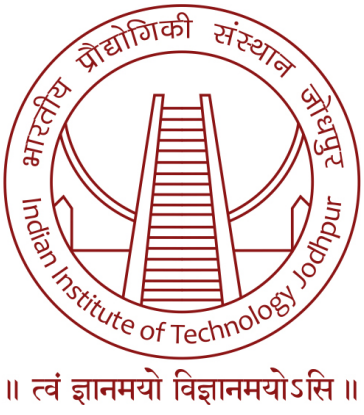
Multimodal Knowledge Graph Embeddings

A Project Report Submitted by

Parsa Revanth

in partial fulfillment of the requirements for the award of the degree of

Masters in Technology



Indian Institute of Technology Jodhpur

Computer Science and Engineering

May, 2022

Declaration

I hereby declare that the work presented in this Project Report titled Multimodal Knowledge Graph Embeddings, submitted to the Indian Institute of Technology Jodhpur in partial fulfilment of the requirements for the award of the degree of Masters in Technology, is a bonafide record of the research work carried out under the supervision of Dr. Anand Mishra. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.

Signature

Parsa Revanth

M20CS058

Certificate

This is to certify that the Project Report titled Multimodal Knowledge Graph Embeddings, submitted by Parsa Revanth (M20CS058) to the Indian Institute of Technology Jodhpur for the award of the degree of Masters in Technology, is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Signature

Dr. Anand Mishra

Acknowledgements

I would like to express my sincere gratitude to my project supervisor, Dr. Anand Mishra for his guidance through out the semester. I would also like to thank all my instructors for showing me a wonderful path forward and making me realise my potential. I would also like to thank all my friends and family members for their encouragement and support.

Abstract

With each passing day, knowledge is generated and its relation to existing knowledge is enriched and often there is a need to analyse this relationship. Link prediction in a knowledge graph is an effective and simple way for such knowledge graph completion or linking. Information in structured knowledge graphs is stored in form of triplets having a subject, predicate and an object. Thus, link prediction is to predict the object when we have the subject and the relation in the database, and this is a never-ending problem. Most knowledge graphs predominantly store information in the form of text. Link prediction on such graphs can be made even more effective using multimodal data. Thus in this project, we explore the use of multimodal information of images and text in knowledge graphs to study the link prediction task. My contributions are detailed survey of knowledge graph embeddings techniques and their implementations and exploring few of knowledge graph embedding techniques for multimodal data on two datasets.

Contents

List of Figures

List of Tables

Multimodal Knowledge Graph Embeddings

1 Introduction

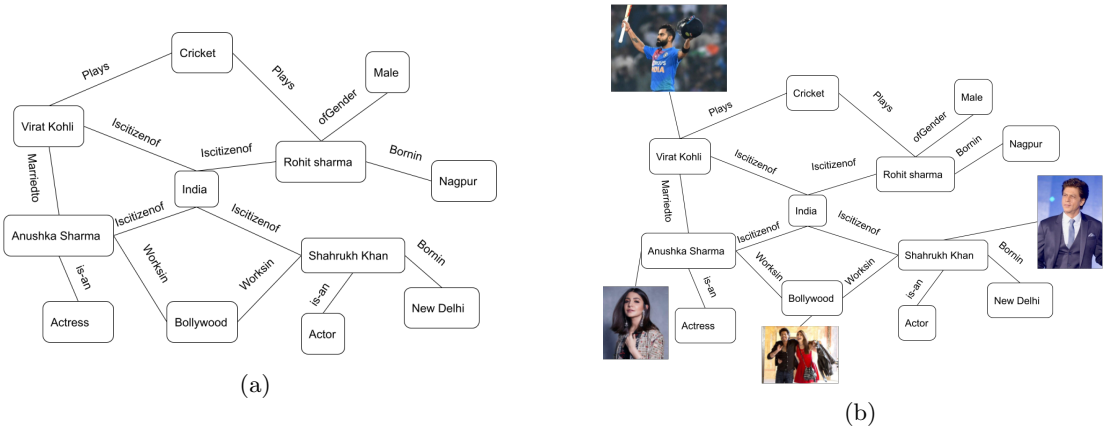


Figure 1.1: **Illustration of textual and multimodal knowledge graphs. (a) An example of knowledge graph where information is stored in text. (b) An example of a multimodal knowledge graph with images and text.**

The knowledge graph is a network of real-world entities and the relations between those entities. The objective of this project is perform the link prediction task on a multimodal knowledge graph, which entails the prediction of object entity when subject entity and its relation is provided. The link prediction task is to predict the object entity when we have the information of the subject entity and relation.

In order to perform link prediction, we need to have the information processed in the form of vectors known as embeddings. The embeddings are the low dimensional representations of the entities and the relations in the knowledge graph. In knowledge graphs the entities are texts only and adding multimodal information like images, text description improves the context and explainability of inferences.

We can make some inferences on the above-shown example knowledge graph in Figure 1.1a but in order to make a complex inference, for example, to predict the gender of Shahrukh khan, it is not a trivial task. If we add the image of the Shahrukh khan into the knowledge graph then the inference will become easier. Another example that requires multimodal information to make inference easier, if we want to predict the movie in which both Shahrukh Khan and Anushka Sharma acted, given the example knowledge graph we cannot predict but if we add an image in which they acted together, then the model can make such a prediction as the embeddings have the image information. In this way adding multimodality can be complementary to make some complex inferences easier.

