The Australian National University
2600 ACT | Canberra | Australia

Australian National University

School of Computing

College of Engineering and Computer Science (CECS)

Temporal Knowledge Graph Prediction

— 12 pt Honours project (S2/S1 2021–2022)

A thesis submitted for the degree
Master of Machine Learning and Computer Vision

By:
Peng Zhang

Supervisors:
Dr. Pouya.Ghiasnezhad.Omran
Prof. Kerry Taylor

October 2022

Declaration:

I declare that this work:

- upholds the principles of academic integrity, as defined in the University Academic Misconduct Rules;
- is original, except where collaboration (for example group work) has been authorised in writing by the course convener in the class summary and/or Wattle site;
- is produced for the purposes of this assessment task and has not been submitted for assessment in any other context, except where authorised in writing by the course convener;
- gives appropriate acknowledgement of the ideas, scholarship and intellectual property of others insofar as these have been used;
- in no part involves copying, cheating, collusion, fabrication, plagiarism or recycling.

October, Peng Zhang

Acknowledgements

First of all, I would like to express my thanks to my supervisor **Pouya.G.Omran** and **Kerry Taylor** for trusting me and providing support to this project. I am so grateful for their guidance, advice, and timely reaction to my query. This project may be hard to accomplish without the sustained and powerful supporting from Kerry and Pouya.

Beside, I want to thanks my parents for supporting me unconditionally all the way through my degree.

Finally, I really appreciate the support from relatives and friends below.

Xiyu Wang @ Shanghai Academy of Social Sciences

Hang Zhang, Jiahao Zhang, Shixuan Liu, Xinran Tian @ ANU

Fengyi Zhang @ University of Illinois at Urbana-Champaign

Abstract

Knowledge graph became a very important structure for storing information, machine learning task has been designed for predicting the missing or upcoming event(information). With the development of traditional knowledge graph, temporal knowledge graph appears for the need of recording temporal information of event.

Our task is set with the topic doing prediction upon temporal knowledge graph, more specifically, we using deep learning method to predict the entity information in temporal knowledge graph

In this project report, we go through technics involved with related topics, give our own design and attempts. After embedding our rule-based reinforcement learning framework, we indeed achieve little improvement in comparison with the baseline model and finds some potential work in rule-based method.

Table of Contents

Chapter 1

Introduction

Knowledge Graphs (KGs) are widely used for many mainstream Artificial Intelligence (AI) applications, such as recommendation systems, dialogue generation, and question answering. However, facts constantly change over time. To reflect the timeliness of facts, Temporal Knowledge Graphs were raised, so that we could additionally associate each triple with a timestamp.

Machine learning upon knowledge graph has many potential values in various industries, like we could predict the upcoming risk in financial market from current market situation, potential critical illness based on one man's health condition and personal medical history and etc. In traditional knowledge graph prediction task, the methods have been researched completely, including the initial translation models, rule reasoning models, language related models and etc. With the appearance of temporal knowledge graph, more complex models for temporal knowledge graph prediction task are being researched.

In current market, knowledge graph modeling also has potential in developing different angle for giving prediction and warning to upcoming risk and crisis in stock market or some specific areas.

Our target is firstly going through and having a command of mainstream methods in this field, then searching more potential, making some modification of my own upon the baseline model.

Chapter 2

Related Work

This chapter reviews the work that is most related to the research questions investigated by our work.

2.1 Background

In this chapter I will go through the content about some basic knowledge including data, models and some outstanding technics in this field.

2.1.1 Knowledge Graph

The concept of knowledge graph was first proposed by Google Singhal (2012), a structured semantic knowledge base used to symbolically describe concepts and their relationships in the real world. It is somehow close to the concept of semantic network in terms of representation, but semantic network prefers to describe the relationship between concepts, while knowledge graph prefers the association between entities.

The nodes of the knowledge graph represent entities, and the edges represent the relationship between two entities, where entities could be store as an attribute item, and in this case the edges between these two entities should also be called attribute edge.

General knowledge graphs are stored in the RDF (Resource Description Framework) data format. The basic units of the RDF (Resource Description Framework) are known as SPO triples, with S representing the Subject. P means Predicate, O means Object. A knowledge graph usually contains more than a million triples, and a triplet represents a piece of knowledge. The knowledge "Jay Chou - wife - Hannah Quinlivan" is represented by the triplet as $\langle JayChou, wife, HannahQuinlivan\rangle$. Knowledge graphs can be categorized into two types according to how they are constructed. One is the vertical domain knowledge graphs such like DBpedia. They construct triples by including a large number of entities and relations from knowledge in official introduction and explanation. We could simply trust sources where those information came from. But encyclopedic knowledge is just the tip of the iceberg when compared to the whole Internet. Another is Open Extraction knowledge graph, which is based on OpenIE(Open Information Extraction), NELL(never-ending Language Learning) and other technologies. Triples are extracted from web pages on the whole Internet. Compared with the vertical domain knowledge graph, the knowledge of this kind of knowledge graph is richer and more diverse, but it is more noisy, and the description of entities and relations is biased towards the form of natural language.

