Capturing Dynamicity of Relations in Temporal Knowledge Graphs

Submitted in partial fulfillment of the requirements for the degree

of

Master of Technology

by

Aditya Dandriyal
142202003

Under the guidance of

Dr Koninika Pal

IIT PALAKKAD

DEPARTMENT OF DATA SCIENCE

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

Table of Contents

List of Figures

Chapter 1

INTRODUCTION

Incorporating human knowledge is one of the research directions of artificial intelligence. Knowledge representation and reasoning, inspired by human problem solving, is to represent knowledge for intelligent systems to gain the ability to solve complex tasks. **Knowledge graphs(KG)** as a form of structured human knowledge representation have gained much attention recently. A KG is a multi-relational graph composed of entities (nodes) and relations (different types of edges) as shown in Fig. 1.1 . Each entity of the KG represents an abstract concept or concrete entity of the world and relationships are predicates that represent facts involving two entities.

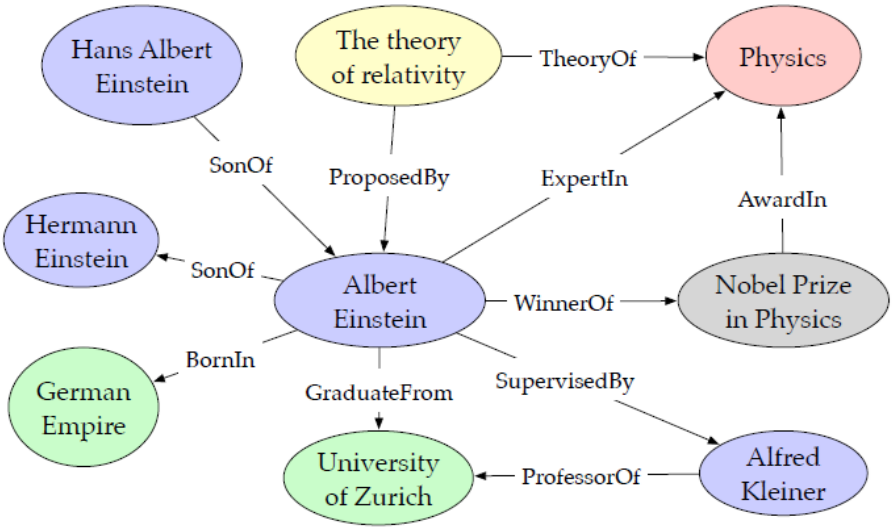

Figure 1.1: Knowledge Graph [1]

Each edge connecting two entities, is represented as a triple of the form (head entity, relation, tail entity), also called a fact, indicating that two entities are connected by a specific relation, e.g., (AlfredHitchcock, DirectorOf, Psycho). Although KGs are effective in representing structured data, the underlying symbolic nature of the way data is encoded as triples (i.e. head, tail, relation) usually makes KGs hard to manipulate.

To tackle this issue and to use a KG efficiently, the idea of **knowledge graph embeddings** has been proposed where the entities and relations of a KG are embedded as low dimensional vectors into a continuous vector space, such that similar information is closer to each other in the vector space, so as to simplify the manipulation while preserving the inherent structure of the KG. These entity and relation embeddings can further be used to benefit all kinds of tasks, such as KG completion, relation extraction, entity classification, and entity resolution.

Relational facts in KG often show temporal dynamics, e.g., the fact (Cristiano_Ronaldo, playsFor, Manchester United) is valid only from 2003 to 2009. Most of the existing KG embedding methods ignore this temporal dimension while learning embeddings of the KG elements. These methods treat the KG as a static graph with the assumption that the beliefs contained in them are universally true which is clearly inadequate and it is quite conceivable that incorporating temporal scopes during representation learning is likely to yield better KG embeddings, hence the need for **Temporal Knowledge Graphs** and temporally aware knowledge graph embeddings.

Temporal information is considered in temporal-aware embedding by extending triples into temporal quadruple as (h, r, t, τ), where τ provides additional temporal information about when the fact held.

Most of the currently available KG embedding techniques perform the embedding task solely based on observed facts i.e. while learning, the learned embeddings are only required to be compatible within each fact, and hence might not be predictive enough for downstream tasks. There is a wide variety of additional information e.g., entity types, relation paths, textual descriptions, as well as logical rules that can be incorporated to learn more predictive embeddings and further improve the downstream tasks. One such additional information is in the form of **literals**.

Relations in a KG are classified into two main categories, **object relations** and **data type relations**. An object relation links an entity to another entity (head entity, object relation, tail entity), e.g. (Albert Einstein, WinnerOf, Nobel Prize) whereas a data type relation links entities to data values also known as literals(head entity, data type relation, literal value), e.g. (John, BirthYear, 2001) where 2001 is a numerical literal (data value). Literals can include various types of information such as text (e.g., names, labels, descriptions), numeric values (e.g., height, population), and units of measurement and they are stored as strings in a KG[2].

