A Brief Survey on Knowledge Graph Update

Jiaqi Xi
Department of Computer Science
Peking University
im.sisixi@pku.edu.cn

March 30, 2020

Abstract

Knowledge graph is now the mainstream of knowledge representation and storage. Obviously, our knowledge updates at an astounding rate, so the graph should also keep up with those changes. However, knowledge graph update is a relatively new field. It is short of targeted datasets and satisfying models. Hence, in this survey, we look into some research that are relevant to or exploring this field. By comparing them we conclude a better choice for updating task, and try to propose ideas that might overcome difficulties this field is facing.

1 Introduction

Knowledge graphs have become a powerful means for representing the current status of knowledge. We decompose knowledge into two parts: entities and relations. More specifically, a fact in a knowledge graph is basically denoted as a triple *(s,p,o)*, where *s,p,o* stands for subject, predicate, and object. For example, *(US, hasPresident, Trump)* could be a triple of the current knowledge graph. Here, *hasPresident* is the relation between two entities: *US* and *Trump*.

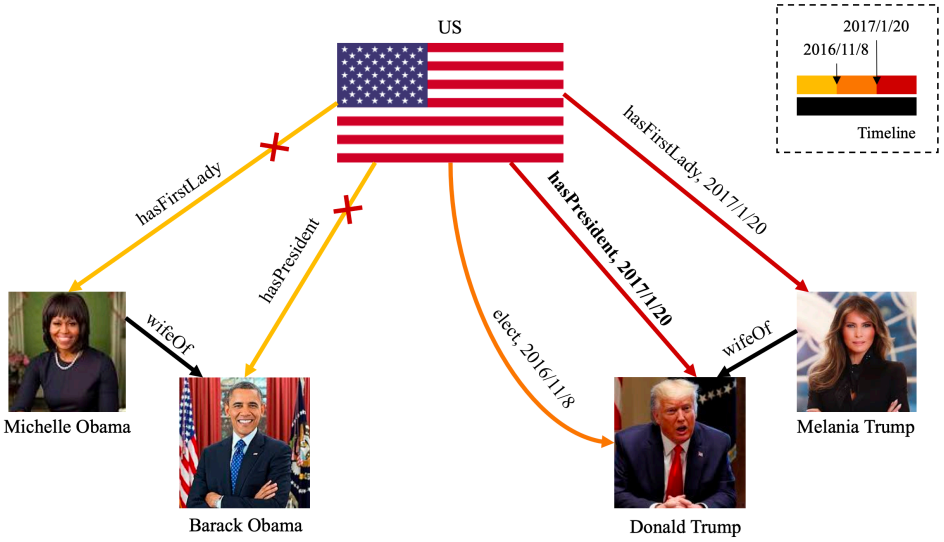

Figure 1: This figure shows a simple dynamic knowledge graph. Some of the triples have time information with them. These triples are generated or deleted following a timeline(from yellow to orange to red). The triple *(US,hasPresident, Trump)* in boldface is usually a given assumption in most research, while other triple changes need to be discovered from it.

However, the president of the United States is not immutable, and Trump was not the president before he won the election in 2016. This means some facts are not indefinitely true. Hence, temporal information should be considered when modifying knowledge graphs. Thus, researchers introduced dynamic knowledge graphs.

Compared with static knowledge graphs, dynamic updates enable the graphs to reflect relation changes of entities. Works on dynamic knowledge graphs deal with time information in two ways:

Table 1: Definitions and examples of Evolution and Update research.

Subfield	Description	Task example in figure 1
Evolution	They try to solve the problem: *What will happen next based on the facts already happened?* Evolution has a sequence of facts as data, and wants the following fact as prediction.	Given *(US, elect, Donald Trump)*, predict the next triple: *(US, hasPresident, Donald Trump)*. Go further and predict *(US, hasFirstLady, Melania Trump)*.
Update	They try to tackle a more realistic problem: *What exactly will happen when a fact take place?* Update learns the relevant changes brought by one given change.	Given the change from *(US, hasPresident, Barack Obama)* to *(US, hasPresident, Donald Trump)*, predict the change from *(US, hasFirstLady, Michelle Obama)* to *(US, hasFirstLady, Melania Trump)*. Note that changes imply the deletion of old triples.

These two subfields both believe that the updating of a knowledge graph follows an evolution rule or causal relationship of facts. Also, due to the difficulty of evaluating the performance of prediction or updating, link prediction is usually their common experiment.(*Link prediction* is a task that fills up incomplete triples based on existing knowledge graph.) Hence, these two subfields both aim to complete a new knowledge graph with the causal relationship they find.