2 Survey of Knowledge Graph Embeddings

In a knowledge graph, data is stored in the form of triplets that is subject, predicate, and object. According to a survey on knowledge graph embeddings [21], the main types of embedding models can be categorized as translational based, matrix-factorization based, and neural network-based. I would like to define some notations, a triplet is *(subject, relation, object)*, subject also known as head is represented by h, relation is represented by r, object also known as tail is represented by t. If we consider the triplet *(Virat Kohli, Playsfor, India)*, Virat Kohli and India are entities and Playsfor is a relation. *Virat Kohli* is head(h), *Playsfor* is relation(r) and *India* is tail(t). The positive sample is the correct triplet and the negative sample is the triplet where either head or tail or both corrupted. Manhattan distance is represented as *l1* and euclidean distance is represented as *l2*.

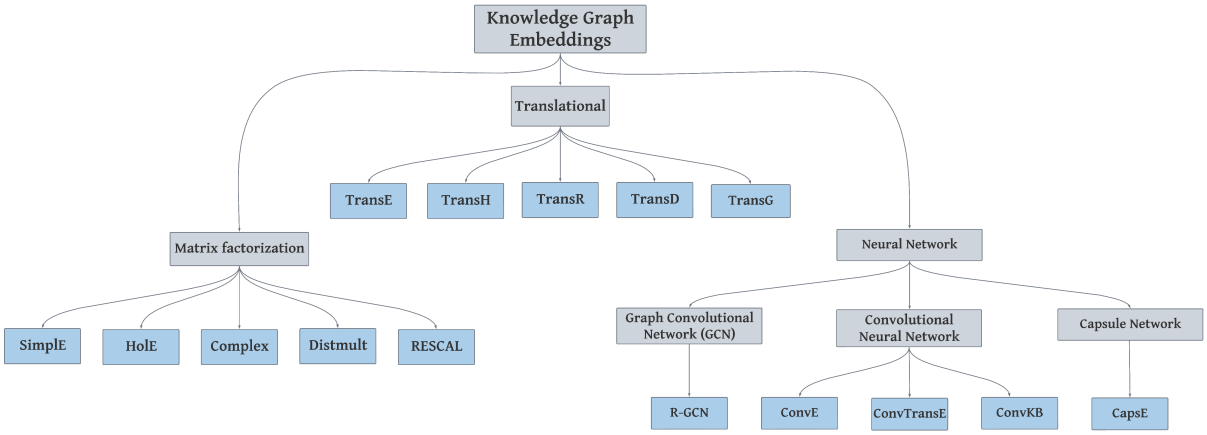


Figure 2.1: **Knowledge graph embeddings mentioned in the figure are TransE [1], TransH [2], TransR [3], TransD [4], TransG [5], Distmult [6], RESCAL [7], ComplEx [8], HolE [9], SimplE [10], R-GCN [11], ConvE [12], ConvTransE [13], ConvKB [14], CapsE [15].**

2.1 Translational based Embeddings

The idea of TransE [1], that is using the distance between two entities is inspired by the Word2vec [22]. In Word2vec the distance between king and male is almost equal to the distance between queen and female. By this the idea of the distance between similar entities will be similar to the relation among them is proposed that is the manhattan distance or Euclidean distance between the subject entity and relation to the object entity is almost zero for a triplet in the knowledge graph. In TransE [1] the embeddings are uniformly initialized for entities and relations. With the help of pairwise margin-based hinge loss between a positive sample and a negative sample, the embeddings are updated. The score is calculated using the TransE [1] scoring function. The TransE [1] score is calculated using equation 1

$$f_r(h,t) = \|h + r - t\|_{l1/l2}. \tag{1}$$

Some of the variants of TransE [1] which are also used as translational-based embeddings are TransR [3], TransH [2], TransD [4], TransG [5]. TransH [2] extends the TransE [1] model by generating embeddings

for each entity specific to each relation hyperplane.

$$\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r, \mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r \tag{2}$$

TransH [2] solves the problem of complex relations by using the relation-specific hyperplane. The entities are projected onto the relation-specific hyperplane using the normal vector w_r, which is mathematically shown in equation 2. The translational distance between the entities in the relation-specific hyperplane uses d_r as the translational distance. The translation distance d_r is used in calculating the TransH [2] score, which is given by the equation 3

$$f_r(h,t) = \|\mathbf{h}_\perp + d_r - \mathbf{t}_\perp\|_{l2}. \tag{3}$$

TransR [3] extends the TransH [2] model by generating embeddings for entity-relation vectors spaces. Some entities might have different semantic meanings and different relations with the same subject entity can lead to the different tail entities. So to overcome this TransR [3] extends the idea of TransH [2] by converting the entities and relations into entity-relation spaces, where entities are mapped to entity-specific space and relations are mapped to relation-specific space. The subject entity h and tail entity t are embedded in entity-specific space of R^d and relation r is embedded in relation-specific space of R^k. The entities embedded in entity-specific space are then projected to the relation-specific space using the projection matrix M_r.