1.1 Motivation

All the knowledge graph embedding approaches focus on learning embeddings for facts having object relations only. Certain studies do focus on approaches that deal with facts having data type relations but they do so by incorporating the literals in such facts as auxiliary information for learning embeddings of object relation facts only, i.e. embedding approaches do not learn embeddings for literals, and thus cannot perform prediction tasks for literals.

This motivated us to come up with an approach that can learn embeddings for literals, specifically numerical literals. We are so focused on numerical literals as it has their own semantics to cover which cannot be covered by string distance metrics, e.g. 777 is more similar to 788 than 77. Learning embeddings for numerical literals will enable us to perform link prediction tasks even for data type relations.

1.2 Objectives

Given a temporal knowledge graph that includes facts having object relations and data type relations(numeric literals), both enriched with temporal information, the aim is to be able to learn embeddings for entities, relations and even literals such that we are able to capture the hidden semantics(hierarchy) among the values of the numeric literals in data type relations and further be able to analyze the dynamicity (frequency, granularity) of such relations.

Chapter 2

LITERATURE REVIEW

2.1 Knowledge Graph Embeddings [Wang et al.,2017]

This literature gives an overview of a knowledge graph, need for an embedding approach and further the downstream tasks that can be implemented using the knowledge graph embeddings. A typical KG embedding technique generally consists of three steps namely, entity and relation representation, defining a scoring function and learning the entity and relation representation. Since this is a survey paper so I had to limit my study to translational distance models which make use of distance based scoring functions, such as TransE, TransH etc.

This paper also explains in detail the training process involved in a KG embedding technique such as the loss function used, initialization of entity and relation embeddings, generation of negative samples etc. Further it also explains the various downstream tasks that can be implemented using the learned embeddings such as link prediction, relation prediction and triple classification to name a few. [3]

2.2 TransE [Bordes et al.,2013]

TransE (Translational Embeddings) is an energy-based model for learning low-dimensional embeddings of entities, focusing on minimal parametrization of the model to primarily represent hierarchical relationships. In TransE, relationships are represented as translations in the embedding space: if (h, r, t) holds, then the embedding of the tail entity t should be close to the embedding of the head entity h plus some vector that depends on the relationship r. This approach relies on a re-

duced set of parameters as it learns only one low-dimensional vector for each entity and each relationship.

The main motivation behind this translation-based parameterization is that hierarchical relationships are extremely common in KBs and translations are the natural transformations for representing them.An interesting observation of this paper is that other models which are seemingly more expressive do not necessarily have better performances. One of the reason for this could be that these models require a large number of parameters and tend to overfit on small and medium sized datasets.

TransE is simple and efficient while preserving state-of-the-art predictive performance. Despite its simplicity and efficiency, TransE has flaws in dealing with 1-to-N, N-to-1, and N-to-N relations as it tends to learn very similar vector representations for an entity for every relation it is involved with, ignoring distributed representations of entities when involved in different relations. [4]

2.3 TransH [Wang et al.,2014]

TransE has flaws when dealing with 1-to-N, N-to-1, and N-to-N relations and some advanced models are capable of preserving these mapping properties but they do not match the TransE approach when it comes to predictive performances(efficiency) and also have greater model complexity(more parameters).

TransH (Translation on Hyperplanes) is a approach which makes a good trade-off between model complexity and efficiency such that it is able to overcome the flaws of TransE while inheriting its efficiency. It does so by enabling an entity to have distributed representations when involved in different relations. To achieve this, for a relation r, a relation specific translational vector is positioned in a relation specific hyperplane rather than in the same space of entity embeddings. Then for a triplet (h,r,t) the h and t embeddings are projected onto the hyperplane to get the relation specific representations of the entities.

This research also focuses on the impact of reducing false negative samples during training process by assigning different probabilities for replacing the head or tail

entity which depends on the mapping property of the relation. Extensive experimentation proved that in downstream tasks such as link prediction TransH shows promising improvements to TransE. [5]

2.4 HyTE [Dasgupta et al.,2018]

Relational facts in knowledge graphs often show temporal dynamics but most of the existing KG embedding methods ignore this temporal dimension while learning embeddings. These methods treat the KG as a static graph with the assumption that the beliefs contained in them are universally true. This is clearly inadequate and it is quite conceivable that incorporating temporal scopes during representation learning is likely to yield better KG embeddings.