Here, we show some differences between them:

Table 2: Comparison between Evolution and Update research.

Evolution	Update
Focuses on predictions. Could simply be rules that $\{A, B\} \to C$.	**Focuses more on changes**[1]**.** Should be $(A \to A') \to \{(B \to B'), (C \to C')\}$.
Triples: $1+ \to 1$. A series of facts generate one fact.	**Triples:** $1 \to 1+$. One change generates a few more.
Time: In a sequence. We have knowledge graphs of time $1, \cdots, t$, and we want to predict the graph of time $t + 1$.	**Time:** Almost simultaneously. Time is not explicitly involved, but the causal relationship between one change and the implicative changes can be represented as a temporal effect.

In the second section, we will discuss the comparison between different methods from these two subfields. We will first learn how temporal information help relate the facts into an ordered sequence. As for "time", if we can shorten the duration of evolution, then the prediction task is quite similar to updating task. From my perspective, models that learn temporal dependencies between facts are more potential that those that merely add time to representations.

Besides, as discussed before, a common experiment is link prediction. This experiment indicates that sequences of facts and related changes often happen among triples that share subjects, relations and, objects. So we will show how context of a triple can help determine updates.

We have mentioned that it is difficult to evaluate the performance of updating tasks. There are more challenges like datasets, problem definition, accuracy, etc. So in the third section, we will explain the difficulties in research on knowledge graph updates and propose some rough ideas about future works.

[1]Here, a change means: When the subject, relation, or object of a triple is changed, a partially new triple is generated while the old one is deleted. Updating task is to find some other influenced triples that undergo changes.

2 Comparison

Updating Knowledge graphs today still requires manual efforts. Intuitively, humans learn logical rules between relations. For instance, we know that people become single when they get divorced, and they no longer have mother-in-law or father-in-law. But it is hard for a computer to learn such rules.

To update a knowledge graph, the computer needs to learn how one fact affects another. There are some patterns that we can follow.

We have mentioned before that in most cases, the related triples share entities or relations. This phenomenon helps determine a scope of potential changing triples from the context. Besides, the changes usually happen either simultaneously(in a short period of time, update way) or in a sequence(evolution way). Hence, context and time are two key factors we should attend to. Many methods have been proposed to apply these two patterns to updating knowledge graphs.

2.1 Time Information

Traditional triples do not have temporal information. Many works try to add time to representations of facts: [2] manually extend the predicate sequence by setting *temporal tokens* for numbers indicating time and setting *predicate tokens* for words like "since", "until"; [8] introduces a time dimension to represent each fact in a quadruplet (e^s, r, e^o, t); [5] focuses on procedural texts that implicitly incorporate time in its contexts. Given a target entity, the model finds its location change as time passes by. Learning this state change is similar to learning a continually changing(updating) fact; [6] views time as a pronoun of a related fact: if A evokes B then A is called the start time of B. Most works use LSTM or other recurrent networks to learn temporal information in representations.

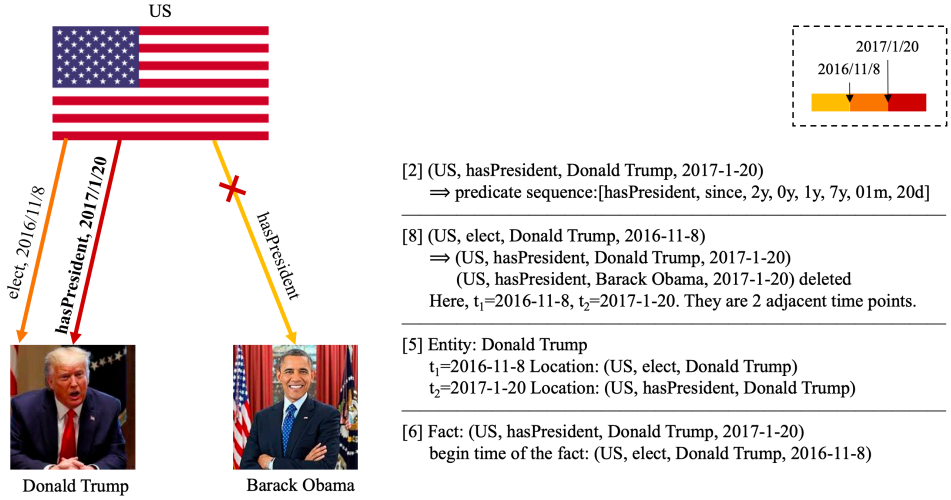

Figure 2: This figure shows examples of how these four methods treat temporal information in knowledge graphs or text contexts differently.