2.1.2 Temporal KG

Most of the current knowledge graph studies focus on static KG, which does not record the development of things over time, while the time-varying KG is less explored. But temporal information is important, because most structured knowledge is only valid for a specific interval, and facts vary according to a time series. Recent studies have begun to integrate temporal information into KRL (knowledge representation learning) and KGC (kowledge graph completion), which is called temporal knowledge graph.

2.1.3 Temporal KG prediction

Categories

TKG completion tasks are generally divided into two types:

- **Interpolation**: complete the missing facts in the middle, like knowing facts in t_1 and t_3, outputs the facts in t_2.
- **Extrapolation**: predict the upcoming events in future, which means having facts before t_3, forecast information in and after t_3.

Problems

At the time we are considering this topic, most of existing method is interpolation type, compared with the interpolation task, there are two challenges for extrapolation. (1) Unseen timestamps: the timestamps of facts to be forecast do not exist in the training set. (2) Unseen entities: new entities may emerge over time, and the facts to be predicted may contain previously unseen entities. For existing methods doing extrapolation task, RE-Net and CyG-Net are representing the previously unseen entities as random vectors and model the link prediction task as a multi-class classification task, so actually the difficulty (2) has not been solved, and also they did not let the history explicitly show its impact upon predicted results

2.2 Traditional(static) KG link prediction

With the beginning of knowledge graph development, people attempt to do link prediction upon static KG which is represented in set of triplets. Methods in this field are relatively complete, so in this chapter we may go through numbers of representative works for getting some inspiration and basic knowledge.

2.2.1 TransE

TransE Bordes et al. (2013) is a good representative of translation model for tradition knowledge graph prediction. Generally speaking, in this type of models we pay attention on generating embedding for entity and relation. In transE, the author simplely want the embeddings of a triplet (head,relation, tail) could fit the ideal relation: $head+relation \approx tail$, so the loss is designed as follow:

$$||head + relation - tail|| \tag{2.1}$$

Training process

The steps of training TransE model are as follow:

- Initialize entities and relations with uniform distribution
- Within each epoch, split batch and initialize set of pairs of triplets
- Loop samples within batch, generate corrupted triplets
- Train target with gradient descent and update 3 vectors.

TransE is only one representative one among a group of translation model.As we could foresee, it has its own drawbacks, the most obvious one is that 'TransE' could not deal with invertible relation, for example, (A, is_friend_of, B). In this case, we could easily tell that (B, is_friend_of, A) also holds, but these two facts can not both make sense in 'TransE', which is represented in $A + B = c$ and $C + B = A$.

So to improve or make optimization upon each other's drawback, a family of translation models have been published, here we draw a table to conclude their properties. Going through these models helps us get to know how we could make use of representation learning.

model	idea	drawback
TransE	addition	invertible
TransR Lin et al. (2015)	projection	composition
TransD Ji et al. (2015a)	two space	composition
TransHJi et al. (2015b)	hyperplane	

2.3 Rule_leaning based

Due to some properties of knowledge graph, including topological structure, triplet store format, so there grows a huge potential of developing rule-based method upon prediction task. So, in this part we introduce a rule-based method in static knowledge graph prediction.

2.3.1 Rule_Guider

Background

At the preliminaries part of the paper, the Lei et al. (2020) author introduced two mainly types of existing methods, one is **walk-based** another is **Symbolic-based**

Idea

Basically,RuleGuider consists of a symbolic-based method, referred to as rule miner, and a walk-based method, referred to as agent. And the former one is for mining new rules and another for walking through the graph to learn probability distribution.

This model also brough up the policy network which implemented with LSTM to record the history relation, which is similar to the mechanism in Time-traveler Sun et al. (2021).

Compared to the state of art performance results upon the other two datasets, this model reached a competitive results(not that good) the relation space in FB15k-237 is much larger and the rules is relatively sparse in the large relational path space, which makes it harder for the relation agent to select a desired rule.

2.4 Recurrent Event Network Jin et al. (2020)

2.4.1 Recurrent Event Encoder

To parameterize the probability of each event, the Jin et al. (2020)author introduces local representation and global representation.Global representation includes information from entire graph before time t which could somehow reflects the global preference of upcoming events. Local presentation focus more on entity and relation, it captures the correlation between those relations and entities.The global and local representations work in complementation so that we can model the graph generation process in a more efficient way.

These two types of representation are updated and learned within a recurrent model RNN. While this process, 3 types of aggregators are used.

2.4.2 Aggregators

The authors introduceed two simple aggregation functions: one is simply a mean pooling aggregator and another is a novel mechanism called 'attentive pooling aggregator' .

These two aggregators module only collect neighboring entities within the same relation r. Besides they present another novel one which is called multi-relational aggregator.