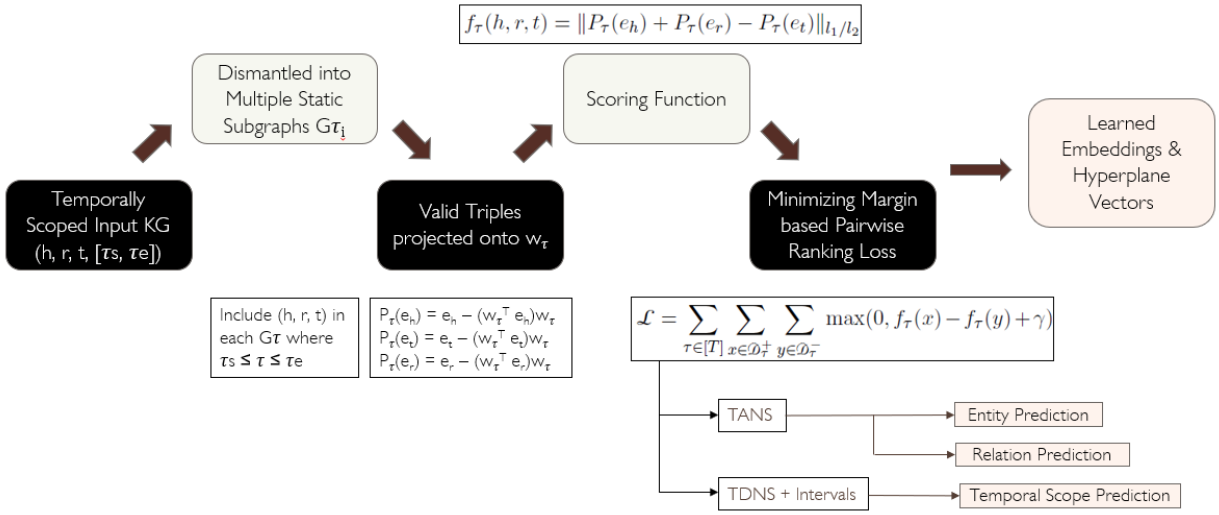

Figure 2.1: HyTE Framework

Previous studies have also utilized temporal scopes but they did so indirectly such as t-TransE which learns time aware embedding by learning relation ordering jointly with TransE by inflicting temporal order on time-sensitive relations. However, HyTE directly incorporates temporal information in the learned embeddings by associating each timestamp with a corresponding hyperplane and then projecting the entities and relations in a valid fact into the associated timestamp hyperplane and learning KG embeddings distributed in time. This further makes it possible to predict the temporal scopes which the previous approaches failed to achieve.

HyTE outperformed both the traditional and time aware embedding methods on tasks such as entity and relation prediction and further even made temporal scope prediction possible.HyTE frame work is shown in Fig. 2.1 [6]

2.5 LiteralE [Kristiadi et al.,2019]

Knowledge graphs are composed of different elements: entity nodes, relation edges, and literal nodes. Each literal node contains an entity's attribute value (e.g. the height of an entity of type person) and thereby encodes information which in general cannot be represented by relations between entities alone. However, most of the existing embedding- or latent-feature-based methods which learn low dimensional vector representations of entities and relations for knowledge graph analysis, only consider entity nodes and relation edges, and thus do not take the information provided by literals into account.

LiteralE is an approach which lets us use the additional information provided by the literals by enriching an entity embedding with its corresponding literals using a learnable parametric function which gets the vanilla embedding and the entity's literals as input, and outputs a literal enriched embedding for the entity. This embedding can then replace the vanilla embedding in any latent feature model, without changing its original scoring function and the resulting system can be jointly trained with stochastic gradient descent, or any other gradient based algorithm of choice, in an end-to-end manner.

LiteralE can be seen as an extension module that can be universally combined with any existing latent feature method and furthermore, through the findings of this research has proven to improve the quality of the entity embeddings as reflected by the improvement in the results of downstream tasks like link prediction.[7]

Chapter 3

METHODOLOGY

As seen from the literature review the approaches available are capable of incorporating literal information into KG embedding learning process but are incapable of learning embeddings for literals and thus literal value prediction. The numerical literal data considered for our approaches has data type relation having "year" literals only. So across all the approaches, we have only "year" literals. For all of the approaches, we used the HyTE framework for learning embeddings. In the HyTE paper and for our approaches we have considered only year-level granularity for temporal scope.

Following are some of the methodologies with their own set of limitations which make learning embeddings for numerical literal data a possibility.

3.1 HyTE_Lit_Entity

In this approach we utilised only the filtered numerical literal data(YAG03-10 plus) and considered each unique numerical literal as a distinct entity while learning the embeddings using the HyTE framework.

Drawback: In real world scenarios data type relations such as profit/loss coupled with a finer level of temporal granularity, e.g. (CompanyA, ProfitOf, 40000, 08-2020) has a month level granularity, can have infinite set of values for numerical literals which would make the number of unique entities in the tail to be very high as compared to the head. Thus it is not a very feasible/effective way to go about handling literals.

3.2 HyTE_Comb_Entity

In this approach we utilized the combination of a filtered YAGO3 data and the numerical literal data and handled literals in the same manner as in approach 1 i.e. considered each unique numerical literal as a distinct entity while learning the embeddings using the HyTE framework.