$$\mathbf{h}_\perp = \mathbf{M}_r\mathbf{h}, \mathbf{t}_\perp = \mathbf{M}_r\mathbf{t} \tag{4}$$

$h_\perp$ is the subject entity projected onto the relation-specific space and $t_\perp$ is the tail entity projected onto the relation-specific space, which can be mathematically shown in the equation 4. The TransR [3] score is calculated using the equation 5

$$f_r(h,t) = \|\mathbf{h}_\perp + r - \mathbf{t}_\perp\|_{l2}. \tag{5}$$

The projection matrix proposed in the TransR [3] is the matrix related to relation. TransD [4] extends on that idea by creating entity-specific dynamic projection matrices using the two different vectors $h \in R^d$, $t \in R^d$ and $r \in R^k$ are one type of vectors. $h^{'} \in R^d$, $t^{'} \in R^d$ and $r^{'} \in R^k$ are another type of vectors which are used to form the projection matrices, which can be mathematically shown in equation 6.

$$M_{rh} = \mathbf{r}^{'}\mathbf{h}^{'\top} + \mathbf{I}, M_{rt} = \mathbf{r}^{'}\mathbf{t}^{'\top} + \mathbf{I}. \tag{6}$$

$M_{rh} \in R^{kxd}$ and $M_{rt} \in R^{kxd}$ represents the head entity projection matrix and tail entity projection matrix respectively and $I \in R^{kxd}$. TransD [4] score is calculated using the equation 8.

$$\mathbf{h}_\perp = \mathbf{M}_{rh}\mathbf{h}, \mathbf{t}_\perp = \mathbf{M}_{rt}\mathbf{t}. \tag{7}$$

Table 2.1: **Translational based embedding models.**

Embedding technique	Scoring function
TransE [1]	$\|h + r - t\|_{l1/l2}$
TransH [2]	$\|(\mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r) + w_r - (\mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r)\|_{l2}$
TransR [3]	$\|(\mathbf{M}_r\mathbf{h}) + r - (\mathbf{M}_r\mathbf{t})\|_{l2}$
TransD [4]	$\|(\mathbf{M}_{rh}\mathbf{h}) + r - (\mathbf{M}_{rt}\mathbf{t})\|_{l2}$
TransG [5]	$\sum_{m=i}^{M_r} \pi_{r,m} \exp\left(-\frac{\|\mu_h + \mu_r^i - \mu_t\|_2^2}{\sigma_h^2 + \sigma_t^2}\right)$

$$f_r(h,t) = \|\mathbf{h}_\perp + r - \mathbf{t}_\perp\|_{l2}. \tag{8}$$

To take the uncertainties in entities and relations into account Gaussian embedding model TransG [5] is proposed. TransG [5] considers the entities and relations as the random vectors taken from the Gaussian distribution, mathematically shown in the equation 9 and calculates the scores of the triplets between these random vectors. The score is calculated using the equation 10.

$$\begin{aligned} \mathbf{h} &\sim \mathcal{N}\left(\mu_h, \sigma_h^2\mathbf{I}\right), \mathbf{r} \sim \mathcal{N}\left(\mu_r, \sigma_r^2\mathbf{I}\right), \\ \mathbf{r} &\sim \sum_{m=1}^{M} \pi_{r,m}\mathcal{N}\left(\mu_h - \mu_t, \left(\sigma_h^2 + \sigma_t^2\right)\mathbf{I}\right). \end{aligned} \tag{9}$$

$$f_r(h,t) = \sum_{m=i}^{M_r} \pi_{r,m} \exp\left(-\frac{\left\|\mu_h + \mu_r^i - \mu_t\right\|_2^2}{\sigma_h^2 + \sigma_t^2}\right). \tag{10}$$

2.2 Matrix factorization based Embeddings

The triplets are represented in the form of binary matrices each corresponding to a relation. The entries in the matrix corresponding to relation k is 1 if the head entity and tail entity have a relation k else the entry will be 0. If the knowledge graph has p entities and q relations, then each binary matrix is represented as $\mathbf{Z}_k$ where where k = {1,2,3,...,q} which is specific to each relation and $\mathbf{Z}_k \in R^{pxp}$. RESCAL [7] uses rank-d decomposition on the binary matrices $\mathbf{Z}_r$, mathematically shown in equation 11.

$$\mathbf{Z_r} = \mathbf{A}\mathbf{M_r}\mathbf{A}^\top. \tag{11}$$

$\mathbf{A} \in R^{pxd}$ represents the embeddings of the entities in the knowledge graph. $\mathbf{M}_r \in R^{dxd}$ represents the latent semantic meaning in the relations. RESCAL [7] calculates the score using the equation 12. $\mathbf{h}, \mathbf{t} \in R^d$ represents the entity embeddings which are the rows corresponding to the entities in matrix $\mathbf{A}$

$$f_r(h,t) = \mathbf{h}^\top \mathbf{M_r}\mathbf{t}. \tag{12}$$

In Distmult [6], we represent the triplets in the form of binary matrices each corresponding to the relation. Distmult [6] is RESCAL [7] with a change in the matrix $\mathbf{M}_r$ matrix. In Distmult [6] $\mathbf{M}_r$ is a

