) \quad (2)$$

$$+\lambda_A \|A\|^2_F + \lambda_W \|W\|^2_F + \lambda_M \|M\|^2_F.$$

A stands for the parameters in latent representation and M stands for the parameters in the M-matrices. For a generalized entity for which we use an M-matrix, we penalize the entries in the M-matrix; for a generalized entity for which we directly estimate the latent representation we penalize the entries in the corresponding latent terms in A. Here, $\|\cdot\|_F$ is the Frobenius norm and $\lambda_A \geq 0$, $\lambda_M \geq 0$ and $\lambda_W \geq 0$ are regularization parameters.

[3]The different functions can be realized by the multiple outputs of a neural network.

B. Predicting Changes in the KG

In our model, each change in the status of the KG is communicated via events. Thus each change in the KG first appears in the event tensor and predictions of events also implies predictions in the KG. The events that change the KG status are transferred into the KG and the latent representations of the KG, i.e., $\mathbf{a}_{e_s}, \mathbf{a}_{e_p}, \mathbf{a}_{e_o}$, are re-estimated regularly (Figure 1).

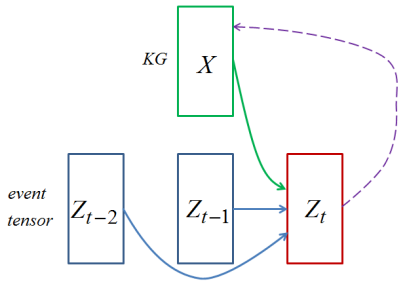

Fig. 1. The figure shows an example where the event tensor is predicted from the representations of the events in the last two time steps and from the KG representation. The dotted line indicate the transfer of observed events into the KG.

C. More Cost Functions

Associated with each tensor model and prediction model, there is a cost function (see Appendix). In our experiments we obtained best results, when we used the cost function of the task we are trying to solve. In the most relevant prediction task we thus use the cost function in Equation 2. On the other hand, we obtained faster convergence for the prediction model if we initialize latent representations based on the KG model.

VI. Experiments

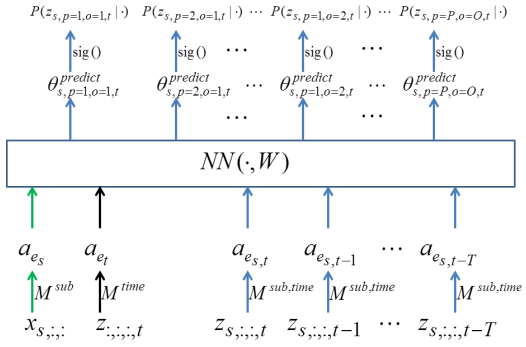

Fig. 2. The prediction model for the clinical data.

A. Modeling Clinical Data

The study is based on a large data set collected from patients that suffered from kidney failure. The data was collected in the Charité hospital in Berlin and it is the largest data collection of its kind in Europe. Once the kidney has failed, patients face a lifelong treatment and periodic visits to the clinic for the rest of their lives. After the transplant has been performed, the patient receives immunosuppressive therapy to avoid the rejection of the transplanted kidney. The patient must be controlled periodically to check the status of the kidney, adjust the treatment and take care of associated diseases, such as those that arise due to the immunosuppressive therapy. The dataset contains every event that happened to each patient concerning the kidney failure and all its associated events: prescribed medications, hospitalizations, diagnoses, laboratory tests, etc. [18], [29]. The database started being recorded more than 30 years ago and it is composed of dozens of tables with more than 4000 patients that underwent a renal transplant or are waiting for it. For example, the database contains more than 1200 medications that have been prescribed more than 250000 times, and the results of more than 450000 laboratory analyses.

This is particularly important for the estimation of drug-drug interactions (DDI) and adverse drug reactions (ADR) in patients after renal transplant.