This approach differs from the first one in the manner that it is using a dataset that includes facts having both object relations and data type relations while the first approach used only data type relation facts.
Drawback: same as the first one.

3.3 HyTE_Comb_Classes

In this approach we utilised the combined dataset, the same as in approach 2 and considered class intervals for the numerical literal values. The literal values within the same class are considered to be a single entity. We considered a class interval of 100 years for the literals. For e.g. If we consider the literal value from 1900 to 2000 ("year" literal) to be within the same class interval, all such literal values within the interval will be considered as the same entity.

This approach is advantageous over the previous one in the manner that if we consider class intervals for the literal values, then by choosing an appropriate interval we could easily accommodate for a very high range of the literal values and thus there would be a balance between the number of unique entities in the head and in the tail.

Drawback: Considering class intervals is a good enough approach but it would lead to low exactness in the prediction as the output would be a range rather than a single value. So to get better accuracy, we are compromising on precision. But in a way we still are learning literal values in the task of temporal scoping. Also not to mention the level of complexity that would come when dealing with multiple different type of numerical data type relations (e.g. year, profit/loss, height, temperature) at the same time, as each relation would require a different class interval to be set up.

3.4 HyTE_Comb_SingleEntity

In this approach also we utilised the combined dataset. Since we only have "year" literals, we considered all the literal values as a single entity.

The HyTE framework for learning embeddings projects the entities and the relations of the quadruple (h, r, t, τ) in the hyperplane of the timestamp τ that they are associated with thereby making the embeddings temporally aware of the timestamp τ. Since we have time (year) literals whose values itself is being used to extend the literal triplets into temporally scoped quadruplets (e.g. (John, BornOn, 1922) to (John, BornOn, 1922, 1922)), so while considering the literals to be the same entity, the literal value is still present in the temporal scope, hence along with learning the temporally aware embeddings we are also learning the projection of the embeddings on to the literal value.

Additionally we can use this approach even for cases where we have multiple numerical data type relations by assigning each literal type as a unique entity (e.g. all "year" literals will be considerd as a single entity and all "monetary" literals will be considered as another single entity), while projecting the embeddings onto the literal values.

The only difficulty that will be faced is in procuring temporally scoped data for literal values other than that for "year".

Chapter 4

EXPERIMENTAL EVALUATION

4.1 Datasets

The HyTE paper uses a filtered version of YAGO3 dataset (YAGO11k) and for our experiments we used a filtered version of YAGO3-10 plus dataset which consists of numerical "year" literals and along with this we also used a combined version of these datasets.

- **HyTE paper data (YAGO11k)**: In the YAGO3 knowledge graph, some temporally associated facts have meta-facts as (factID, occurSince, ts), (factID, occurUntil, te). The total number of time annotated facts containing both occursSince and occursUntil are 722,494. Out of them, we selected top 10 most frequent temporally rich relations. In order to handle sparsity, we recursively remove edges containing entity with only a single mention in the subgraph. This ensures a healthy connectivity within the graph.

  Finally, we obtain a purely temporal graph of 20.5k quadruples and 10,623 entities by following this procedure.

- **Numerical Literal data (YAGO3-10 plus)**: This data has only data type relations having only "year" literals. It had a total of 111,406 triplets and 5 relations. We handled sparsity, by including only those triples whose head entity had atleast two mentions in the knowledge graph.

  Finally, we obtained a non-temporal graph of 37.2k triples and 17,948 unique head entities, and 1,270 unique literals. In order to make these triples temporally aware, since the literal value is associated with time (year), we consid-

ered the literal value itself to be the temporal scope. For e.g. (Sam_Thomson, diedOnDate, 1943) triple is converted to temporally aware quadruple, (Sam_Thomson, diedOnDate, 1943, [1943,1943]).

A Table showing the details of the dataset used for different methodologies:

DATASETS	#ENTITY	#Entities in head	#Entities in tail	#RELATIONS	Train/Valid/Test	
YAGO11K (HyTE)	10,623	6498	6351	10	16.4k/2k/2k	(20.5k)
HyTE_Lit_Entity	19,218	17948	1270	5	30.2k/3.5k/3.4k	(37.2k)
HyTE_Comb_Entity	28,800	23488	7621	15	47.7k/5k/5k	(57.7k)
HyTE_Comb_Classes	27,550	23488	6372	15	47.7k/5k/5k	(57.7k)
HyTE_Comb_SingleEntity	27531	23488	6352	15	47.7k/5k/5k	(57.7k)

Figure 4.1: Dataset Distribution

4.2 Experiments

HyTE paper approach

The data before being given as input to HyTE has to be pre-processed in a very unique way. Originally our data would be in the form of quadruple (head entity, relation, tail entity, [start date, end date]). We assign each unique entity and each unique relation a numerical ID and map the entities and relation in the quadruple to these ID's, thereby modifying the quadruple to contain ID's corresponding to the entities and relations instead of the entities and the relations themselves.
This approach provides us a great leverage for implementing our further methodology.
The following figures showcase the pre-processing done on the data to implement the methodologies stated in Chapter 3.