Table 2.2: **Matrix Factorization based embedding models.**

Embedding technique	Scoring function
RESCAL [7]	$\mathbf{h}^\top \mathbf{M_r t}$
Distmult [6]	$\mathbf{h}^\top \mathbf{diag(r)t}$
ComplEx [8]	$\mathbf{Re(h}^\top \mathbf{diag(r)t}^{'})$
HolE [9]	$\mathbf{r}^\top(\mathbf{h} * \mathbf{t})$
SimplE [10]	$\frac{1}{2}(\mathbf{h} \circ \mathbf{rt} + \mathbf{t} \circ \mathbf{r}^{'}\mathbf{h})$

diagonal matrix. Here also we perform the rank-based factorization to generate a matrix $\mathbf{A}$ and matrix $\mathbf{M}_r$. The matrix $\mathbf{M}_r$ is equal to $\mathbf{diag(r)}$, $\mathbf{r} \in R^d$. The Distmult [6] score is calculated using the equation 13

$$f_r(h,t) = \mathbf{h}^\top \mathbf{diag(r)t}. \tag{13}$$

In Distmult [6] the symmetric relations can be handled which are symmetric with respect to head and tail entities and inorder to handle the asymmetric relations ComplEx [8] embedding technique is proposed. In ComplEx [8] the relations and entities belongs to complex number space $\mathbf{C}^d$ rather than real number space $\mathbf{R}^d$. The scoring function for the ComplEx [8] embedding technique is given by equation 14. $\mathbf{t}^{'}$ is the complex conjugate of the tail entity $\mathbf{t}$.

$$f_r(h,t) = \mathbf{Re(h}^\top \mathbf{diag(r)t}^{'}). \tag{14}$$

To capture the compositional representation between the entities, a HolE [9] method was proposed by introducing the circular correlation operation. The scoring function for the HolE [9] embedding technique is given by the equation 15.

$$f_r(h,t) = \mathbf{r}^\top(\mathbf{h} * \mathbf{t}). \tag{15}$$

SimplE [10] makes use of the Canonical Polyadic (CP) decomposition, which helps in computing the score by introducing the reverse of relations. The score is taken as the average CP score of (h, r, t) and (t, r^{-1}, h). The scoring function for the SimplE [10] embedding technique is given by the equation 16.

$$f_r(h,t) = \frac{1}{2}(\mathbf{h} \circ \mathbf{rt} + \mathbf{t} \circ \mathbf{r}^{'}\mathbf{h}). \tag{16}$$

2.3 Neural Network based Embeddings

Based on the type of the neural network architecture there are three types of embeddings, which are convolutional neural network-based, graph convolutional network-based, and capsule network-based. Convolutional neural network-based are ConvE [12], ConvKB [14], and ConvTransE [13]. Capsule network-based is CapsE [15]. Graph convolutional network-based is R-GCN [11].

2.3.1 Convolutional neural network-based Embeddings

In ConvE [12], the embeddings are first uniformly initialized with d dimension, and the subject embeddings or head and the relation embeddings are concatenated and reshaped. Secondly, the concatenated embeddings are passed onto one or more convolutional layers with w filters thereby generating the feature maps of these embeddings. Thirdly, feature maps are passed through a dense layer where they are multiplied with the weight matrix W. Finally, the feature maps are multiplied with the tail embeddings using the dot product. The final output will be the ConvE [12] score and if we pass the entity matrix then the output passed upon the softmax layer gives the probability of head and relation corresponding tail entity. Mathematically shown in the equation 17.

$$f_r(h,t) = f(vec(f([h;r] * w))W)t. \tag{17}$$

In ConvKB [14], the embeddings are uniformly initialized. Firstly, the head h, the relation r, and the tail t embeddings are concatenated with each taken as a row of the matrix thereby resulting in the matrix of dimension $dx3$. Secondly, the concatenated embeddings are passed onto one or more convolutional layers with w filters thereby generating the feature maps of these embeddings. Thirdly, feature maps are passed through a dense layer where they are multiplied with the weight matrix W. The output will be the ConvKB [14] score. Mathematically shown in the equation 18.

$$f_r(h,t) = f(vec(f([h;r;t] * w))W). \tag{18}$$

ConvTransE [13] is the fusion of ConvE [12] and TransE [1] which enables ConvE [12] to retain its link prediction performance and be translational between entities and relations. In ConvTransE [13], unlike ConvE [12] there is no reshaping of the concatenated embeddings of head and relation. Firstly, the head and relation embeddings are passed to the set of kernels, and convolutions between the head and relation embeddings are performed to generate feature maps upon performing vectorization feature maps are converted to a vector. A matrix is constructed by aligning the outputs of all the kernels $M_{(h,r)}$. Finally, the feature maps are multiplied with the tail embeddings using the dot product and if we pass the entity matrix then the output is passed to the logistic sigmoid function. The final output will be the ConvTransE [13] score. Mathematically shown in the equation 19.

