EduEmbedd – A Knowledge Graph Embedding for Education

Anurag Mohanty[1]

[1]IIIT Bangalore, 26/C, Opposite of Infosys gate 1, Electronics City Phase 1, Hosur Road, Bengaluru - 560100, Karnataka, India

Abstract

Motivated by emerging strength of Knowledge graphs as an integrated information representation and repository that interlinks heterogeneous data from different domains and the growing adoption and application of artificial intelligence for various use-cases on education domain. We propose EduEmbedd, a framework to develop an Embedding (Knowledge Graph Embedding) for the education domain and demonstrate the usefulness of such embedding. We understand that in the emerging era of Large Language Model (LLMs), domain specific embeddings based on Knowledge Graph has the potential to aid the LLMs to overcome some of the most pressing challenges like hallucinations along with improving its interpretability. The knowledge held up in the education domain is an assimilation of information from multiple contexts. EduEmbedd leverages all these different contexts into one to learn an effective embedding which can be used for various upstream machine learning tasks. The heterogeneous data from different contexts is often related to each other. In order to derive value, the data should be integrated, structured and the relationships should be made explicit. Knowledge Graphs (KG) can play a key role in achieving these goals and gives us an opportunity to assimilate information from these multiple contexts into a single unified structure and semantic form. We also understand that several novel enhancements would be required on top of this base idea to ensure that we are able to deal with the nuances of the domain for which we are creating the embedding. EduEmbedd is a step towards this direction where we introduce a systematic framework to create an Embedding for the education domain by leveraging Knowledge Graph Embedding (KGE) approaches. We also demonstrate how these embeddings are useful in terms of its ability in representing the composite knowledge being held up in them along with the efficacy it brings to machine learning using this approach.

Keywords

Knowledge Graph, Knowledge Graph Embedding, Large language Model,

1. Introduction

The information held up in education learning data(courses, concepts etc.) inherently possesses multiple contexts. There are two major contexts which we will factor currently in EduEmbedd. These are the information held up in the **Pedagogical context** and information from the actual content of learning data or **Content context**. So the inherent knowledge that the entities possess is made up by the collective knowledge from these contexts.

Pedagogical context - The pedagogical contexts denotes the various pedagogical information for the learning course. It is a form of meta information for the learning data which captures the learning entities interactions with other leaning entities. To understand this, we can visualize the entire information in an education domain as comprising of entities that share one or more relationship between each other. For example, a educational learning course (an example of entity) could have a particular learning complexity level (beginner, expert etc.) and this could be visualized as a form of relationship between them, Similarly, educational learning courses can possess inter-dependencies contexts (supplementary, complementary, composite, contradicting) between each other and these inter-dependencies could be also visualized as a form of relationship between them. Similarly various learning entities like courses, subjects, concepts, topics etc. could possess an inheritance (IS-A) or composition (Has-A) relationship between each other. These are some of the examples of pedagogical relationships seen in the learning universe.

Content context – Education Learning data also possesses the knowledge from the actual learning content or knowledge coming from the learning text and the concepts present in them. For example, the instruction concepts that are being described in a course, the topics present in the course etc. are some examples of information representing the content context. These information from the content context can be efficiently linked to the entity(example; a course) through appropriate content type relationships.

With respect to the above understanding, we could say that an entity (course, topic etc.) in a learning universe has two broad context and these contexts are translating to corresponding type of relationships which defines an entity; pedagogical relationships and content type relationships with other entities. For example, if we take any education course or any chapter of the course as an entity, then this entity(chapter) is defined both in terms of

Workshop on Enterprise Knowledge Graphs using Large Language Models, Oct 22, 2023, Birmingham, UK

*Corresponding author.

anurag.mohanty@iiitb.ac.in (A. Mohanty)

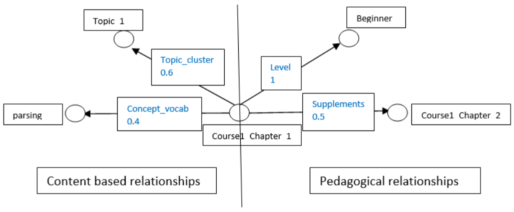

Figure 1: Knowledge Graph with Content based and Pedagogical type relationship with weights associated to triples.

the pedagogical relationship of this chapter/course with other chapters/course and also the actual content being held in the chapter. As our intention is to develop a KG, it makes sense to see these context as a form a relationship.