- **Mean aggregator**: simply takes the element-wise mean of representations
- **Attentive aggregator**:This method could distinguish the importance of each different entity-relation pair. The weight is trainable, it can determine how relevant each object entity is to the subject and the relation.
- **Multi-Relational Graph (RGCN) Aggregator.**This is a general aggregator that can make use of information from multi-relational and multi-hop neighbors

2.4.3 Parameter Learning via Event Predictions

Predicting object given (s,r) and predicting relation given(s,o) are being regarded as a multi-class classification task in this model. There are 3 terms in their designed loss, and using weight parameter to control the importance of each sub loss term.

2.4.4 Multi-step Inference over Time

They model predicting upcoming events as inferring the conditional probability.

Besides, author also mentioned, though performance could be improved by generating multiple sample graphs, they still use one-sample idea to achieve a better efficiency. Based on these process, the model could further predict upcoming events.

2.4.5 summary

This paper raised a novel autoregressive architecture neural network, making use of past temporal sequences. To be specific, this model uses a recurrent event encoder to encode the past events and use a neighborhood aggregator to model the relation between facts within same timestamp t, so that future event could be inferred in sequential form.

2.5 Reasoning like human

The Wan et al. (2021) author did experiment for visualizing entity embeddings of different relation cluster. Illustration of PCA result shows that a relation may have multiple unknown semantics.

Highlight

- Come up with id using an agent model to travel across entities to achieve multi-hop infer.
- Multi-hop leads to a path, history encoding work should be taken
- Using markove random field for action selection.

- Reinforcement learning framework

2.6 Know-Evolve

Deep evolutionary knowledge network that learns non-linearly evolving entity representations over time. This paper Trivedi et al. (2017) introduces the use of powerful mathematical tool of temporal point process framework for temporal reasoning over dynamically evolving knowledge graphs. This paper also shows the whole evolutionary knowledge network architecture and intensity computation over a timeline.

The architecture of model process showed in this paper is based on ordinary RNN, the authors said other advanced units like GRU and LSTM could also be used.Correspondingly, a well-demonstrated training procedure is given with pesudo-code of algorithm for survival loss computation.Finaly, the authors give both link prediction and time prediction experiments regarding TransE, TransR, Rescal and so forth as competitors, within ICEWS and GDELT datasets.

2.7 DyRep

In this paper, the Trivedi et al. (2019) author brought up a way describing 2 types of changes, association -dynamics of the network (realized as topological evolution) and communication - dynamics on the network (realized as activities between nodes).

And they regard their task as modeling a latent mediation process that bridges the above 2 processes. To achive this, they design a two-time scale temporal point process model of observed processes and parameterize it with an inductive representation.

The algorithm is also detailedly given generating the adjacency matrix A for graph Gt at time t and initializing, updating stochastic matrix S capturing the strength between pair of vertices at time t.

Also, the authors say attention layer in their model also shares motivation in 'graph attention network' and 'gated attention network' in the spirit of applying non-uniform attention over neighborhood.

Beside, author did tSNE van der Maaten and Hinton (2008) dim-reduce work too see the illustration of embedding of both DyRep and GraphSage Hamilton et al. (2017) to show that DyRep embeddings have more discriminative power as it can effectively capture the distinctive and evolving structural features over time as aligned with empirical evidence.

Chapter 3

Our rule-based reinforcement Framework

In this main content section, we will detailly go through our solution including problem formatting, dataset, model structure, training process design and experiment results.

3.1 Dataset

There are several famous TKG dataset in current field of TKG prediction task, such like Integrated crisis early warning system (ICEWS) which combines a database of political events, a system that uses these facts to provide early conflict Warning. It is supported by Defense Advanced The United States Research Projects Agency.

Dozens of papers related to link prediction or temporal graph modeling have done experiment upon this dataset or sub dataset like 'ICEWS-14'. To better compare our experiment results with other models we also take this dataset as main benchmark in our experiments.

dataset	entity num	relation num
ICEWS14	7128	230

3.2 Inspiration from review

In this research project, we have went through numbers of paper related to both KG prediction and TKG prediction Trivedi et al. (2019),Trivedi et al. (2017),Wan et al. (2021), Jin et al. (2020),Bordes et al. (2013). So at this stage before introduction of modeling, we give a summary on what we have learned from those methods and models that we have reviewed.

3.2.1 entity, relation embedding

From the very beginning of the research we have already done review about many representation learning works for KG prediction task. On one hand, we could simply regard embedding learning as to learn a soft projection method so that the rule and pattern could be expressed well in a new embedding space.

On the other hand, the knowledge graph itself if developed from semantic web, so from view of mature NLP feild, to improve embedding learning could help digging out more potential semantic information inside the enrity or relation.

In our baseline model TimeTravelor, the author takes different policy for generating embedding of entities and relations. For relation, we directly use static embeddings for each relation. But things might be different for entities, because the attributes or the connotation of an entity may change with time passing, we define a policy for generating a dynamic(time sensitive) embedding for entity.