We work with a subset of the variables available in the dataset. Specifically, we model medication prescriptions, ordered lab tests and lab test results. We transformed the tables into an event oriented representation where the subject is the

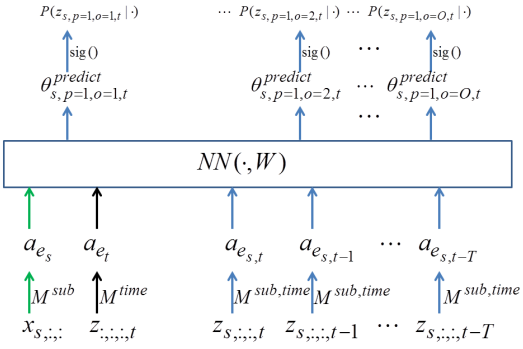

Fig. 3. The prediction model for the recommendation data.

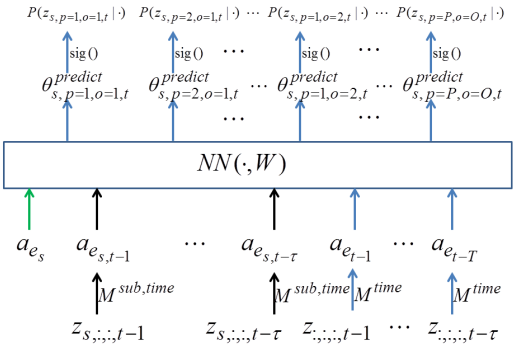

Fig. 4. The prediction model for the sensor data. $\mathbf{a}_{e_s}$ is directly estimated without using an M-mapping.

TABLE I
SCORES FOR NEXT VISIT PREDICTIONS. AUPRC STANDS FOR AREA UNDER PRECISION-RECALL CURVE. AUROC STANDS FOR AREA UNDER ROC CURVE. *ET* STANDS FOR OUR PROPOSED MODEL THAT USES ONLY PAST EVENT INFORMATION BUT NO INFORMATION FROM THE KG.

	AUPRC	AUROC	Time (hours)
ET	0.574 ± 0.0014	0.977 ± 0.0001	6.11
Logistic Regression	0.554 ± 0.0020	0.970 ± 0.0005	4.31
KNN	0.482 ± 0.0012	0.951 ± 0.0002	17.74
Naive Bayes	0.432 ± 0.0019	0.843 ± 0.0015	39.1
Constant predictions	0.350 ± 0.0011	0.964 ± 0.0001	0.001
Random	0.011 ± 0.0001	0.5	-

patient and where time is a patient visit. We encoded the lab results in a binary format representing normal, high, and low values of a lab measurement, thus *Value* is always binary.

The prediction model is

$$\theta^{predict}_{s,p,o,t} = f^{predict}_{p,o}(\mathbf{a}_{e_s}, \mathbf{a}_{e_t}, \mathbf{a}_{e_{s,t}}, \mathbf{a}_{e_{s,t-1}}, \ldots, \mathbf{a}_{e_{s,t-T}}).$$

Note that we have a separate function for each (p, o)-pair. $\mathbf{a}_{e_s}$ are patient properties as described in the KG. $\mathbf{a}_{e_{s,t}}$ represents all events known that happened at visit t for the patient (e.g., the same visit for which we want to make a prediction). $\mathbf{a}_{e_{s,t-1}}$ represents all events for the patient at the last visit, etc. $\mathbf{a}_{e_t}$ stands for the latent representation of all events at visit t for all patients and can model if events are explicitly dependent on the time since the transplant. Regarding the input window, we empirically found that $T = 6$ is optimal. The architecture is shown in Figure 2.

