# T2LP: mining hidden relationships between relationships

## Abstract

Recently, knowledge reasoning technology has been made towards improving with the development of knowledge map embedding. Researchers have evolved knowledge graph embedding research based on TransE. But people ignores that there are often connections between relationships in reality. In this paper, we propose to generate the time-based knowledge graph link prediction (T2LP) according to time-evolution knowledge graph embedding, which enables knowledge graph contains time information. We effectively use hidden Markov theorem to explore the possible relationships between relationships from T2LP and improve the accuracy of link predictions. Experimental results on temporal datasets extracted from read-world KGs show that our model achieves significant improvements compared to baselines.

## 1 Introduction

Recent years, With the development of machine learning, knowledge graphs (KGs) are applied to more and more fields. For example, it is applied in the dialogue system [<https://arxiv.org/abs/1906.02738> // <https://www.aclweb.org/anthology/P19-1081>]. They can help users complete specific tasks such as ordering, driving and also keep users from feeling lonely. Recently KGs are also more popular and used in NLPs [<https://drive.google.com/file/d/1-sIhvWD-kPiwabImXS-dgvDNPOsGNV-n/view> // [https://www.aclweb.org/anthology/P19-1598 //](https://www.aclweb.org/anthology/P19-1598%20//) <https://arxiv.org/abs/1904.03396>]. They express triples as natural language like pic1[<https://drive.google.com/file/d/1-sIhvWD-kPiwabImXS-dgvDNPOsGNV-n/view>]. As a type of reading comprehension task, researchers like to use a "QA system" to track the latest progress of large models such as BERT [<https://arxiv.org/abs/1810.04805>]. [A. Saha, G. Ahmed, A. Laddha][ <https://www.mitpressjournals.org/doi/pdf/10.1162/tacl_a_00262>] includes several actions (e.g., set up intersection. Knowledge graph embedded lookup, etc.) is proposed for reinforcement learning to use to derive logical programs that can answer complex questions in a conversational environment.

In fact, KGs embedding is also as a very active area of research over the last few years. TransE【】tried to dig out the relationships between triples <head, relation, tail>. TransH[] addresses TransE's inability to do one-to-many, many-to-one, many-to-many. TransR[] solves the mapping problem of triples through multi-vector space. However, we find that transactions in real life always have a certain time dimension, which requires us to take time information into account when exploring triple mapping. T-transE【】 and HyTE【】 have added time dimension to KGs embedding and indeed improved the effect, but they did not consider that the occurrence of relationship requires time limitation. For example, <Bruce\_lee was\_born\_in San\_Francisco 1940-11-27> and <Bruce\_lee died\_in Hong\_Kong 1973-07-20> that the time of death is longer than the time of birth. Therefore, we consider using time-evolution knowledge maps to explore potential relationships between relationships. To test our idea, we used the YAGO[] with time information to analyze the possible implicit relationships among the relationships. In order to verify the effect, t-TransE and HyTE were used as baseline to compare MeanRank and hit@10 and hit@1 parameters.

## 2 Background

### 2.1 Traditional KGs embedding

The concept of triples proposed by Bordes in 2013 is the most classical KG embedding method**.** It introduced the mapping method between head and tail entity vectors. Given two entity vectors , and their relation vector . It uses formula to illustrate the relationship between <head, relationship, tail>. There is the score function which is graded the triples.

|  |  |  |
| --- | --- | --- |
|  | , |  |

where, is the or -norm of the difference vector. TransH proposed a new score function to solve one-to-many, many-to-one and many-to-many problems.

|  |  |  |
| --- | --- | --- |
|  | , |  |

where

### 2.2 Temporal KGs embedding

## 3 T2LP

## 4 Experiments

## 5 Conclusion