Most of the prominent past related work have considered either one of these contexts to arrive at an embedding for an educational learning entity. However, the advent of KG provides us an opportunity to assimilate these heterogeneous type of data in order to collectively represent the entity. This now opens up to the idea of generating an embedding (KGE) of an entity in the education domain by assimilating the information and knowledge held in multiple(both) contexts; thus potentially giving the embedding representation a much broader perspective and semantic coverage. KGE is a machine learning task of learning a low-dimensional representation of a knowledge graph's entities and relations while preserving their semantic meaning.

An entity in an education universe could be visualized as a function of its existence as per all the relationship it shares with other entities. This concept fundamentally gives more breadth to the representation of these entities and a KGE fulfils this requirement to a large extent. These generated embedding vectors aim to capture the latent properties of the semantics in the KG and so similar entities and similar relationships will be represented with similar vectors.

However, Education KG could not be perceived as just having the relationship between entities. In many scenarios we find that the relationship must carry certain weights. The weights would signify the degree of association (of the relationship) between the entities. This degree of association could semantically mean confidence, strength, probability etc. We explain this with few examples for both pedagogical and content based relationships.

For pedagogical relationship, we take the example of two different course chapters (entities) which are related with the 'supplements' relationship. From an education learning perspective it is a common understanding that for a given learning course chapter, multiple other course chapter could supplement differently or in different degrees i.e. chapterA 'supplements' chapterC more than chapterB 'supplements' chapterC. Similarly other pedagogical relationship could be efficiently weighted so that we can capture the degree of association, confidence, probability etc of the relationship into the embeddings.

For content relationship, a similar example is of a certain topic which could be comparatively covered / described more exhaustively in a particular course chapter than other.

So, EduEmbedd derives the embeddings by factoring these two most important aspects seen prevalently in education domain:

- Amalgamating the knowledge from multiple context (Pedagogical context and Content context);
- Weighted relationship between entities.

2. Related Work

In this section, we explore and report some of the pioneering works around the area of creating a KG and KGE for academia or in an educational realm. The novelties introduced in this paper are quite salient where we use both Pedagogical as well as content based information. On top of this we also use weights to arrive at the final KGE. However we have studied few pioneering work done on this domain which we have listed here.

Few studies focus on systematic construction of domain specific KG. We are yet to find any prominent work of construction of KGE for such domain specific KG or for a KG on education domain. However, there are some recent works investigating different relation extractions between certain known educational entities [1] extract concepts hierarchies from the textbooks; [2] induce structures of multiple units in a course; and [3] recover prerequisite relations from university course dependencies. One of relevant work to our research is carried out by Carnegie Mellon University: the researchers utilize observed relations among courses to create a directed concept graph [4], and the relations are assumed to be known in advance. In educational industry, MOOC providers, like Khan Academy [5][8], have built some dedicated knowledge graphs for their online courses, but most are undirected graphs built by domain experts. Yu Lu et al. proposed a system , called KnowEdu [6] to automatically construct KG for education, however the KG focusses more on only one type of automatically extraction of a concept relation to build the KG. All these pioneer studies and efforts demonstrate the increasing interests and pressing needs of knowledge graph construction in education domain. Majority of the known published work in this area has either utilised the concepts/text from the course content to derive the embeddings or have utilised course hierarchy structure to derive the embeddings. The KG behind EduEmbedd is designed to amalgamate multiple heterogeneous context into one view and incorporate the concept of weighted relationship between the nodes of the KG.

Although a comprehensive survey of KGE is out of the scope of this work (recent surveys provide a good coverage of the landscape [7]), it is worth listing the most popular KGE models proposed to date. TransE [8] is the forerunner of distance-based models, and inspired a number of models commonly referred to as TransX. TransH [9] projects entities and relations into a hyperplane, TransR [10] introduces separate projection spaces for entities and relations. The symmetric bilinear-diagonal model DistMult [11] paved the way for its asymmetric evolutions in the complex space, ComplEx [12] and RotatE [13]. Holographic Embeddings of Knowledge Graphs (HOLE) [14] is related to holographic models of associative memory in that it employs circular correlation to create compositional representations. Some models such as RESCAL [15], TuckER [16], and implE [17] rely on different tensor decomposition techniques. Models such as ConvE [18] or ConvKB [19] leverage convolutional layers. Attention is used by [20].