The first experiment consisted of predicting the events that will happen to patients in their next visit to the clinic given the events that were observed in the patients' previous visits to the clinic (i.e. by using the events that occurred to the patient from $\mathbf{a}_{e_{s,t}}$ until $\mathbf{a}_{e_{s,t-6}}$). The experiment was performed 10 times with different random splits of the patients. Thus we truly predict performance on patients which were not considered in training! Table I shows how our proposed model outperforms the baseline models. The "constant predictor" always predicts for each event the occurrence rate of such event (thus the most common event is given the highest probability of happening, followed by the second most common event, and so on). Note that we are particularly interested in the Area Under Precision-Recall Curve score due to the high sparsity of the data and our interest in predicting events that will actually happen, as opposed to the task of predicting which events will not be observed. In the last column of Table I we also report the time that it took to train for each model with the best set of hyperparameters in the first random split.

Next we repeat the experiment including the KG-representation of the patient, which contains static variables of the patient such as blood type and gender, i.e., $\mathbf{a}_{e_s}$, and also used $\mathbf{a}_{e_t}$. Table II shows the improvement brought by the inclusion of the KG representation. The last row in Table II shows the result of making the predictions just with the KG representation of the patient (i.e. without the past event information), demonstrating clearly that information on past events is necessary to achieve best performance.

TABLE II
SCORES FOR FULL VISIT PREDICTIONS WITH AND WITHOUT THE INFORMATION IN THE KG. AUPRC STANDS FOR AREA UNDER PRECISION-RECALL CURVE. AUROC STANDS FOR AREA UNDER ROC CURVE. *ET+KG* STANDS FOR OUR PROPOSED MODEL THAT USES PAST EVENT INFORMATION AND INFORMATION FROM THE KG. *ET* ONLY USES PAST EVENT DATA AND *KG* ONLY USES KG DATA.

	AUPRC	AUROC
ET+KG	0.586 ± 0.0010	0.979 ± 0.0001
ET	0.574 ± 0.0014	0.977 ± 0.0001
KG	0.487 ± 0.0016	0.974 ± 0.0002

B. Recommendation Engines

We used data from the MovieLens project with 943 users and 1682 movies.[4] In the KG tensor we considered the triples *(User, rates, Movie; Rating)*. For the event tensor, we considered the quadruples *(User, watches, Movie, Time; Watched)* and *(User, rates, Movie, Time; Rating)*. Here, *Rating* $\in \{1, \ldots, 5\}$ is the score the user assigned to the movie and *Watched* $\in \{0, 1\}$ indicates if the movie was watched and rated at time t. *Time* is the calendar week of the rating event. We define our training data to be 78176 events in the first 24 calender weeks and the test data to be 2664 events in the last 7 weeks. Note that in both datasets there are only 738 users since the remaining 205 users watched and rated their movies only in the test set.

It turned out that movie ratings depended on past ratings but not on when the rating was done or what movies were watched recently, and thus they could be predicted from the KG model alone with

$$\theta^{predict}_{s,rates,o,t} = f^{predict}_{rates,o}(\mathbf{a}_{e_s}).$$

We obtained best results by modeling the function with a neural network with 1682 outputs (one for each movie). The user specific data was centered w.r.t. to their average and a numerical 0 would stand for a neutral *rating*. We obtain an RMSE score of 0.90 ± 0.002 which is competitive with the best reported score of 0.89 on this data set [25]. But note that we predicted *future ratings* which is more difficult than predicting randomly chosen test ratings, as done in the other

[4] http://grouplens.org/datasets/movielens/

studies. Since we predict ordinal ratings, we used a Gaussian likelihood model.