Figure 4.2: HyTE Data Preprocessing

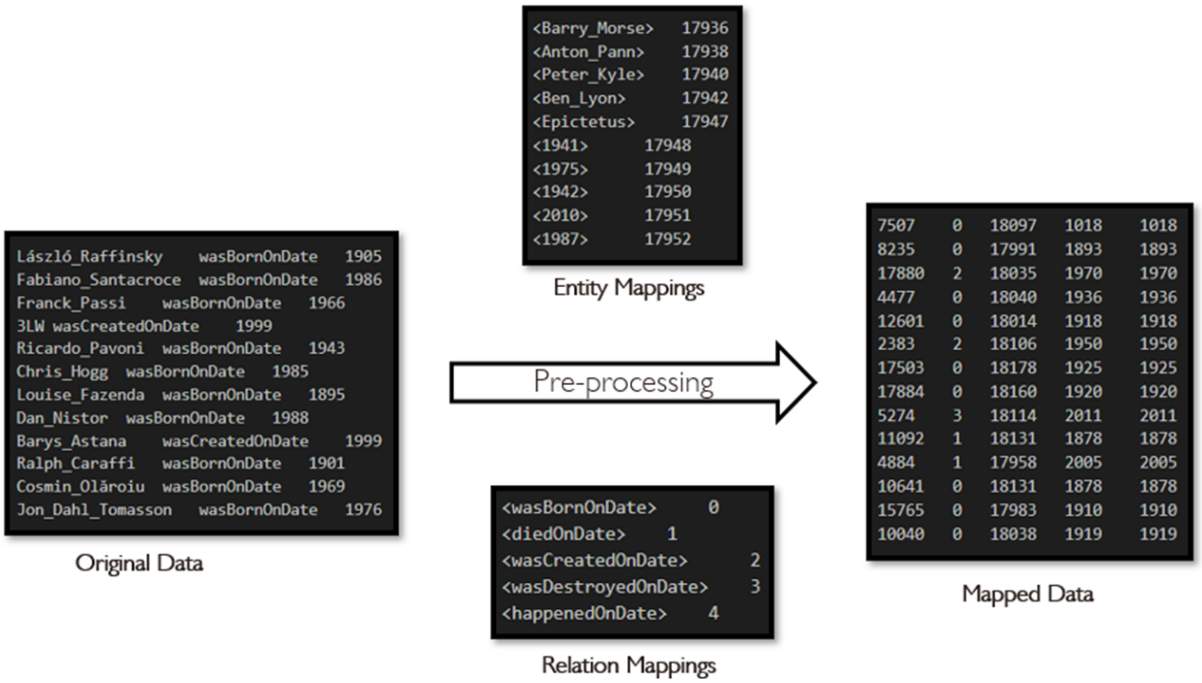

Figure 4.3: HyTE_Lit_Entity Data Preprocessing

Figure 4.4: HyTE_Comb_Entity Data Preprocessing

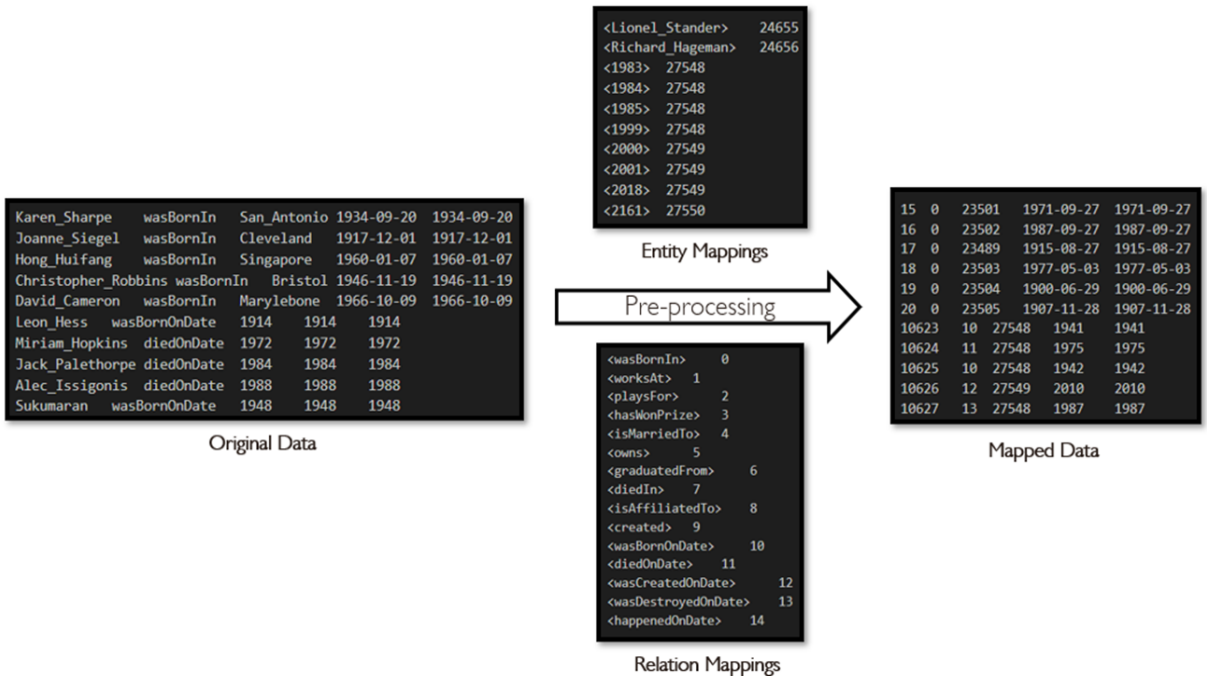

Figure 4.5: HyTE_Comb_Classes Data Preprocessing

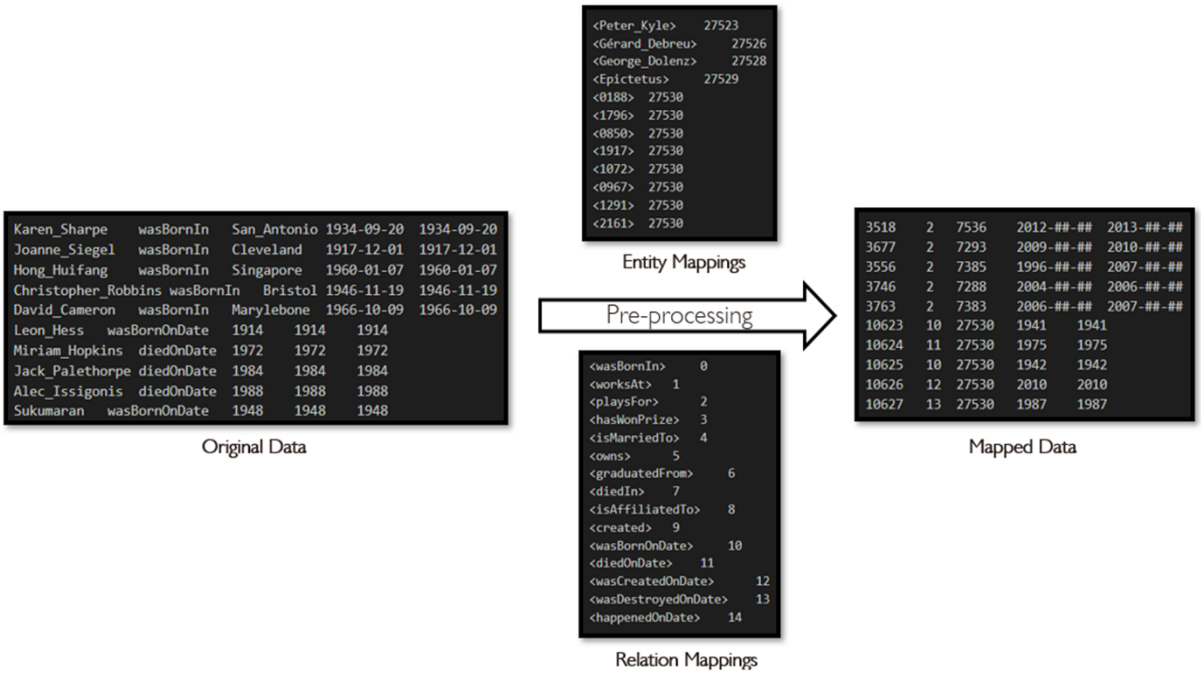

Figure 4.6: HyTE_Comb_SingleEntity Data Preprocessing

4.3 Evaluation on Downstream Tasks

After learning the embeddings, the following downstream tasks were considered for evaluation:

- **Entity Prediction**: The task here is to predict the missing entity ((?, r, t, τ) or (h, r, ?, τ)), given an incomplete relational fact with its time. For a test triple (h, r, t, τ), we generate corrupted triples by replacing tail entity (for tail prediction) or head entity (for head prediction) with all possible entities. Filtered protocol, says that the corrupted triples must not be a part of the graph itself. After corrupting we rank all the triples in increasing order of their score and find the rank of the actual triple. We report the mean rank over all the test queries (MR) and proportion of correct entities in top 10 rank (Hits@10).

- **Relation Prediction**: The aim of this task is to predict the relation between two entities, i.e., for a given time-stamped triple with missing relation (h, ?, t, τ), we predict the relation r. For evaluation, we corrupt the triples with all possible relations and report the rank of the actual relation.