None of the models listed above leverage numeric attributes of any kind. A number of recent works does support multimodal knowledge graphs and learn from numeric values associated to node entities. LiteralE enriches node embeddings with numeric information before scoring the triples [21]. KBLRN combines latent, relational and numeric features using product of experts model [22]. TransEA learns a vanilla structural model using TransE scoring, and an attribute model for attributed triples, using regression over the attribute values, which is jointly trained [23]. Nevertheless, such models are not designed to learn from numeric values associated to edges. On KGE to the best of our knowledge, the only work designed to work with numeric-aware edges is UKGE [24]. UKGE generates confidence scores for known triples by squashing numeric values in the [0 - 1] interval. It then uses probabilistic soft logic [25] to predict probability estimates for unseen triples, by jointly training a model to regress over the confidence values. A limitation of this approach is that out-of-band logical rules are required as additional input. It is also worth noting that UKGE design rationale aims at supporting uncertain knowledge graphs, i.e. graphs whose edge numeric values represent uncertainty. Our concept of weights is to ensure that triples with high numeric edge values have high contribution on the overall learning of the embeddings. [26] and Ampligraph [27] does supports numeric values for Representation learning of Knowledge Graphs and comes close to our work but it does not have support for TransH which is a base model for us and also our work has various improvisations required at different components of the KGE for TransH, including a different evaluation mechanism.

3. EduEmbedd Framework

3.1. Preliminaries

We first introduce the notations used in this paper. Lowercase letters in italics denote entities, relations, or types, whose bold forms denote embedding vectors. A knowledge graph G = (s, p, o) $\subseteq$ E × R × E is a set of triples t = (s, p, o) each including a subject s $\in$ E, a predicate p $\in$ R, and an object o $\in$ E. E and R are the sets of all entities and relation types of G.

Knowledge Graph with weights to the triples. In a knowledge graph G, each triple is assigned a weight attribute w $\in$ R, leading to G = t = (s, p, o, w). The weights are assigned to triples.

3.2. EduEmbedd Framework Details

Compared to a generic KGE, construction of educational KGE takes the following consideration into account:

3.2.1. Identification of Entities

Most often it is seen in the case of a generic KG that the nodes/entities are real world entities like people, place, things etc. Education domain may not have such real world entities. We have to identify the entities that we are interested with and this identification of entities is often dictated by the use case/s we are interested in solving/working using the KGE. For EduEmbedd, the entities are:

- The Course Learning chapters.
- The topics generated by a topic modelling algorithm Latent Dirichlet Allocation (LDA) – We apply LDA to the learning content text data and derive the available topics in the course chapters. Any other topic modelling method like BERTopic etc, which potentially has the capability to also determine the probability of the topic could also be used here. The topic here is a cluster of keywords that is identified by a topic modelling algorithm and is often more abstract. Here we assume that every chapter comprises of a statistical distribution of topics.
- The instructional concepts in the course chapters – Instructional concepts are the learning concepts that a learner would learn from a course chapter. For Example, a course chapter on grammar for Natural Language Processing could include instructional concepts like 'CFG grammar', 'Dependency grammar' etc. To understand the difference between Topics and instructional concepts; topics here are more abstract and statistically relevant keyword pattern which may or may not be explicitly understood by a human but the instructional

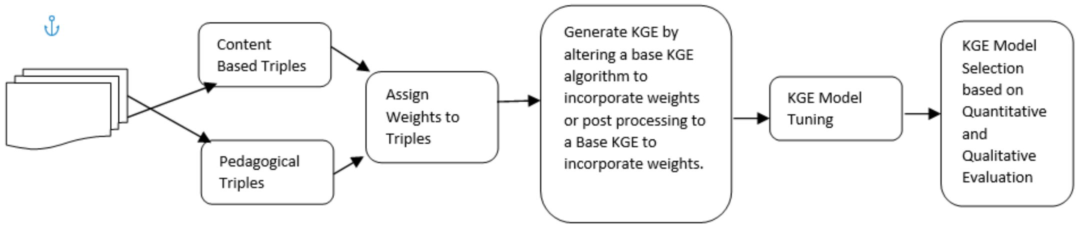
Content Based Triples
Pedagogical Triples
Assign Weights to Triples
Generate KGE by altering a base KGE algorithm to incorporate weights or post processing to a Base KGE to incorporate weights.
KGE Model Tuning
KGE Model Selection based on Quantitative and Qualitative Evaluation