Of more interest in this paper is to predict if a user will decide to watch a movie at the next time step. We used a prediction model with

$$\theta^{predict}_{s,watches,o,t} = f^{predict}_{watches,o}(\mathbf{a}_{e_s}, \mathbf{a}_{e_t}, \mathbf{a}_{e_{s,t}}, \mathbf{a}_{e_{s,t-1}}, \ldots, \mathbf{a}_{e_{s,t-T}}).$$

Here, $\mathbf{a}_{e_s}$ stands for the profile of the user as represented in the KG. $\mathbf{a}_{e_t}$ stands for the latent representation of all events at time t and can model seasonal preferences for movies. $\mathbf{a}_{e_{s,t}}$ stands for the latent representation of all movies that the user watched at time t. The architecture is shown in Figure 3. When training with only the prediction cost function we observe an AUROC an score of 0.728 ± 0.001. We then explored sharing of statistical strength by optimizing jointly the M-matrices using all three cost functions cost^{KG}, cost^{event} and $\text{cost}^{predict}$ and obtained a significant improvement with an AUROC score of 0.776 ± 0.002.

For comparison, we considered a pure KG-model and achieved an AUROC score of 0.756 ± 0.007. Thus the information on past events leads to a small (but significant) improvement.

C. Sensor Networks

TABLE III
MEAN SQUARED ERROR SCORES FOR PREDICTING MULTIVARIATE SENSOR DATA 20 TIME STEPS AHEAD.

Model	MSE
Pred1	0.135 ± 0.0002
Pred2	0.139 ± 0.0002
Pred3	0.137 ± 0.0002
Feedforward Neural Network	0.140 ± 0.0002
Linear Regression	0.141 ± 0.0001
Last Observed Value	0.170

In our third experiment we wanted to explore if our approach is also applicable to data from sensor networks. The main difference is now that the event tensor becomes a sensor tensor with subsymbolic measurements at all sensors at all times.

Important research issues for wind energy systems concern the accurate wind profile prediction, as it plays an important role in planning and designing of wind farms. Due to the complex intersections among large-scale geometrical parameters such as surface conditions, pressure, temperature, wind speed and wind direction, wind forecasting has been considered a very challenging task. In our analysis we used data from the Automated Surface Observing System (ASOS) units that are operated and controlled cooperatively in the United States by the NWS, FAA and DOD[5]. We downloaded the data from the Iowa Environmental Mesonet (IEM)[6]. The data consists of 18 weather stations (the *Entities*) distributed in the central US, which provide measurements every minute. The measurements we considered are wind strength, wind direction, temperature, air pressure, dew point and visibility coefficient (the *Attributes*).

In the analysis we used data from 5 months from April 2008 to August 2008. The original database consists of 18 tables one for each station.

The event tensor is now a sensor tensor with quadruples *(Station, measurement, SensorType, Time; Value)*, where *Value* is the sensor measurement for sensor *SensorType* at station *Station* at time *Time*. The KG-tensor is a long-term memory and maintains a track record of sensor measurement history.

As the dataset contains missing values we only considered the periods in which the data is complete. This results in a total of 130442 time steps for our dataset. In order to capture important patterns in the data and to reduce noise, we applied moving average smoothing using a Hanning window of 21 time steps. We split the data into train-, validation- and test set. The first four months of the dataset where used for training, and the last month as test set. 5 % of the training data where used for validation.

We considered three different prediction models with Gaussian likelihood functions, each with different latent representations at the input. The first model (Pred1) is

$$\theta^{predict}_{s,p,o,t} = f^{predict}_{p,o}(\mathbf{a}_{e_s}, \mathbf{a}_{e_{s,t-1}}, \mathbf{a}_{e_{s,t-2}}, \ldots, \mathbf{a}_{e_{s,t-T}})$$

where $\mathbf{a}_{e_{s,t}}$ stands for all measurements of station e_s at time t and $\mathbf{a}_{e_{s,t-1}}, \mathbf{a}_{e_{s,t-2}}, \ldots, \mathbf{a}_{e_{s,t-T}}$ can be considered a short term memory. $\mathbf{a}_{e_{s,t-1}}$ represents all measurements for station s between $t-T-1$ and $t-1$, i.e., and can represent complex sensor patterns over a longer period in time. Since measurements take on real values, a Gaussian likelihood model was used.