From my perspective, [2, 8] simply add time to the representations of either predicate embedding or triples. They are based on one assumption that time is clearly written, or that we know the exact location of time information in the texts. Thus, these methods are more helpful for completion of a knowledge graph given structured data.

While in most realistic cases, time is implicit information. For example, we know that *bornIn* happens before *dieIn* even if we do not know the exact time. Note that our goal is to update knowledge graphs by finding relations between facts. Time information is one of the means for extracting these updating rules. Hence, [5, 6] shows a more generalized understanding of time information. They both focus on one entity or one fact, and time information can be relative time points as long as it shows a state change.

Works that treat time as a relative quantity and focus on underlying dependency between facts tend to be more adaptive to real world updating tasks. Finding dependency between facts is also important in the upcoming discussion about context.

2.2 Context

Another important factor for updating is context. Here, context can be the context of an entity in a paragraph or the surroundings of a triple in the knowledge graph.

[5, 6] understand context in the former way.

The procedural texts [5] concerns are mainly about biology and chemistry. Several changes happen in one paragraph, which highly concentrates the update work both in time and scale. Since the whole process is included, context is essential for LSTM to learn changes of relations between several entities.

[6] aggregates state vectors for entities undergoing a state change. Context offers two information: (1) What is the state at time *t*? (2) How is the state changed compared with time *t-1*? This model uses seed entities to learn the scope of context where state change happens.

> *A Republican, Trump was a* businessman and reality television personality from New York City at the time of his 2016 presidential election victory over Democratic nominee Hillary Clinton. The *presidency of Donald Trump began at* noon EST(17:00 UTC) on January 20, 2017, when Donald Trump was inaugurated as the 45th president of the United States, succeeding Barack Obama.

Figure 3: This paragraph[2] shows examples of how the methods treat context information. Entities and relations are colored differently in the paragraph. Given the entity *Trump*, [5] extracts location of the fact(underlined in the paragraph), and learn the state change. [6] uses multi-gram method like the italic words here to represent context of an entity.

[1, 3, 4] understand context in the latter way.

[3] imitates TransE in learning sequences between two facts. Suppose that $r_i M \approx r_j$, here r_j is in the context of r_i, and the chronological order between these two relations is depicted by matrix M. The network is based on the assumption that if given two new relations whose chronological order is unchanged, then M between new relations is almost the same.

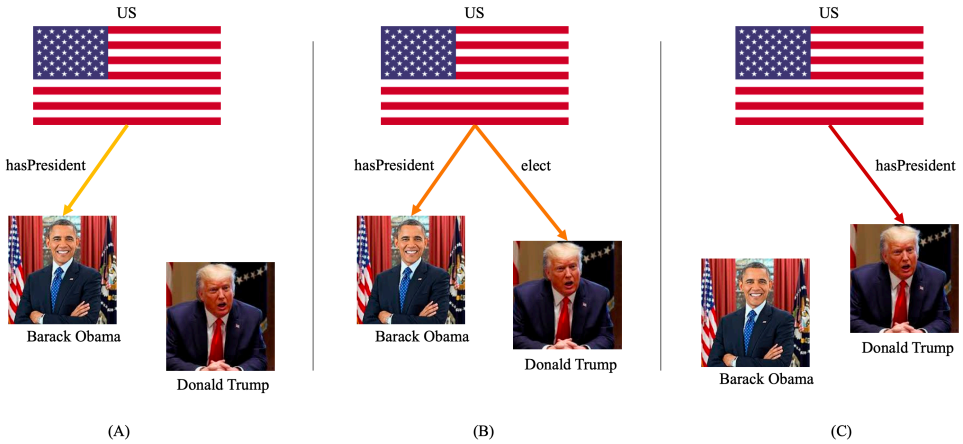

Figure 4: In this updating task, [3] contends that *(US, hasPresident, Donald Trump)* is a fact that can be reached by multiplying a learned matrix M to *(US, elect, Donald Trump)*. Thus, a rule of change from *(s, elect, o)* to *s, hasPresident, o* is learned. [1, 4] considers the knowledge graph as a distribution map and concerns about topological features of the knowledge graph. Two context features are worth noticing: (1) Entity *Donald Trump* is moving closer to *US* as their relation changes; (2) When *(US, hasPresident, Donald Trump)* is generated, triple *(US, hasPresident, Barack Obama)* with the same subject and relation is facing deletion.

[4] introduces a multi-relational aggregator. Given one entity, it aggregates information from multi-relations in multi-hop neighbors. [1] also uses neighbors to capture context in a knowledge graph. Furthermore, it uses an encoder-

[2]This paragraph is abstracted from Wikipedia: *Presidency of Donald Trump*.

decoder network. By assigning new positions to entities according to the text(encoder), links between entities are added or deleted due to distance change(decoder).