Figure 2: EduEmbedd High Level Process Flow.

concepts are concepts that more tangible to understand and many of these instructional concepts are mentioned in the index of a text book. We can employ either or both of the below ways to extract instructional concepts (IC):

  - Build a master list of ICs by referring and extracting the ICs from the index section of a standard book on that subject and thereafter search the available ICs in the course chapter.
  - Leverage a pre-trained Large Language Model (LLM) like GPT to retrieve the ICs for the course chapter. Relevant prompt engineering needs to be leveraged to ensure the model understand the type of ICs to be retrieved by the LLM (for example ICs for Natural Language Processing).

- Complexity level of the course/chapters like Beginner, Intermediate, Expert etc. – This information can be found either using manual experts review or by leveraging a LLMs to aid us with this information by utilizing the course introduction or course synopsis to get the possible complexity level of the course. Prompt engineering using few shot learning was used to experiment this with LLMs. It showed mixed results with the limited experimentation that was conducted. For better consistency at this time, we leveraged expert manual review of a part of the course abstract and introduction to determine the complexity level.

3.2.2. Identification of Relationships

Relationships are identified based on all possible contexts which we have factored in the design of the KG. So in EduEmbedd, we have pedagogical and content based relationship information. Ideally there could be many possible relationship that could be imagined for the entities but some relationship may not be useful in the context of the entities and the use cases that we are targeting, it is best to ignore such relationship. For example if the corpus belongs to resources from a very specific stream (ex. engineering) then including this common relationship for every entity will not help much for the embedding and can be discarded.

Though there can be many possible relationships which could be used but we have only considered the following relationship in the current scope of EduEmbedd:

- `Text_topic`: This relationship links the course learning chapter(head) to the topic identifiers (tail) (topic based on the implementation of a topic modelling algorithm (LDA) on the learning content text data) found in the course chapters. This is a content type relationship.
- `Concept_vocab`: This relationship links the course Learning chapter(head) to the IC(tail) available in the chapters hence depicting the available IC in a course chapter. This is a content type relationship.
- `Prerequisite`: This relation links the course learning chapter(head) to the other course learning chapters(tail) those are considers as a pre-requisite. At this time, we have leveraged expert manual review of the course mandates and the learning curriculum to determine this. This is a pedagogical type relationship.
- `Level`: This relationship link the course learning chapter(head) to the complexity level(tail) of the course learning chapter. At this time, we have leveraged expert manual review of a part of the course abstract and introduction to determine the complexity level. This is a pedagogical type relationship.

3.2.3. Assign weights to the relationship edges

EduEmbedd factors weights given to the relationship edges to arrive at the resultant embeddings. The concept of weights play a vital role to highlight the confidence, probability, degree of the association (of a relationship) between the two entities. Different relationships requires different criteria to assign the weights. We have considered a combination of machine learning based process and expert reviews to arrive at the weights for the relationships (edge) and they are as follows:

- Text_topic: LDA for machine learning based topic modelling algorithm considers the notion of probability to arrive at the topics. The probabilistic model specifies a probability distribution over the k topics for each course chapter text document. Each word in the document is attributed to a particular topic with probability given by this distribution. So the topics are inherently defined as probability distributions over the vocabulary. The probablity scores acts as weights for this relationship type.
- Concept_vocab: The weights to this relationship specifies the notion of the relative strengths of the ICs in the course chapter document. Term Frequency Inverse Document Frequency (TF-IDF). It is a weighting system that assigns a weight to each word in a document based on its term frequency (tf) and the reciprocal document frequency (idf). The words with higher scores of weight are deemed to be more significant. The TF-IDF weights to the ICs in the document is considered as the weights for this relationship edge.
- Prerequisite: The weights for this relationship type is determined using an expect assessment mechanism meant to quantify the strength of the prerequisite course chapter which could aid in better understanding of the head node course chapter.
- Level: The weights for this relationship edge is determined using an expert assessment mechanism meant to quantify the complexity level and the content depth of the course chapter. It determines the extent to which the course chapter content impart the learning either as beginner, intermediate or at expert level. For example a course chapter could be classified as 80 percent Intermediate in complexity. In most cases if it is quite evident that the course chapter is fully at an beginner or intermediate or expert level then the weights could be given as 1.