3.2.2 multi-hop reasoning(prediction)

According to methods we reviewed, the core idea of make prediction step by step had been mentioned many times including keyword like 'multi-hop','history encoding', 'agent', 'walk-based' method. Some of them like **Rule Guider** is built upon static knowledge graph prediction task, some like RE-net are designed for temporal modeling. Compared with normal prediction task, using multi-step inference extend the range of searching correct destination, in multi-step prediction implemented in our baseline model, we do not urge the agent to reach a new state for each step, we keep the right of agent to stay put, meanwhile the possibility of searching somewhere new is kept well.

Another part in multi-hop prediction is when the agent take step by step, we want the agent to be aware of the history that it has been through. So, to model the history information, we introduce the history encoding method which is also adopted in many works mentioned above. In our baseline model, the Time_Traveler uses a LSTM cell for enocoding the history (previours action).

3.2.3 Reinforcement Learning

Some of model mentioned in previous section such like Rule_Guider and RARL are using Reinforcement learning framework in their training procedure design.

3.3 Task modeling

3.3.1 Notation

Firstly, what we are doing here is an extrapolation task which is a sub-type of knowledge graph prediction, then we state the symbol used in our following theoretical sections.

Use $\mathcal{E}$ to denote the sets of entity,similarly, $\mathcal{R}$ for relations,$\mathcal{T}$ for timestamps and $\mathcal{F}$ for facts. In traditional knowledge graph, information is carried within triples, however, in temporal knowledge graph, we add one timestamp dimension extra, in other words, facts are saved within a format like (e_S, r, e_o, t) which stands for $(subject, relation, object, timestamp)$, here the relation could be regarded as a directed edge from subject to object.1

Also, we could regard TKG as a set or series of static KGs, we call all components of knowledge graph within timestamp t a snapshot, denoted by $\mathcal{G}$ in following content. Due to the need that we talk about entities within different snapshot, so we add extra time sign in expression of entity, like e_i^t, e_o^t, e_s^t

3.3.2 what we predict

As we mentioned, there are 2 types of TKG prediction in current field, one is about filling the missing information, another is predicting the upcoming events like forecasting. Our task is the later one

To put it in a more specific way, we are given querys like $(e_S, r, ?, t)$ or$(?, r, e_O, t)$, and what we have is all the knowledge grph snapshots before time 't'. Our goal is to build up a model to find the e_o^t or e_s^t using previous knowledge.

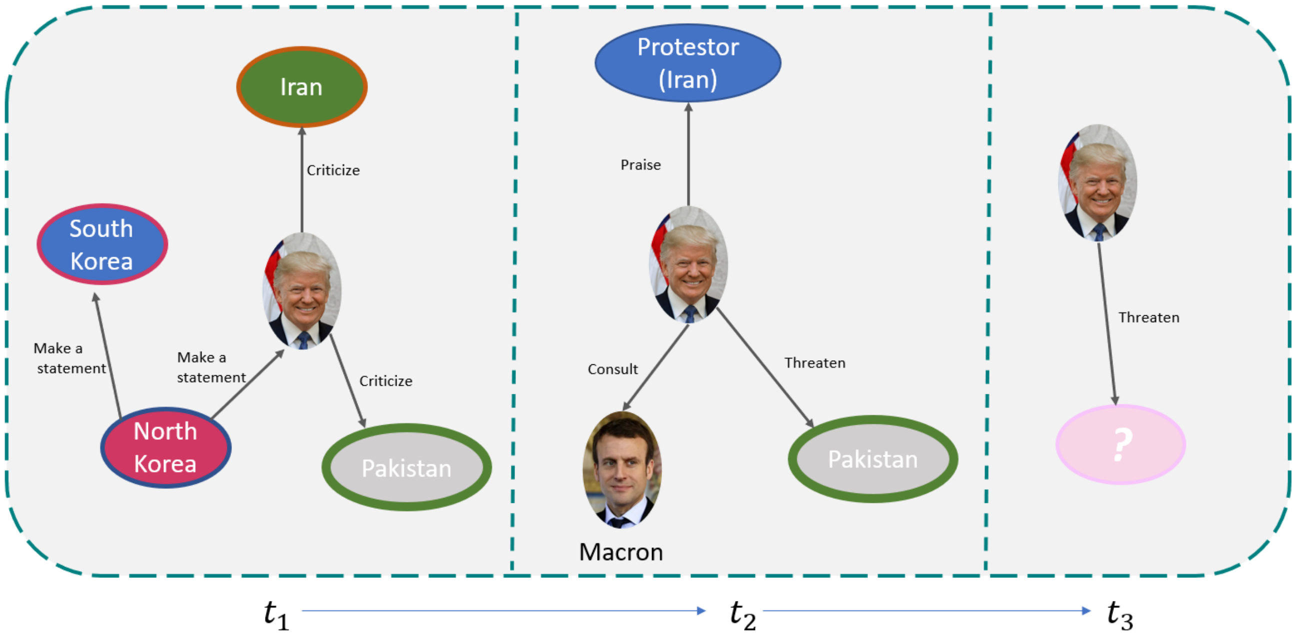

Figure 3.1: in different snapshot, there are different entities connected to Donald Trump with different edge(relation), we use the information before time t_3 to predict who would be threatened by Donald Trump in t_3 snapshot

3.3.3 Framework

Temporal edge

The first challenge that we met is that the relation is a concept that is well-defined within static KG scope, so that the 'agent' could transfer from entity to entity. However, when we are faced with TKG, we need it possible that our agent transfer between different snapshot. So we need to modeling this kind of 'temporal edge'.