The second model (Pred2) is

$$\theta^{predict}_{s,p,o,t} = f^{predict}_{p,o}(\mathbf{a}_{e_{s,t-1}}, \ldots, \mathbf{a}_{e_{s,t-T}}, \mathbf{a}_{e_{t-1}}, \ldots, \mathbf{a}_{e_{t-T}}).$$

Here, $\mathbf{a}_{e_t}$ stands the latent representation of all measurements in the complete network at time t.

And finally the third model (Pred3) combines the first two models and uses the combined sets of inputs. The architecture of Pred3 is shown in Figure 4.

In our experiments we considered the task of predicting 20 time steps into the future. All three models performed best with $T = 10$ and the rank of the latent representations being 20. Table III summarizes the results of the three prediction models together with three baseline models. The most basic baseline is to use the last observed value of each time series as a prediction. More enhanced baseline models are linear regression and feedforward neural networks using the previous history $z_{s,:,:,t-1}, z_{s,:,:,t-2}, \ldots, z_{s,:,:,t-T}$ of all time series of a station s as input. The experiments show that all three prediction models outperform the baselines. Pred1, which adds the personalization term for each sensor shows the best results. Pred2 performs only slightly better than the feedforward neural network. However, we assume that in sensor networks with

[5]http://www.nws.noaa.gov/asos/

[6]https://mesonet.agron.iastate.edu/request/asos/1min.phtml

a stronger cross correlation between the sensors, this model might prove its strength. Finally, the result of Pred3 shows that the combination of the multiple latent representations is too complex and does not outperform Pred1.

VII. Conclusions and Extensions

We have introduced an approach for modeling the temporal evolution of knowledge graphs and for the evolution of associated events and signals.The goal is to integrate temporal and static context information for prediction and decision support. We have demonstrated experimentally that models using latent representations perform well in these high-dimensional and highly sparse dynamic domains in a clinical application, a recommendation engine and a sensor network application. The clinical application is explored further in a funded project [35], [12]. As part of future work we plan to test our approach in general streaming frameworks which often contain a context model, an event model and a sensor model, nicely fitting into our framework. In [34] we are exploring links between the presented approach and cognitive memory functions.

In general, we assumed a unique representation for an entity, for example we assume that $\mathbf{a}_{e_s}$ is the same in the prediction model and the semantic model. Sometimes it makes sense to relax that assumption and only assume some form of a coupling. [15], [3], [2] contain extensive discussions on the transfer of latent representations.

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007.

[2] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. *Unsupervised and Transfer Learning Challenges in Machine Learning*, 7:19, 2012.

[3] Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *PAMI*, 2013.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 2003.

[5] Bettina Berendt, Andreas Hotho, and Gerd Stumme. Towards semantic web mining. In *ISWC*. Springer Berlin Heidelberg, 2002.

[6] Tim Berners-Lee. Linked Data - Design Issues, 2006.

[7] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web, 2001.

[8] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[9] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning Structured Embeddings of Knowledge Bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[10] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In *SIGKDD*, 2014.

[11] Daniel M. Dunlavy, Tamara G. Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *TKDD*, 5(2), 2011.

[12] Cristóbal Esteban, Danilo Schmidt, Denis Krompaß, and Volker Tresp. Predicting sequences of clinical events by using a personalized temporal latent embedding model. In *IEEE ICHI*, 2015.

[13] Nicola Fanizzi, Claudia dAmato, and Floriana Esposito. Dl-foil concept learning in description logics. In *Inductive Logic Programming*. Springer, 2008.

[14] Tamara G. Kolda and Brett W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 2009.

[15] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. *AAAI*, 1(2):3, 2008.

[16] Jens Lehmann. Dl-learner: learning concepts in description logics. *The Journal of Machine Learning Research*, 10:2639–2642, 2009.

[17] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *SIGKDD*, 2005.