3.2.4. Build the EduEmbedd KGE model

There are two primary considerations that we consider to build a suitable KGE model. The first is to consider building a KGE model from scratch and the second is to consider a standard boot strapped KGE model and fine tune this as per Eduembedd requirements.

From a KGE model perspective, the requirements for Eduembedd is to ensure that the model is able to comprehend 1-to-1 relationship, 1-to-many relationship and many-to-many relationships between the head and the tail entities. The model should be fairly interpretable as far as the scoring objective function is concerned. This is to enable us to efficiently work on updating the model (to add the notion of weights to the base model) as well as efficiently evaluate the model. Building a model from scratch would be required if the critical requirements are not being supported by the available base models. We reviewed various base KGE models before arriving at the base KGE model for EduEmbedd.

We studied various KGE models and decided to specifically evaluate compositional and non-compositional models or translations models. Compositional models elegant way to learn the characteristic functions of the relations in a knowledge graph, as they allow to cast the learning task as a problem of supervised representation learning. We specifically selected Holographic Embedding (HOLE) in this class of KGE.

For Translational KGE models, we understand that TransE is not capable of supporting 1-to-many and many-to-many type of relations and hence we looks into other translation models which supports this, like TransH and TransR etc. After reviewing the computational aspects of TransR we decided to select TransH as it met most of our base criteria.

A very extensive and thorough selection of base models is not in the scope of the current version of Eduembedd and is a work for future. Our endeavor for the current version is to arrive at a viable baseline for Eduembedd. We understand that the version of this KGE model would be developed using custom KG data where we do not have any prior performance baselines. Hence we were inclined to use a base model which would be fairly straightforward to interpret and evaluate based on the scoring objective function. This is required because we would need to functionally evaluate the model embeddings to verify the extent to which the model is able to capture the scoring objective function into its learning, to derive the embeddings. With this in mind, the translational type of KGE models comes as a natural choice because it is more straightforward to verify the extent to which the scoring objective function gets full-filled by the final embeddings (vector computation could mean adding a head to a relation should approximate to the relation's tail). TransH as a base KGE model is shortlisted as part of this understanding. We also wanted to practically experiment with models of different genre and which can be further modified to suit Eduembedd design with a reasonable effort to develop and test. HOLE fits well to this bracket.

A thorough evaluation between various KGE embedding models (including TransH and HOLE) was performed before arriving at TransH as the base KGE model on top of which we will further build EduEmbedd.

3.2.5. EduEmbedd Support for TransH edge weights

The base KGE models do not factor the numerical value as a weight to a relationship(edge) attribute. Though there are few multimodal KGE like LiteralE, TransEA etc. but none of them are primarily designed to support numerical value associated to edges. We have to build a version of TransH which supports numerical value associated to edges. We account for the edge weights by revamping the base TransH model such that the scoring layer which computes the TransH objective scoring function is able to account for the edge weights while computing the scoring and the loss function of the TransH network. For a practical implementation, we should be able to effectively change the base model (a neural network) to handle numerical weights by ensuring that the below components of a standard TransH model are revamped:

- `Scoring function` - Assign scores to Triples based on the TransH scoring function, high scores to positive triples and comparatively low scores to negative (corrupted) triples.
- `Loss function` - Optimize the embedding by maximizing the margin between positive and negative triples.
- `Optimization algorithm` - Margin-based ranking loss framework limits the scores of positive and negative triplets to have a suitable margin.
- `Regularization mechanism` - Initialize and compute regularization of the KGE neural network Tensors.
- `Initializer` - Entity and Relation embedding initialization functions.
- `Negatives generation strategy` - Corruptions are synthetic negative triples generated by a corruption generation layer that follows the protocol proposed in [Bordes et al., 2013]: we define a corruption of t as $\overline{t}$= (s; p; $\overline{o}$) or $\overline{t}$ = ($\overline{s}$; p; o) where $\overline{s}$; $\overline{o}$ are respectively subject or object corruptions (i.e. other entities randomly selected from E). We generate synthetic negatives by corrupting one side of the triple at a time to comply with the local closed world assumption [Nickel et al., 2016].