Therefore, 3 types of edges are introduced in this method.

Reverse edge

As we introduced above, we have two types of query, one is for subject entity, another for object entity. So, to unify our solution, we create inverse edge for all edge(relation) in target dataset. For example, we could change the query (?, is_wife_of, b,t) into (b, is_husband_of, ?,t). We need to know that not every relation could have an interpretable inverse version, here we doing so is just for modeling.

Self-loop edge

We do prediction or candidates selection within path_length times and take steps, but we do not necessarily transfer to another entity, we should leave the possibility of staying and holding for our 'agent'. So self-loop edge is define for this purpose.

Actions

In our following description, we use $\mathcal{A}$ to denote action space, so $\mathcal{A}_1$ means the candidate action when taking step 1. More specifically, $\mathcal{A}_1$ is sampled from all outgoing edges which starts from node e_l and before the query time stamp t_q.

to state in formula,

candidate action (r^*, e^*, t^*) is sampled from:

$$\{(r^*, e^*, t^*) | (e_x, r^*, e^*, t^*) \in \mathcal{F}, t^* < t_q, t^* \leq t_l\}$$

Here $\mathcal{F}$ stands for all quadruples in training set.

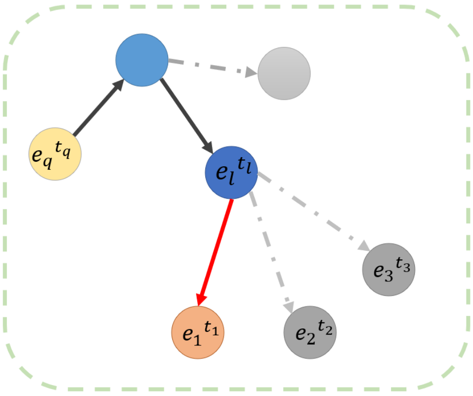

Figure 3.2: Actions: the agent goes from the yellow node as beginning, at each step the agent(e_q) firstly samples actions then score and determine 1, here shows when path-length = 3 and the orange node would be the final answer

3.4 Forward Model Structure(Policy Network)

3.4.1 Brief overview

Take a query for deriving object entity as example, it could be expressed as $(e_q, r_q, ?, t_q)$, in our forward model, our agent travel from $e_q^{t_q}$ as initial state. Then within each loop among path length, the model sample one edge and arrive an entity, the **path length** is a hyper-parameter in our model, after fixed steps of traversing, we take the final destination as the answer to the query.

This following illustration shows the forward path of the policy network, showing how an action being selected from candidates in action space. However this is not complete part of our solution, another important part is the reinforcement learning(Reward mechanism) which is well explained in next section.

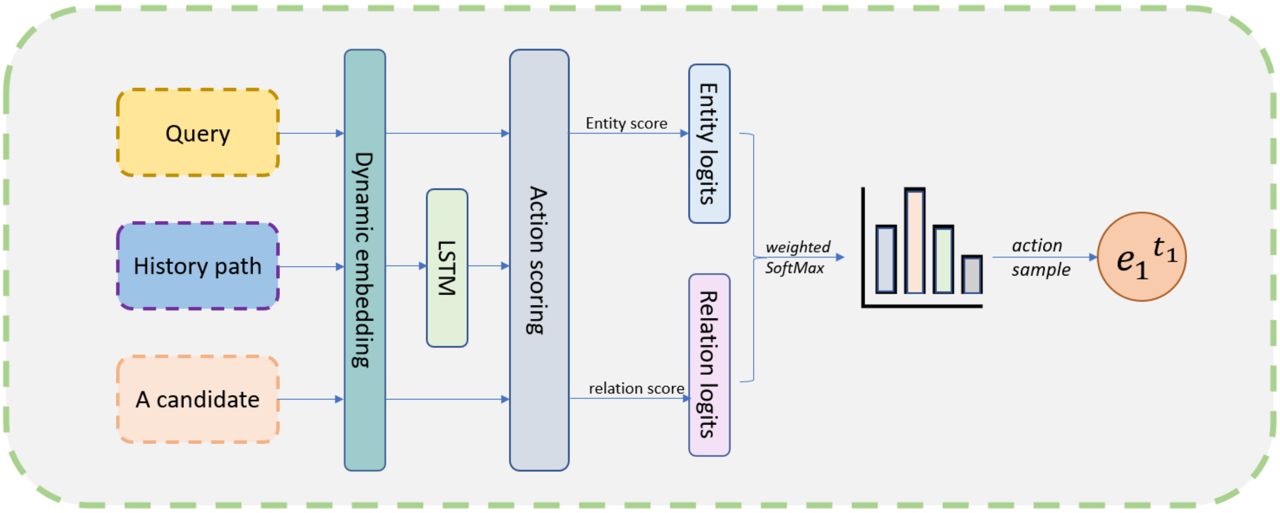

Figure 3.3: Policy Network: this diagram shows the procedure of how actions are scored and determined, output entity and relation embedding are generated by query, candidates and history encoding, then candidates relations and entities are score by calculating similarity with the output representation.Finally, actions are ranked by weighted sum of scores of entity and relation.