$$f_r(h,t) = f(vec(M_{(h,r)}))W)t. \tag{19}$$

2.3.2 Graph convolutional network-based Embeddings

Graph convolutions networks are extended to relational databases and analyzed in on the link prediction task. Those graphs are known as R-GCN [11]. R-GCN is an improved model, which provides relation-specific transformation to represent knowledge graphs. Using the equation 20 the information from each layer is passed onto the next layer and so on until the final layer. The embedding h_0 for each node or

Table 2.3: **Neural network based embedding models.**

Embedding technique	Scoring function
ConvE [12]	$f(vec(f([h;r] * w))W)t$
ConvKB [14]	$f(vec(f([h;r:t] * w))W)$
ConvTransE [13]	$f(vec(M_{(h,r)}))W)t$
R-GCN [11]	$(\mathbf{q_i^{(l+1)}})^\top \mathbf{Rt}$
CapsE [15]	$\|\|(capsnet(f[h;r:t] * k))\|\|$

entity is uniformly initialized and later using the $q_i^{(l+1)}$ of the last layer are taken as the embeddings of the entities in the graph.

$$q_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}q_i^{(l)}\right). \tag{20}$$

R-GCN [11] is used to only generate the embeddings representations and for the link prediction task the R-GCN [11] embeddings are modified using the Distmult [6] score. For link prediction task the equation to calculate the score is 21. R is the diagonal matrix of Distmult [6] model. Let the last layer value be $h = q_i^{(l+1)}$.

$$f_r(h,t) = \mathbf{h}^\top \mathbf{Rt}. \tag{21}$$

2.3.3 Capsule network-based Embeddings

In CapsE [15], Firstly, the head h, the relation r, and the tail t embeddings are concatenated with each taken as a row of the matrix thereby resulting in the matrix of dimension $dx3$. Secondly, the concatenated matrix is passed through a convolutional layer with k 1x3 filters which generates a matrix of size $d \times k$. Thirdly, the matrix is passed to the capsule layer, where each column is handled by each capsule, and finally passing through another capsule layer with one capsule generates the CapsE [15] score. Mathematically shown in the equation 22.

$$f_r(h,t) = ||(capsnet(f[h;r;t] * k))||. \tag{22}$$

3 Problem definition and Objective

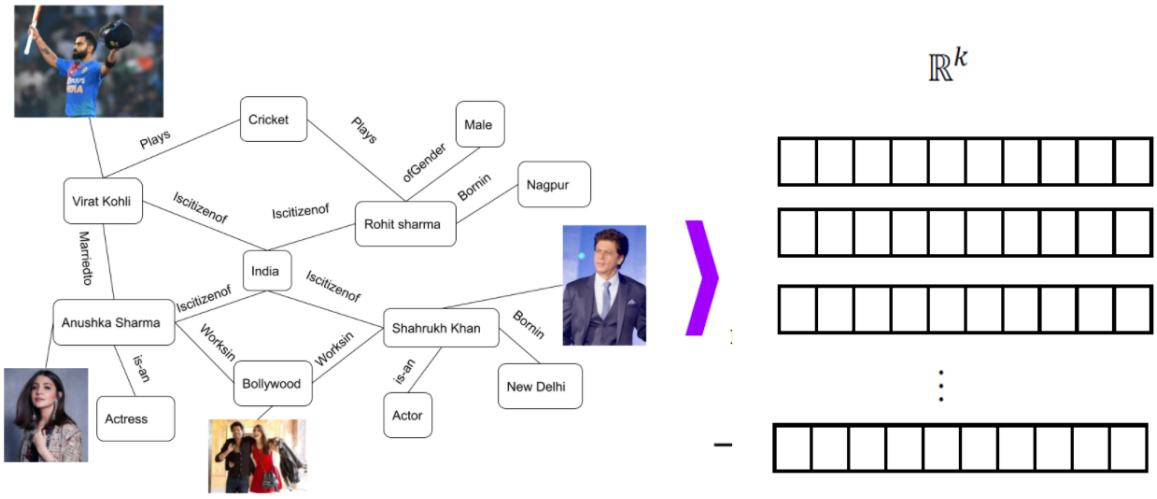


Figure 3.1: **Figuratively showing the objective of the project that is to generate embeddings from the multimodal knowledge graph.**

The objective of this project is to learn the low-dimensional representations of entities and the relations in a knowledge graph in-order to link prediction tasks on the multimodal knowledge graph. Some entities in the above-shown knowledge graph in Figure 3.1 are 'Virat Kohli', 'India', 'Shahrukh Khan', 'Actress', 'Cricket', 'New Delhi', 'Nagpur', and 'Male'. Some relations are 'plays', 'is-an', 'bornin', 'iscitizenof'. For all these entities and relations, we generate the embeddings. In order to generate, we uniformly initialize the embeddings and train the embeddings using the margin-based ranking loss function. After generation, we perform the link prediction task. We evaluate the outcome of the link prediction using metrics mean rank(MR) [23],mean reciprocal rank(MRR) [24], hits@1 [25], hits@3 [25], hits@5 [25] and hits@10 [25].