Let f(t) be the scoring function of a KGE model. In the case of TransE [Bordes et al., 2013] this is:

$$f(t) = -||e_{\mathrm{s}} + r_{\mathrm{p}} - e_{\mathrm{o}}||_{\mathrm{n}} \tag{1}$$

where e_s, r_p and e_o are the embeddings of the subject s, predicate p, and object o. We use a softplus non-linearity to make sure the scores returned by f(t) are greater or equal to zero, without introducing excessive distortion:

$$g(t) = (f(t)) = ln(1 + e^{\mathrm{f(t)}}) >= 0 \tag{2}$$

If we consider as the impact of the numerical edge weights then our scoring function becomes

$$h(t) = g(t) \tag{3}$$

Now the loss function becomes L is a modified, more numerically stable version of the negative log-likelihood of normalized softmax scores proposed in [Kadlec, 2017]:

$$L = -(log(e^{\mathrm{h(t^+)}}/(e^{\mathrm{h(t^+)}} + e^{\mathrm{h(t^-)}})) \tag{4}$$

Where, (t^+) are all positive triples and (t^-) are corrupted triples either of the subject or the object entity is corrupted. Through this updated training functions, we modulate the network output based on numeric values associated to triples. We leverage numeric weights associated to triples so that during training the model focuses on triples with higher numeric weights. We want our model to learn from training triples with high numeric weights, and at the same time use triple numeric weights to maximise the margin between scores assigned to true triples and those assigned to their corruptions. This increases the loss of the model and helps it focus on triples with higher weights.

3.2.6. Model Fine tuning and Final model selection

There are various critical hyperparameters which influences the quality of embeddings in EduEmbedd. These are:

- `Embedding dimension` - This depends on the amount of training data in our corpus and the number of entity features and many other factors beyond the scope of this work. In EduEmbedd we choose the embedding dimension empirically.
- `Batch size` - The number of training example in a batch also plays an important role. A value between 5-20 is fit empirically before arriving at a desirable value.
- `Learning rate` - The model learning rate is an important hyper-parameter and we try various combinations by increasing by a factor of 10 from the default value of 0.1.
- If early stopping is not configured then the number of epoch should also be tuned to ensure that the neural network is efficiently converging.

4. Experiments

We assess the predictive power of EduEmbedd by performing a combination of technical predictive evaluation and functional evaluation. The technical predictive evaluation focuses on the evaluation of the KGE model using link prediction. Here we use the commonly used KGE metrics:

- `Mean Rank` - The average of the ranks of all positive predictions.

- Mean Reciprocal Rank - The average of the reciprocal of the ranks of all positive predictions.
- Hits@1 - The fraction of positive predictions that rank better than all their negative predictions, i.e., have a rank of 1.
- Hits@3 - The fraction of positive predictions that rank in the top 3 among their negative predictions.
- Hits@5 - The fraction of positive predictions that rank in the top 5 among their negative predictions.
- Hits@10 - The fraction of positive predictions that rank in the top 10 among their negative predictions.

4.1. Technical Predictive Evaluation

We conducted this Evaluation with 3 different objective in our mind:

- Evaluate multiple models to select the best performing model.
- Evaluate the best set of hyper-parameters for a particular model.
- Relative evaluation of the model performance on out custom dataset.

Dataset: Custom data created using Open source educational courses featuring multiple courses from the Natural Language Processing field of education. The data spans over more than 300 lesson chapters in total. Based on the evaluation results Table 1 we found one of the version of TransH with weights performs comparatively better on most of the KGE predictive metrics. As this evaluation is conducted on a custom dataset we look forward more towards a comparative assessment of the models which could act as a viable baseline for future work. We also understand that improvements to the data, feature selection and base model would be critical to achieve future improvements to the Evaluation baseline scores.

Going further we also evaluate our best performing TransH_with_weights model to the base TransH (without weights) and observe Table 2 that the TransH_with_weights is performing even better than the base TransH on the various KGE predictive metrics.

4.2. Functional Evaluation of TransH with weights

This evaluation is important for the work because the technical predictive evaluation scores do not have any prior baselines and it would be hard to practically quantify an acceptable score achieved by a model on this data. So our objective here is to verify the effectiveness of the model in capturing the semantic aspect of our base data. For this, we constructed a test set of 22 similar entities taken from the base data and perform cosine similarity test. The average cosine similarity scores are mentioned in Table 3

This clearly shows that the revamped model of TransH with weights is able to create embedding which are more closer to the embeddings other similar entities. It shows that it is having comparatively better semantic prowess.