3.4.2 Dynamic Embeddings

We keep the embedding generating strategy of baseline. There are two types of embedding, one is for entities, another is for the relations.We take different policy for generating embeddings of entities and relations. For relation, we directly use static embeddings for each relation. But things might be different for entities, because the attributes or the cannotation of an entity may change with time passing, we define a policy for generating a dynamic(time sensitive) embedding for entity.

3.4.3 Path encoding

Since we are doing multi-hop reasoning, we want to make use of information from actions taken in previous steps (history). Same as many multi-hop model in this field, we also keep the path encoder(an LSTM cell) in baseline model to encode the information of previous action taken, also the LSTM is kept unchanged when last action is a self-loop edge.

3.4.4 Action scoring

The scoring process consists from simply 3 step:

- Using 2 MLPs separately generate representations for entity and relation.

- Among all the relation and entiry in candidates action, we assign score by calculating their similarity with above representation output.
- Rank the weighted sum of 2 scores in each action, choose the result.

3.5 Reinforcement Training Process

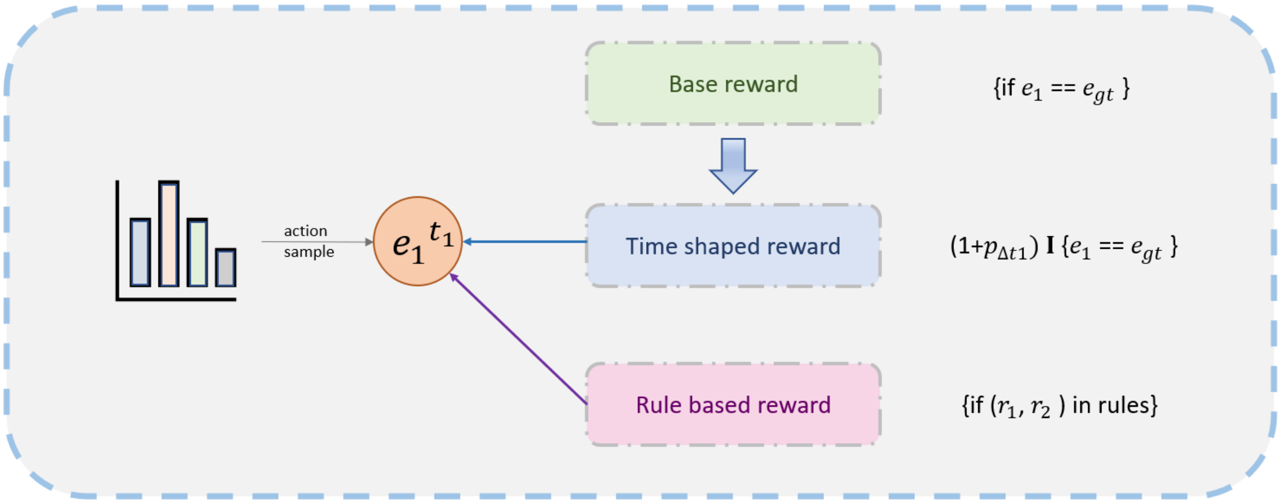

Figure 3.4: Our Reinforcement Learning Framework: giving reward to the chosen entity from 3 different types of mechanism

As we all know, the core concept of a reinforcement learning framework is the reward mechanism design, in this section an aggregated reinforcement learning process will be introduced.

3.5.1 Training Process

In following pseudo-code introduces our reinforcement learning process including how we give, expand our reward and calculate the final reinforcement learning loss.

Algorithm 1: Reinforcement Learning Process

```
Input: input parameters A, B, C
Output: output result
if do_train;
foreach epoch do
    foreach batch do
        loss, logits, relations, current_entities, current_time =
          self.model(src_batch, time_batch, rel_batch)

        base_reward = pg.get_reward(current_entities, ground_truth)
        rule_reward = pg.get_rule_reward(relations)
        reward = (1 + rule_reward)*base_reward
        shaped_reward = get_time_shaped(reward)

        Reinforcement_loss = Calc_reinforce_loss(loss,logits, shaped_reward)
        Reinforcement_loss.backward()
        total_norm = torch.nn.utils.clip_grad_norm(mode.parameters)
        optimizer.step()
    end
end
```

3.5.2 Base reward

After fixed times(path length of prediction, if the terminal gets correct, which means the agent indeed get right the destination entity the same as ground truth, it will receive a basic reward of value 1, otherwise it receives 0 reward.