- **Temporal Scope Prediction**: In this task, we predict the time interval or the time instance τ for a given test triple (h, r, t, ?). We, project the relation and the entities of the triple on all the time hyperplanes and check the plausibility of that test triple on each of them. For evaluation, we order the time frames in increasing order of their plausibility score for that particular triple. Now, we select the rank of the time (τ) associated with the test triple. If the associated time is an interval, we consider the lowest rank among the times in between the interval.

Following are the evaluation results:

Entity Prediction	Mean Rank		HITS@10(%)		HITS@20(%)	
	HEAD	TAIL	HEAD	TAIL	HEAD	TAIL
HyTE	1069	107	16.0	38.4	---	---
HyTE Re-implementation	1069	111	13.1	35.1	17.99	46.24
HyTE_Lit_Entity	11361	302	0.0	30.38	0.0	43.72
HyTE_Comb_Entity	6900	277	1.26	18.68	2.24	30.8
HyTE_Comb_Classes	4955	218	1.68	58.38	3.0	63.68
HyTE_Comb_SingleEntity	6999	177	2.10	62.95	3.26	66.05

Figure 4.7: Entity Prediction Results

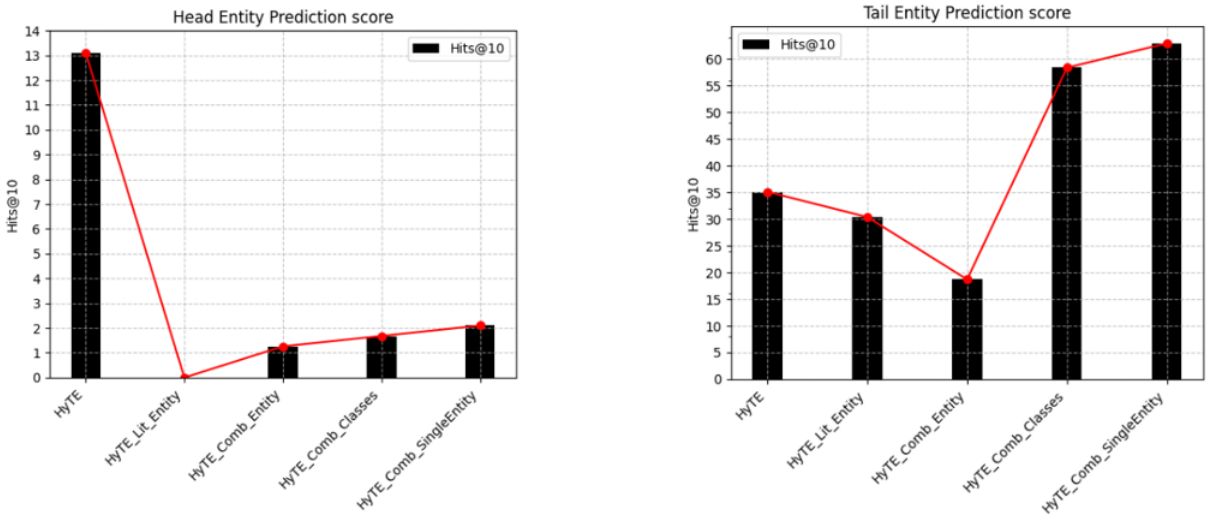

Figure 4.8: Bar Plots for Hits@10 metric for Entity Prediction

Relation Prediction	Mean Rank
HyTE	1.23
HyTE Re-implementation	1.41
HyTE_Lit_Entity	2.97
HyTE_Comb_Entity	2.22
HyTE_Comb_Classes	1.88
HyTE_Comb_SingleEntity	2.05

Figure 4.9: Relation Prediction Results

Temporal Scoping	Mean Rank (TDNS)
HyTE	9.88
HyTE Re-implementation	8.47
HyTE_Lit_Entity	44.24
HyTE_Comb_Entity	53.63
HyTE_Comb_Classes	39.8
HyTE_Comb_SingleEntity	41.23

Figure 4.10: Temporal Scope Prediction Results

4.4 Insights gained from evaluation results

- From the HyTE paper and its re-implementation results for entity prediction we can see that even though the number of unique entities in the head and the tail are almost same there is a significant difference between the evaluation score of head and tail. Hence, we can infer that the HyTE framework is learning the tail entity embeddings better as compared to head entity embeddings.

- Insights gained from Tail Entity Prediction:

    1. From the dataset distribution we can see that the number of unique entities in the head far outweighs those in the tail. Thus we can say that

a lot of N-to-1 relations exist in the knowledge graphs. This implies that since tail entities appear in a lot more facts than head entities, our framework learns better embeddings of tail entities than that of head. This is quite clearly reflected upon the score shown by the evaluation metric, as tail predictions score much better than head predictions.