We also wanted to verify that incorporating the notion of weights is helping us to get model with better semantic understanding. We evaluated the functional testing of TransH with weights and compared with the version of TransH without weights. The average cosine similarity scores are mentioned in Table 4

This evaluation again points that having the notion of weights in our model is performing better in comparison to a model which does not account for weights.

Table 1
Predictive Evaluation of various models modified to handle edge weights

Model_name	epochs	k	lr	mrr	mr	hits_10	hits_5	hits_3	hits_1
transH_with_weights	50	40	0.1	0.205753	46.85977	0.389785	0.287634	0.219086	0.113351
transH_with_weights	100	40	0.1	0.187792	50.31586	0.364695	0.260305	0.193996	0.101703
transH_with_weights	50	50	0.1	0.20682	47.23477	0.40233	0.28853	0.219086	0.114247
transH_with_weights	100	50	0.1	0.187761	50.98477	0.364247	0.262993	0.196237	0.09991
transH_with_weights	50	40	0.01	0.130619	68.99283	0.254032	0.175627	0.138889	0.060484
transH_with_weights	100	40	0.01	0.11856	69.306	0.242832	0.166667	0.124552	0.051075
transH_with_weights	50	50	0.01	0.148806	66.39695	0.297043	0.203405	0.155466	0.073925
transH_with_weights	100	50	0.01	0.139172	65.30287	0.267473	0.183244	0.142473	0.0681
transE_with_weights	50	40	0.1	0.060491	148.4347	0.138544	0.083481	0.051954	0.019982
transE_with_weights	100	40	0.1	0.098372	97.52931	0.222025	0.145648	0.099023	0.036856
transE_with_weights	50	50	0.1	0.067951	147.4596	0.139876	0.088366	0.060835	0.026643
transE_with_weights	100	50	0.1	0.095743	95.56883	0.214032	0.141652	0.09103	0.035524
transE_with_weights	50	40	0.01	0.073888	155.9827	0.166075	0.102131	0.071936	0.027531
transE_with_weights	100	40	0.01	0.088706	113.3472	0.190053	0.129663	0.087922	0.03286
transE_with_weights	50	50	0.01	0.073352	153.3091	0.153197	0.094139	0.071492	0.028863
transE_with_weights	100	50	0.01	0.099445	112.1399	0.209591	0.138988	0.099467	0.042185
holE_with_weights	50	40	0.1	0.132954	76.07416	0.267762	0.186501	0.138988	0.061723
holE_with_weights	100	40	0.1	0.118447	69.79707	0.248668	0.167407	0.122114	0.047513
holE_with_weights	50	50	0.1	0.128366	78.98712	0.25222	0.171403	0.127886	0.062611
holE_with_weights	100	50	0.1	0.113633	79.7349	0.224245	0.15897	0.117673	0.049734
holE_with_weights	50	40	0.01	0.094671	104.1794	0.211812	0.139876	0.093694	0.034192
holE_with_weights	100	40	0.01	0.09835	83.54973	0.230018	0.152309	0.099911	0.033304
holE_with_weights	50	50	0.01	0.001254	800.0719	0	0	0	0
holE_with_weights	100	50	0.01	0.001254	800.0719	0	0	0	0

Table 2
Predictive Evaluation of TransH model with edge weights and vanilla TransH (without edge weights)

Model_name	epochs	k	lr	mrr	mr	hits_10	hits_5	hits_3	hits_1
transH_with_weights	50	40	0.1	0.205753	46.85977	0.389785	0.287634	0.219086	0.113351
transH	50	40	0.1	0.202403	47.34515	0.399013	0.280072	0.213196	0.106373

Table 3
Average cosine similarity scores for various models with edge weights)

Model_name	Average Cosine Similarity scores
Hole_with_weights	0.48
TransE_with_weights	0.33
TransH_with_weights	0.64

Table 4
Average cosine similarity scores for TransH models with edge weights and standard TransH (without edge weights)

Model_name	Average Cosine Similarity scores
TransH_with_weights	0.64
TransH_without_weights	0.49

References

[1] S. Wang, C. Liang, Z. Wu, K. Williams, B. Pursel, B. Brautigam, S. Saul, H. Williams, K. Bowen, C. L. Giles, Concept hierarchy extraction from textbooks, in: Proceedings of the 2015 ACM Symposium on Document Engineering, 2015, pp. 147–156.