3.5.3 With shaping

Also, in base paper Sun et al. (2021), some entities and relations are intensive in a specific time range, in other words, there could be a large difference among graph distributions of various timestamps and the data may look sparse across the time dimension.

So this 'reward with shaping' comes from the prior knowledge of entity distribution across the time to give a 'hint' to the agent. In an ideal way, we hope the this reward taught the agent about what range of timestamps an entity occurs in.

In base paper this part is implemented with 'dirichlet' distribution, to build up the prior distribution knowledge, firstly iterate each entity e_i, count how many times e_i appeared within recent 'K' snapshot in past, here we could regard 'K' as a hyperparameter.

3.5.4 Rule affirmed

Except the reward mechanism that we mentioned above in the baseline model , we also designed our own additional 'rule_based reward' mechanism. Having made use of a reinforcement learning framework and prior distribution guidance, we are thinking about making use of existing rules in corresponding dataset. For example we take top rules digged from tranining set, if current destination or part of path exists in our rule list, we could give extra reward to this path.

For the whole predicted path, we give reward to taking step that meets the rules, which means that , the more choices you made seen in our rules, the more reward you receive. The drawback here is that here calculating the reward could be very time-consuming, there are quantities of iterations to go.

Chapter 4

Evaluation

4.1 Experiment

4.1.1 Software environment

Table 4.1: I run experiments upon my personal PC with Windows 10 operating system, specific package name and version are stated in following table.

Number	**Package**	**version**	**Year**
1	Python	3.7	2022
2	Pytorch	1.7.0	2022
3	CUDA	10.1	
4	CUdnn	7.6.5	

4.1.2 Hardware platform

Table 4.2: The detailed configuration of main modules including CPU,GPU and RAM of two PC are given below.

CPU	**Main Frequency**	**GPU**	**Memory**
Ryzen 7 3700X	3.59GHz	RTX2070s	16GB DDR4
Core i5-9300H	2.40GHz	GTX1650	16GB DDR4

4.1.3 Experiment Design

There are many hyper-parameters in our models, for some of them we keep the optimal value used in baseline model, such like path length, learning rate, step sizes, and many kinds of weights. we run limited experiments with different rule reward weight to see how our model performs.

4.1.4 Experiment record

Table 4.3: This table records our experiment results, the best performance is achieved when batch size is 2048 and rule weight at 0.1.

num epoch	path length	batch size	rule weight	MRR	H@1	H@2	H@3
400	3	64	0.05	0.4094	0.3148	0.4607	0.5837
100	3	2048	0.1	**0.4113**	**0.319**	**0.4615**	0.5817
100	3	2048	0.3	0.405	0.3126	0.4532	0.5786
100	3	2048	0.15	0.403	0.3087	0.4522	0.5804
100	3	1024	0.1	0.4029	0.3089	0.4538	0.5789
100	3	4096	0.1	0.4033	0.3108	0.454	0.577
100	3	512	0.1	0.4059	0.3123	0.4565	0.5796

MRR: Mean reciprocal rank H@n: Hits@n.

4.2 Comparison

Table 4.4: This table shows results comparison between some outstanding model doing similar work. We could see that rule-base reward indeed take improvement to our baseline model in three metrics out of four. (models over mid-line are not sensitive to unseen timestamp)

model	MRR	H@1	H@2	H@3
TTransE Jiang et al. (2016)	13.43	3.11	17.32	34.55
TA-DistMult García-Durán et al. (2018)	26.47	17.09	30.22	45.41
DE-SimplE Goel et al. (2019)	32.67	24.43	35.69	49.11
TNTComplEx Lacroix et al. (2020)	32.12	23.35	36.03	49.13
RE-NETJin et al. (2020)	38.28	28.68	41.34	54.52
TITerSun et al. (2021)	40.99	**31.96**	45.69	57.81
TITer + rule	**41.13**	31.90	**46.15**	**58.17**

4.3 Summary(Explaination and Insight to the Result)

From the current results we get from limited experiments, we can see our modification (TITer+rule) indeed made a bit improvement compared with our baseline model, which shows that rules-based reward mechanism may indeed has potential contributing to the TKG model performance.

This also affirmed the potential of reinforcement learning framework in TKG prediction task. Also, on one hand, we need more datasets within multiple fields to construct a multiple-perspective assessment to our model. On the other hand, we just build up a simple implementation of our initial idea, searching more creativeness in reward design might be a new research orientation.

Chapter 5

Concluding Remarks

5.1 Conclusion

In this report, we went through some outstanding works in fields related to our topic. We introduced the baseline model TITer, gave our own design by making use of rules and build new reinforcement learning framework.

We also did experiment upon 'ICEWS-14' dataset and performed comparison with other models in the same field. This fact gives us a positive feedback that rule-based reinforcement learning could indeed work in this task. Though we indeed achieved little improvement, there could be much more work to explore.