4 Methodology

The architecture used in the project is shown in Figure 4.1. The architecture is based on [23], which explores embeddings for multimodal relation data. In order to incorporate the image and text information, we first process the data. If we have an image for the entity, we generate its image embedding using a pre-trained neural network that uses Resnet50 [16]. While if we have a text description for entity, we then use S-Bert [17] to generate the its text embedding. After that, we pass those embeddings through linear encoders, which decreases the size of the embedding. We used a score from the TransE [1] model to train the embeddings. Currently, we explore two different models which are entity only model and the entity and image model, labelled as Entity and Entity + Image respectively

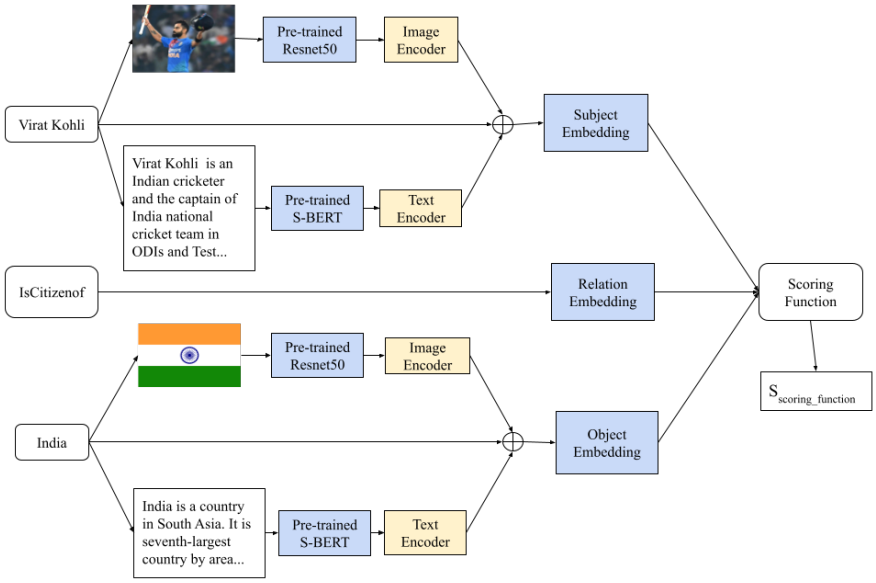


Figure 4.1: **Architecture for multimodal embedding generation. The model takes input triplets and process each entity and relation information. If the model has images it will be passed through Resnet50 [16] to generate embeddings and if they have text description it will be passed to S-Bert [17] to generate embeddings. After that, each embeddings information is accumulated and passed to a scoring function which generates a score.**

S.No	Dataset	Entities	Relations	Triplets			Images
				Train	Test	Valid	
1	FB15k-237 [18]	14505	237	272115	17535	20466	12237
2	WN18RR [12]	40943	11	86835	3034	3134	-
3	YAGO3-10 [19]	123143	37	1079040	5000	5000	61223

Table 5.1: **Table describing the number of entities, relations, triplets, and images in the datasets.**

4.1 Entity model

In this model, we use only the entity information of all entities and the relations and perform the embedding generation using TransE [1], TransH [2], TransR [3], TransD [4], ComplEx [8], Distmult [6], SimplE [10] scoring functions.

4.2 Entity and image model

In this model, we use the entity information of all entities and the relations and also use the image information for the entities which have images in the database. Later perform the embedding generation using TransE [1] and TransH [2] scoring functions.

5 Experiments

5.1 Datasets

The datasets used in this work are FB15k-237 [18], WN18RR [12], YAGO3-10 [19]. From table 5.1 we can understand that there are 14,505 entities and 237 relations are present in the FB15k-237 [18] dataset. FB15k-237 [18] dataset is subset of FB15k [1] dataset. FB15k suffers from the problem of having almost identical relations or inverse relations. FB15k-237 [18] is built by removing all such equivalent and inverse relations. The image dataset is curated using the URLs provided by the authors on the Github site of Onoro-Rubio et al. [24]. The total number of images downloaded is 12237. From table 5.1 we can understand that there are 40943 entities and 11 relations present in the WN18RR [12] dataset. WN18RR [12] is a subset of the WN18 dataset and was also proposed in Dettmers et al. [12]. WN18 [1] is derived from the Wordnet. From table 5.1 we can understand that there are 123143 entities and 37 relations are present in the YAGO3-10 [19] dataset. YAGO3-10 [19] was built from the YAGO3 KG [19] in Dettmers et al. [12]. The image dataset is used from the Pezeshkpour et al. [23]. The total number of images downloaded is 61223. Descriptive properties such as gender or citizenship are the major part of the data.

5.2 Evaluation Metrics

$rank_{(h,r,t)i}$ represents the ranks of the i^{th} triplet. Q represents the total number of triplets.