The key is how we depict and learn "relation between relations". We can use a change vector, a matrix, a distribution map, etc. They all seem capable of representing relation, but we should look further to judge which one is better. Most works are still restricted in extracting a state change given by the text, few like [1] concerns about the updates that should be done on the knowledge graphs but not explicitly written. For example, when a basketball player changes his team, his team member should be changed on the knowledge graph, but it is not explicitly given by any text.

How can we handle the implicit changes? We need to learn rules of changes, like "if the subject of r_i changes from s_i to s_i', then the subject of r_j is likely to change from s_j to s_j'". Such rules are statistically true, and often occur in near neighbors. So we need a representation that leverages the knowledge graph structure and can handle multi-relations. However, vectors and matrices are often used to depict relation between two relations. Hence, it is more possible to find the interaction between facts if we use a distribution map instead of vectors or matrices.

2.3 Learning goals

Time information and context are concrete details that researchers look into to learn updating rules. Here, we head into a bigger picture to analyze similarities and differences in learning goals. We separate the models into three categories:

Table 3: Three main kinds of models solving updating tasks.

	Distribution Map Updating[1, 4]	Sequence Representation[2, 5, 7, 8]	Temporal Relation between Two Triples[3, 6]
Model	**Encoder-Decoder** Encoder uses attention of potential changing triples to correctly place them in a new knowledge graph. Attention is learned according to text context and knowledge graph context. Decoder completes the knowledge graph reconstructed by encoder.	**LSTM with time-encoding sequences** The encoding methods of time information have already been introduced above.	**TransE or Vector** Learn a matrix or vector to denote a state change. As a result, to detect whether an exact state change happens, apply the learned matrix or vector to the new context.
Attention	1 Hop-subgraph: normalized adjacent matrix for relation r; Text-subgraph: text context attention vector	Temporal tokens; leverage MRC mechanisms to extract text spans; use soft co-reference to tract different forms of the same entity.	Some form of subtraction: $state_t - state_{t-1}$. In detection stage, test the similarity between seeds for state change and the new data.
Task	Known: *(B, playerof, T_2)*; *(A, playerof, T_1) → (A, playerof, T_2)*; To learn: *(A, teammateof, T_2)*	Known: t_1: *(US, hasPresident, Barack Obama)*; t_2: *(US, elect, Donald Trump)*; To learn: t_3: *(US, hasPresident, Donald Trump)*, delete *(US, hasPresident, Barack Obama)*	Known: *(US, elect, Donald Trump) → (US, hasPresident, Donald Trump)* To learn: M that satisfies $elect \cdot M = hasPresident$

This table concludes three kinds of current methods dealing with updating task. As is discussed before, "Distribution Map Updating" method understands time information and context in a most intuitive way, just like how human beings update a knowledge graph. Meanwhile, it faces the most challenges among these three methods, and is least explored either.

As for "Sequence Representation" and "Temporal Relation between Two Triples", the former is adept at relating a series of events by time, but lacks some ability to deal with implicit changes; the latter has a narrow view on relations between triples, and might fail to solve multi-relation problems as data is accumulating.

Hence, by comparison we have found a relatively better way "Distribution Map Updating" to make full use of information to tackle the updating task. Nevertheless, research on this method is facing great difficulties and strenuously making progress. Thus, in the next section, we will discuss about the exact obstacles and propose some ideas about what we can try next.

3 Ideas

In this section, we talk about the difficulties that result in the slow progress of knowledge graph updating. Afterwards, we will provide suggestions that try to reduce some of the difficulties and go further.

3.1 Difficulties

Knowledge Graph Update is a relatively new subfield. Newborn subfields tend to have some problems in common: the lack of rigorous definition of the task and the lack of specific datasets. Besides, there is still not a common idea of what a state-of-the-art method or model in this subfield looks like. The following explanation will show some more details about the difficulties and how they affect this subfield.

3.1.1 Definition

We intuitively define updating task as: when an event happens(explicit), all the related triples(implicit) are either created or deleted. Most updating works sound common sense to humans, but even common sense is achieved by some kind of learning. Thus, we need to find a more concrete definition for computers.

As for research on knowledge graphs, it took some time for researchers to consider time information. Similarly, moving from temporal knowledge graphs to updating knowledge graphs requires: (1) compressing a duration into a simultaneous case; (2) finding implicit updates that happen at the same time.