[2] D. S. Chaplot, Y. Yang, J. Carbonell, K. R. Koedinger, Data-driven automated induction of prerequisite structure graphs., International Educational Data Mining Society (2016).

[3] C. Liang, J. Ye, Z. Wu, B. Pursel, C. Giles, Recovering concept prerequisite relations from university course dependencies, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 31, 2017.

[4] H. Liu, W. Ma, Y. Yang, J. Carbonell, Learning concept graphs from online educational data, Journal of Artificial Intelligence Research 55 (2016) 1059–1090.

[5] K. Academy, [online], 2017. URL: http://en.www.khanacademy.org/.

[6] P. Chen, Y. Lu, V. W. Zheng, X. Chen, B. Yang, Knowedu: A system to construct knowledge graph for education, Ieee Access 6 (2018) 31553–31563.

[7] M. Palmonari, P. Minervini, Knowledge graph embeddings and explainable ai, Knowledge Graphs for Explainable Artificial Intelligence: Foundations, Applications and Challenges 47 (2020) 49.

[8] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Advances in neural information processing systems 26 (2013).

[9] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: Proceedings of the AAAI conference on artificial intelligence, volume 28, 2014.

[10] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: Proceedings of the AAAI conference on artificial intelligence, volume 29, 2015.

[11] B. Yang, W.-t. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, arXiv preprint arXiv:1412.6575 (2014).

[12] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: International conference on machine learning, PMLR, 2016, pp. 2071–2080.

[13] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, arXiv preprint arXiv:1902.10197 (2019).

[14] M. Nickel, L. Rosasco, T. Poggio, Holographic embeddings of knowledge graphs, in: Proceedings of the AAAI conference on artificial intelligence, volume 30, 2016.

[15] M. Nickel, V. Tresp, H.-P. Kriegel, et al., A three-way model for collective learning on multi-relational data., in: Icml, volume 11, 2011, pp. 3104482–3104584.

[16] I. Balažević, C. Allen, T. M. Hospedales, Tucker: Tensor factorization for knowledge graph completion, arXiv preprint arXiv:1901.09590 (2019).

[17] S. M. Kazemi, D. Poole, Simple embedding for link prediction in knowledge graphs, Advances in neural information processing systems 31 (2018).

[18] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings, in: Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.

[19] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, D. Phung, A novel embedding model for knowledge base completion based on convolutional neural network, arXiv preprint arXiv:1712.02121 (2017).

[20] D. Nathani, J. Chauhan, C. Sharma, M. Kaul, Learning attention-based embeddings for relation prediction in knowledge graphs, arXiv preprint arXiv:1906.01195 (2019).

[21] A. Kristiadi, M. A. Khan, D. Lukovnikov, J. Lehmann, A. Fischer, Incorporating literals into knowledge graph embeddings, in: The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part I 18, Springer, 2019, pp. 347–363.

[22] A. García-Durán, M. Niepert, Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features, arXiv preprint arXiv:1709.04676 (2017).

[23] Y. Wu, Z. Wang, Knowledge graph embedding with numeric attributes of entities, in: Proceedings of The Third Workshop on Representation Learning for NLP, 2018, pp. 132–136.

[24] X. Chen, M. Chen, W. Shi, Y. Sun, C. Zaniolo, Embedding uncertain knowledge graphs, in: Proceedings of the AAAI conference on artificial intelligence, volume 33, 2019, pp. 3363–3370.

[25] A. Kimmig, S. Bach, M. Broecheler, B. Huang, L. Getoor, A short introduction to probabilistic soft logic, in: Proceedings of the NIPS workshop on probabilistic programming: foundations and applications, 2012, pp. 1–4.

[26] S. Pai, L. Costabello, Learning embeddings from knowledge graphs with numeric edge attributes, arXiv preprint arXiv:2105.08683 (2021).

[27] L. Costabello, S. Pai, C. L. Van, R. McGrath, N. McCarthy, P. Tabacof, AmpliGraph: a Library for Representation Learning on Knowledge Graphs, 2019. URL: https://doi.org/10.5281/zenodo.2595043. doi:10.5281/zenodo.2595043.