1. Mean Rank (MR): The average of the obtained ranks, mathematically shown in equation 23

$$MR = \frac{1}{Q}\sum_{i=1}^{Q} rank_{(h,r,t)i}. \tag{23}$$

2. Mean Reciprocal Rank (MRR): The average of the inverse of the obtained ranks, mathematically shown in equation 24

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q} \frac{1}{rank_{(h,r,t)i}}. \tag{24}$$

3. Hits@K: It is the total ranks predicted below or equal to rank k upon the total predicted ranks, mathematically shown in equation 25

$$Hits@K = \sum_{i=1}^{Q} \frac{rank_{(h,r,t)i} : rank_{(h,r,t)i} \leq K}{rank_{(h,r,t)i}}. \tag{25}$$

5.3 Implementation

I implemented four translational embedding techniques TransE [1], TransH [2], TransR [3], and TransD [4], three matrix factorization-based embedding techniques Distmult [6], ComplEx [8], and SimplE [10], two neural network-based embedding techniques ConvE [12], and R-GCN [11] for the entity only data. For the entity and image data, I implemented TransE [1] and TransH [2] embedding techniques. I used

Pykeen and OpenKE libraries to implement the embedding techniques. To implement and create the knowledge graph for R-GCN [11] using WN18RR [12] and YAGO3-10 some changes are made in the DGL library file. As the authors of the R-GCN [11] have implemented on the FB15k-237 [18] dataset there are no changes made in the DGL for that dataset. For entity-only data, I used the OpenKE library to implement TransE [1], TransH [2], TransR [3], TransD [4], Distmult [6], ComplEx [8], and SimplE [10].

For implementing the following translation-based embedding techniques on the entity-only data, I trained the models for 2000 epochs, the loss function is pairwise margin-based loss with a margin of 5.0, and the optimizer is SGD and used manhattan distance metric to calculate their respective scores. For the implementation of TransE [1] on both the FB15k-237 [18] dataset and on the WN18RR [12] dataset, I used an embedding dimension of 200 and for the implementation of TransE [1] on the YAGO3-10 [19] dataset, I used embedding dimensions {50, 100, 200, 300}. For the implementation of TransH [2] on both the FB15k-237 [18] dataset and on the WN18RR [12] dataset, I used an embedding dimension of 200 and for the implementation of TransH [2] on the YAGO3-10 [19] dataset, I used embedding dimensions {50, 100, 200, 300}. For the implementation of TransR [3] on both the FB15k-237 [18] dataset and on the WN18RR [12] dataset, I used an embedding dimension of 200 for both entities and relations, and for the implementation of TransR [3] on the YAGO3-10 [19] dataset, I used embedding dimension 300 for both entities and relations. For the implementation of TransD [4] on both the FB15k-237 [18] dataset and on the WN18RR [12] dataset, I used an embedding dimension of 200 for both entities and relations, and for the implementation of TransD [4] on the YAGO3-10 [19] dataset, I used embedding dimension 300 for both entities and relations.

For implementing the following matrix factorization techniques on the entity-only data, I trained the models for 2000 epochs, using the soft plus loss function and Adagrad [25] optimizer. For the implementation of Distmult [6] on both the FB15k-237 [18] dataset and on the WN18RR [12] dataset, I used an embedding dimension of 200 and for the implementation of Distmult [6] on the YAGO3-10 [19] dataset, I used embedding dimensions {50, 100, 200, 300}. For the implementation of ComplEx [8] on both the FB15k-237 [18] dataset and on the WN18RR [12] dataset, I used an embedding dimension of 200 and for the implementation of ComplEx [8] on the YAGO3-10 [19] dataset, I used embedding dimensions {50, 100, 200, 300}. For the implementation of SimplE [10] on both the FB15k-237 [18] dataset and on the WN18RR [12] dataset, I used an embedding dimension of 200 and for the implementation of SimplE [10] on the YAGO3-10 [19] dataset, I used an embedding dimension of 300.

For the implementation of ConvE [12] on the FB15k-237 [18] dataset using entity-only data, I used embedding dimension of {100, 200}, trained for epochs {100, 500}, the optimizers used are Adam [20] optimizer and Adagrad [25] optimizer. For the implementation of ConvE [12] on the WN18RR [12] dataset using entity-only data, I used embedding dimension of {100, 200}, trained for epochs {100, 500, 1000, 2000}, the optimizer is Adam [20] optimizer. For the implementation of ConvE [12] on the YAGO3-10 [19] dataset using entity-only data, I used the embedding dimension of {100, 200}, trained for epochs {100, 1000}, the optimizer is Adam [20] optimizer. For the implementation of R-GCN [11] on the FB15k-237 [18] dataset, WN18RR [12] dataset, and YAGO3-10 [19] dataset using entity-only data, I used an

S.No	Dataset	Embedding Technique	Embedding Dimension	Epochs	MRR	MR	Hits @ (%)		
							10	3	1
1	FB15k-237 [18]	TransE [1]	200	2000	0.29	**228.57**	47.73	32.71	19.28
2		TransH [2]			**0.30**	242.09	48.59	33.55	20.39
3		TransR [3]			**0.30**	309.01	**48.70**	**34.09**	**21.29**
4		TransD [4]			**0.30**	235.24	48.36	33.32	19.96

Table 5.2: **Comparison of the mean reciprocal rank(MRR), mean rank(MR), hits@1, hits@3, and hits@10 using the translational embeddings TransE [1], TransH [2], TransR [3], TransD [4] performed the experiments using embedding dimensions 200 and epochs of 2000 for the entity-only model. The experiments are performed on the FB15k-237 [18] dataset.**