How can we define a "simultaneous" update for a computer? In fact, the computer is given an explicitly changed triple from the text, and its task is to find other implicitly changing triples. Though all the changes seem to happen simultaneously to humans, it is deemed as an A cause B case by a computer. Such a causal relation is why we introduce research on temporary knowledge graphs to help understand updating task.

3.1.2 Dataset

We have mentioned that most research use link prediction tasks to evaluate their performance. Completing a triple is similar to completing a potential triple that needs change.

However, we still need specific datasets that can satisfy the definition of updating task. Change is important here. Most link prediction tasks are simply triples related by time, but those triples may not undergo changes. Also, as is said in definition part, we need to separate triples into explicit and implicit parts.

Furthermore, most models currently focus on a sequence of facts or relation between two facts. Hence, we lack some multi-relation data, in which one triple change leads to a bunch of changes. Such one-to-many data is still missing.

Besides, the size of data remains a problem. People learn rules by logic or few trials, but networks usually need lots of data. Some rules do not even have enough data. Hence, feeding a model too many data will undermine its practical value. How to balance the value and performance of model is one of the concerns when constructing a dataset.

3.1.3 Model

Though we introduced some research on temporal knowledge graphs and evolution, the LSTM method that fits sequence data might not work well for updating tasks. The way they represent time and context is still instructive, but we need a more proper model.

Encoder-decoder method is a reasonable trial. But we still face restrictions: we can only learn one rule at a time. Would we be able to construct a model that can react to different updating tasks on its own? It is like the model has learned how to learn a common sense of updating with few data given.

3.2 Suggestions

A few rough ideas are proposed below to either reduce some difficulties or help designing the model:

1. Although I said that updating task can be defined as a causal relation so that we could introduce the conception of time, simultaneous changes still have advantages. If change C_1 and change C_2 happen simultaneously, we can say $C_1 \rightarrow C_2$ as well as $C_2 \rightarrow C_1$.

   So if we have a bunch of changes that will happen at the same time, we might use GAN-like methods to train several models. We take each change as the cause and generate a bunch of models. They output the triples

changing at the same time. These models check whether they share the same bunch of changing triples. Thus, they could evaluate their performance and collectively optimize themselves.

Here, we should carefully choose the "cause" change, since not all changes can lead to the others. For example, when *(A, teammateof, B)* is generated, it could be *A* or *B* or both of them that belong to a new team.

2. We could refer to tasks like few-shot learning. It is fact that some rules do not have enough data for normal supervised learning.

   For example, methods like separating data into a sequence of small parts(e.g. changes happen one by one) by force enable us to make full use of limited data and selectively memorize the important ones.

3. Regarding the generality of updating model, combining meta learning with probabilistic methods might be one way.

   Use meta learning for basic rules of how one change affects another. And use probabilistic methods to specifically deal with secondary rules or special cases.

   Actually [1] is similar to this idea, instead of the fact that it replaces meta learning part with the assumption that changes happen in 1 hop subgraph.

From my perspective, the final goal of knowledge graph update is to imitate how humans finish this task. Considering the errors that almost always occur, I contend that errors are reasonable and affordable, while it is more meaningful if we could concern more about the brain-like construction of models regarding updating task.

References

[1] Jizhi Tang, Yansong Feng, and Dongyan Zhao. Learning to Update Knowledge Graphs by Reading News. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2632–2641.

[2] Alberto García-Durán, Sebastijan Dumani, and Mathias Niepert. Learning Sequence Encoders for Temporal Knowledge Graph Completion. In *arXiv preprint arXiv:1809.03202*, 2018.

[3] Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. Encoding Temporal Information for Time-Aware Link Prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2350–2354.

[4] Woojeong Jin, He Jiang, Meng Qu, Tong Chen, Changlin Zhang, Pedro Szekely, and Xiang Ren. Recurrent Event Network: Global Structure Inference Over Temporal Knowledge Graph. In *CoRR*, abs/1904.05530, 2019.

[5] Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. Building Dynamic Knowledge Graphs from Text using Machine Reading Comprehension. In *arXiv preprint arXiv:1810.05682*. ICLR 2019.

[6] Derry Tanti Wijaya, Ndapandula Nakashole, and Tom M. Mitchell. CTPs: Contextual Temporal Profiles for Time Scoping Facts using State Change Detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1930–1936.

[7] Cristóbal Esteban, Volker Tresp, Yinchong Yang, Stephan Baier, and Denis KrompaSS. Predicting the Co-Evolution of Event and Knowledge Graphs. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 98-105.

[8] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs. In *ICML*, 2017.

[9] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A Survey on Knowledge Graphs: Representation, Acquisition and Applications. In *arXiv:2002.00388*, 2020.