[18] Gabriela Lindemann, Danilo Schmidt, Thomas Schrader, and Dietmar Keune. The resource description framework (RDF) as a modern structure for medical data. *International Journal of Medical, Health, Biomedical and Pharmaceutical Engineering*, 2007.

[19] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 2001.

[20] Morten Mørup. Applications of tensor (multiway array) factorizations and decompositions in data mining. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 2011.

[21] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proc. of the IEEE*, 2015.

[22] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A Three-Way Model for Collective Learning on Multi-Relational Data. In *ICML*, 2011.

[23] Joshua O'Madadhain, Jon Hutchins, and Padhraic Smyth. Prediction and ranking algorithms for event-based network data. *SIGKDD Explorations*, 7(2), 2005.

[24] Heiko Paulheim, Petar Ristoski, Evgeny Mitichkin, and Christian Bizer. Data mining with background knowledge from the web. *RapidMiner World*, 2014.

[25] Steffen Rendle. Factorization Machines. In *Proceedings of the 10th IEEE International Conference on Data Mining*. IEEE Computer Society, 2010.

[26] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, 2010.

[27] Achim Rettinger, Uta Lösch, Volker Tresp, Claudia d'Amato, and Nicola Fanizzi. Mining the semantic web. *Data Mining and Knowledge Discovery*, 2012.

[28] Ryan A. Rossi, Brian Gallagher, Jennifer Neville, and Keith Henderson. Modeling dynamic behavior in large evolving graphs. In *WSDM*, 2013.

[29] K. Schröter, G. Lindemann, and L. Fritsche. Tbase2 a web-based electronic patient record. *Fundamenta Informaticae*, 2010.

[30] Amit Singhal. Introducing the Knowledge Graph: things, not strings, May 2012.

[31] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *NIPS*, 2013.

[32] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *WWW*, 2007.

[33] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *SIGKDD*, 2007.

[34] Volker Tresp, Cristóbal Esteban, Yinchong Yang, Stephan Baier, and Denis Krompaß. Learning with memory embeddings. *arXiv preprint arXiv:1511.07972*, 2015.

[35] Volker Tresp, Sonja Zillner, Maria J. Costa, Yi Huang, Alexander Cavallaro, and et al. Towards a new science of a clinical data intelligence. In *NIPS Workshop on Machine Learning for Clinical Data Analysis and Healthcare*, 2013.

[36] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SIAM*, 2010.

APPENDIX: COST FUNCTIONS

We consider cost functions for the KG tensor, the event tensor and the prediction model. The tilde notation $\underline{\tilde{X}}$ indicates subsets which correspond to the facts known in training. If only positive facts with *Value* $=$ *True* are known, as often the case in KGs, negative facts can be generated using, e.g., local closed world assumptions [21]. We use negative log-likelihood cost terms. For a Bernoulli likelihood, $-\log P(x|\theta) = \log[1+\exp\{(1-2x)\theta\}]$ (cross-entropy) and for a Gaussian likelihood $-\log P(x|\theta) = const + \frac{1}{2\sigma^2}(x-\theta)^2$. We use regularization as described in Equation 2.

We describe the cost function in terms of the latent representations A and the M-mappings. W stands for the parameters in the functional mapping.

KG: The cost term for the semantic KG model is

$$\text{cost}^{KG} = -\sum_{x_{s,p,o} \in \underline{\tilde{X}}} \log P(x_{s,p,o}|\theta^{KG}_{s,p,o}(A,M,W))$$

Events:

$$\text{cost}^{event} = -\sum_{z_{s,p,o,t} \in \underline{\tilde{Z}}} \log P(z_{s,p,o,t}|\theta^{event}_{s,p,o,t}(A,M,W))$$

Prediction Model: The cost function for the prediction model is

$$\text{cost}^{predict} = -\sum_{z_{s,p,o,t} \in \underline{\tilde{Z}}} \log P(z_{s,p,o,t}|\theta^{predict}_{s,p,o,t}(A,M,W))$$