2. In HyTE_Lit_Entity has 30k training data points and only 1270 tail entities(literals), thus it learns good embeddings for tail entities and shows a comparable score to HyTE tail entity prediction.
3. In HyTE_Comb_Entity, since now we have more number of unique entities in the tail, we get a lower score for tail prediction as compared to the HyTE_Lit_Entity method.
4. In HyTE_Comb_Classes, we now have 21 entities corresponding to 1270 literals(since we considered classes), that too in 30k training data hence the embeddings learned for tail are good and it is shown by a great improvement in the tail prediction score.
5. In HyTE_Comb_SingleEntity, we now have 1 entity corresponding to 1270 literals in 30k data points thus this approach shows the best tail prediction results.

- Insights gained from Head Entity Prediction:

1. As for head entity prediction, we earlier established that the HyTE framework is intrinsically learning better embeddings for tail entities, since in HyTE_Lit_Entity no tail entity occurs in the head, thus the embeddings learned for head entities is quite poor as is depicted by the evaluation metric.
2. In HyTE_Comb_Entity, we have a combined dataset so now tail entities do occur in the head and thus there is some minor improvement in the head prediction.
3. In HyTE_Comb_Classes, we have the same number of tail entities that also occur as head entities, but still it shows some minor improvement over the previous approach since the number of tail entities that don't occur in the head has reduced (1270 to 21), thus the negative samples

that will be generated will majorly have head entities only and since negative samples are also important for providing context, hence the improvement.

4. In HyTE_Comb_SingleEntity, still some minor improvements for the same reason as above.

- Insights gained from Relation Prediction:

  1. The results for the relation prediction for our methodologies is more or less the same. It is a bit worse than that in HyTE paper approach as it dealt with only 5 relations, but we in our combined dataset are dealing with 15 relations.

- Insights gained from Temporal Scope Prediction:

  1. Our approaches show very poor results when it comes to temporal scoping.
  2. The HyTE framework for temporal scoping tasks works by considering intervals instead of timestamps as hyperplanes by applying the constraint that a range is considered an interval if it has a minimum threshold of 300 points. The motive is to distribute the time annotations in the KG uniformly.
  3. Since taking the literal data into consideration, which has only 1270 unique literal values in a dataset of over 30k points, we can safely say that now most of the hyperplanes won't be an interval, they would be specific timestamps as they would individually achieve the threshold. Thus the results for temporal scope prediction deteriorate.

NOTE: In all of our approaches we are consistently getting very poor results for head entity prediction, no matter the improvement. This is mainly due to the fact that the number of unique entities in the head far outweighs that in the tail. In a real-world dataset in which literals are present, we'll have an opposite scenario, the number of entities in the tail (literals considered as entities) will far outweigh those in the head and thus we'll get poor prediction for the tail. Here our approaches will be handy to manage this imbalance and improve the prediction results.

Chapter 5

FUTURE SCOPE

- As mentioned before, the dataset that we are using doesn't strongly represent a real-world scenario. A dataset having literals should have a lot more unique literal values as compared to the unique head entities. The challenge we faced with curating such a dataset was the unavailability of temporal scopes for literal data. Hence next step would be to curate a suitable dataset that would satisfy the above mentioned criteria.
- In all of our approaches we considered only year-level temporal granularity. We could even consider finer temporal granularity.
- Coming up with a methodology that learns different flavors of embeddings for literals and entities. This is because we have numerical literal data which has its own hidden semantics, thus its only natural to learn its embeddings independently.
- To improve the temporal scoping when dealing with "year" literals, when considering the classes approach as seen in HyTE_comb_classes, we could change the temporal scope of the data type relation which has the same start and end value, by randomly assigning it a temporal scope within the class range. This will again make intervals as hyperplane instead of just a single timestamp and will improve the results.

REFERENCES

[1] Erik Cambria Pekka Marttinen Philip S. Yu Shaoxiong Ji, Shirui Pan. A survey on knowledge graphs: Representation, acquisition and applications. *IEEE Transactions On Neural Networks And Learning Systems*, 2021.

[2] Russa Biswas1 Genet Asefa Gesese1 and Harald Sack. A comprehensive survey of knowledge graph embeddings with literals: Techniques and applications. 2019.

[3] B. Wang Q. Wang, Z. Mao and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29:2724–2743, 1 Dec 2017.

[4] A. Garc_ıa-Dur_an J. Weston Bordes, N. Usunier and O. Yakhnenko. Translating embeddings for modeling multi-relational data. *Proc. Adv. Neural Inf. Process. Syst*, 138:2787–2795, 2013.

[5] J. Feng Z. Wang, J. Zhang and Z. Chen. Knowledge graph embedding by translating on hyperplanes. *Proc. 28th AAAI Conf. Artif. Intell.*, page 1112–1119, 2014.

[6] S. N. Ray S. S. Dasgupta and P. Talukdar. Hyte: Hyperplane-based temporally aware knowledge graph embedding. *EMNLP*, pages 2001–2011, 2018.

[7] D. Lukovnikov J. Lehmann A. Kristiadi, M.A. Khan and A. Fischer. Incorporating literals into knowledge graph embeddings. *ISWC2019*, 2019.