5.2 Future Work

5.2.1 Different rule applied

- We need to firstly determine how to dig the rules, as we are doing prediction upon TKG, we may need to search rule including time dimension, however this may cause sparsity on the rule result that we dig. So, we consider firstly give attempt upon purely static version of corresponding dataset. In future work, we could how can we introduce temporal information into the rules we mined.

- Another view is that we could improve the method of how to give reward more precisely. For the whole predicted path, we give reward to every step that meets the rules, which means that , the more steps you took right, the more reward you receive. Besides this, we could give the reward along whole path with a precise reward policy.

5.2.2 Application Scenarios

Our temporal knowledge graph modeling method also has many fields that could be applied in.

- Both medical institutions and insurance industry could embeds this model into their business. For example we could construct each patient's health history and conditions information into a temporal knowledge graph. With adequate data from tons of patients, we could train models for predicting upcoming health problems, finding potential risk and giving timely warning.
- This model may also helps in financing institution, the stock market is changing all the time, and many transactions are happening every, if we use TKG to model all this facts with time information, we may let the model be able to predict some upcoming risk and crisis.

Their are many strong rules and laws in both of this two fields mentioned above, so we have more chances to make use of these information constructing rule-based reinforcement learning framework.

Appendix A

Appendix: Explanation on Appendices

Appendix B

Appendix: Explanation on Page Borders

Bibliography

Bordes, A.; Usunier, N.; Garcia-Durán, A.; Weston, J.; and Yakhnenko, O., 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13 (Lake Tahoe, Nevada, 2013), 2787–2795. Curran Associates Inc., Red Hook, NY, USA. [Cited on pages 5 and 9.]

García-Durán, A.; Dumancic, S.; and Niepert, M., 2018. Learning sequence encoders for temporal knowledge graph completion. *CoRR*, abs/1809.03202 (2018). http://arxiv.org/abs/1809.03202. [Cited on page 21.]

Goel, R.; Kazemi, S. M.; Brubaker, M. A.; and Poupart, P., 2019. Diachronic embedding for temporal knowledge graph completion. *CoRR*, abs/1907.03143 (2019). http://arxiv.org/abs/1907.03143. [Cited on page 21.]

Hamilton, W. L.; Ying, R.; and Leskovec, J., 2017. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216 (2017). http://arxiv.org/abs/1706.02216. [Cited on page 8.]

Ji, G.; He, S.; Xu, L.; Liu, K.; and Zhao, J., 2015a. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 687–696. Association for Computational Linguistics, Beijing, China. doi:10.3115/v1/P15-1067. https://aclanthology.org/P15-1067. [Cited on page 5.]

Ji, G.; He, S.; Xu, L.; Liu, K.; and Zhao, J., 2015b. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 687–696. Association for Computational Linguistics, Beijing, China. doi:10.3115/v1/P15-1067. https://aclanthology.org/P15-1067. [Cited on page 5.]

Jiang, T.; Liu, T.; Ge, T.; Sha, L.; Chang, B.; Li, S.; and Sui, Z., 2016. Towards time-aware knowledge graph completion. In *Proceedings of COLING 2016, the 26th*

International Conference on Computational Linguistics: Technical Papers, 1715–1724. The COLING 2016 Organizing Committee, Osaka, Japan. https://aclanthology.org/C16-1161. [Cited on page 21.]

Jin, W.; Qu, M.; Jin, X.; and Ren, X., 2020. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. [Cited on pages vii, 6, 9, and 21.]

Lacroix, T.; Obozinski, G.; and Usunier, N., 2020. Tensor decompositions for temporal knowledge base completion. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rke2P1BFwS. [Cited on page 21.]

Lei, D.; Jiang, G.; Gu, X.; Sun, K.; Mao, Y.; and Ren, X., 2020. Learning collaborative agents with rule guidance for knowledge graph reasoning. doi:10.48550/ARXIV.2005.00571. https://arxiv.org/abs/2005.00571. [Cited on page 6.]

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X., 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15 (Austin, Texas, 2015), 2181–2187. [Cited on page 5.]

Singhal, A., 2012. Introducing the knowledge graph: things, not strings. https://www.blog.google/products/search/introducing-knowledge-graph-things-not/. 2020-11-13. [Cited on page 3.]

Sun, H.; Zhong, J.; Ma, Y.; Han, Z.; and He, K., 2021. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. [Cited on pages 6, 16, and 21.]

Trivedi, R.; Dai, H.; Wang, Y.; and Song, L., 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. doi:10.48550/ARXIV.1705.05742. https://arxiv.org/abs/1705.05742. [Cited on pages 8 and 9.]

Trivedi, R.; Farajtabar, M.; Biswal, P.; and Zha, H., 2019. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HyePrhR5KX. [Cited on pages 8 and 9.]

van der Maaten, L. and Hinton, G., 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9, 86 (2008), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html. [Cited on page 8.]

Wan, G.; Pan, S.; Gong, C.; Zhou, C.; and Haffari, G., 2021. Reasoning like human: Hierarchical reinforcement learning for knowledge graph reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20 (Yokohama, Yokohama, Japan, 2021). [Cited on pages 7 and 9.]