S.No	Dataset	Embedding Technique	Embedding Dimension	Epochs	MRR	MR	Hits @ (%)		
							10	3	1
1	FB15k-237 [18]	Distmult [6]	200	2000	0.18	395.28	34.31	20.01	9.94
2		ComplEx [8]			**0.24**	566.82	**40.94**	**26.82**	**14.88**
3		SimplE [10]			0.19	**369.44**	36.12	21.30	10.14

Table 5.3: **Comparison of the mean reciprocal rank(MRR), mean rank(MR), hits@1, hits@3, and hits@10 using the matrix factorization-based embeddings Distmult [6], ComplEx [8], and SimplE [10] performed the experiments using embedding dimensions 200 and epochs of 2000 for the entity-only model. The experiments are performed on the FB15k-237 [18] dataset.**

embedding dimension of 100, trained for 6000 epochs, the optimizers used are Adam [20] optimizer.

For implementing the following translation-based embedding techniques on the entity-image data, the optimizer used is SGD optimizer, regularization is done with a regularizer of 0.01 and used Euclidean distance metric to calculate their respective scores. For the implementation of TransE [1] on the FB15k-237 [18] dataset using entity-image data, I used embedding dimension of {50, 350}, trained for epochs {100, 820} and for the implementation of TransE [1] on the YAGO3-10 [19] dataset using entity-image data, I used embedding dimension of {50, 350}, trained for epochs {100, 207}. For the implementation of TransH [2] on the FB15k-237 [18] dataset using entity-image data, I used the embedding dimensions of {200, 350}, trained for 100 epochs. For the implementation of TransH [2] on the YAGO3-10 [19] dataset using entity-image data, I used an embedding dimension of 300, trained for 100 epochs.

5.4 Results

From the table 5.2 we can infer that for the FB15k-237 [18] dataset, TransR [3] is the best embedding technique among the translational-based embeddings techniques TransE [1], TransH [2], TransR [3], and TransD [4]. MRR 24 is 0.29 for TransE [1] and 0.3 for the remaining. MR 23 is best for the TransE [1] which is 228.57. Hits@10 25, Hits@3, Hits@1 are 48.70, 34.09, and 21.29 respectively for the TransR [3] are the best among the TransE [1], TransH [2], TransR [3], and TransD [4].

From the table 5.3 we can infer that for the FB15k-237 [18] dataset, ComplEx [8] is the best embedding technique among the matrix-based embeddings techniques Distmult [6], ComplEx [8], and SimplE [10]. MRR 24 is best for ComplEx [8] which is 0.24. MR 23 is best for the SimplE [10] which is 369.44. Hits@10 25, HIts@3, Hits@1 are 40.94, 26.82, and 14.88 respectively for the ComplEx [8] are the best among the

S.No	Dataset	Embedding Technique	Embedding Dimension	Epochs	MRR	MR	Hits @ (%)		
							10	3	1
1	WN18RR [12]	TransE [1]	200	2000	**0.20**	5223.35	**45.51**	37.24	0.22
2		TransH [2]			**0.20**	5116.50	45.42	**37.68**	0.21
3		TransR [3]			0.18	7453.44	39.52	36.18	**0.59**
4		TransD [4]			**0.20**	**4933.26**	45.42	37.38	0.24

Table 5.4: **Comparison of the mean reciprocal rank(MRR), mean rank(MR), hits@1, hits@3, and hits@10 using the translational embeddings TransE [1], TransH [2], TransR [3], TransD [4] performed the experiments using embedding dimensions 200 and epochs of 2000 for the entity-only model. The experiments are performed on the WN18RR [12] dataset.**

S.No	Dataset	Embedding Technique	Embedding Dimension	Epochs	MRR	MR	Hits @ (%)		
							10	3	1
1	WN18RR [12]	Distmult [6]	200	2000	0.32	**4012.12**	46.09	36.57	23.87
2		ComplEx [8]			**0.39**	4531.49	**47.52**	**42.17**	**33.77**
3		SimplE [10]			0.35	4399.46	45.33	39.04	28.65

Table 5.5: **Comparison of the mean reciprocal rank(MRR), mean rank(MR), hits@1, hits@3, and hits@10 using the matrix factorization-based embeddings Distmult [6], ComplEx [8], and SimplE [10] performed the experiments using embedding dimensions 200 and epochs of 2000 for the entity-only model. The experiments are performed on the WN18RR [12] dataset.**

Distmult [6], ComplEx [8], and SimplE [10].

From the table 5.4 we can infer that for the WN18RR [12] dataset, MRR 24 is 0.18 for TransR [3] and 0.20 for the remaining. MR 23 is best for the TransE [1] which is 4933.26. Hits@10 25 are 45.51, 45.42 and 45.42 respectively for TransE [1], TransH [2] and TransD [4]. Hits@3 25 are 37.24, 37.68 and 37.38 respectively for TransE [1], TransH [2] and TransD [4]. We can make an inference that TransE [1], TransH [2], and TransD [4] are better for top-3 and top-10 predictions.

From the table 5.5 we can infer that for the WN18RR [12] dataset, ComplEx [8] is the best embedding technique among the matrix-based embeddings techniques Distmult [6], ComplEx [8], and SimplE [10]. MRR 24 is best for ComplEx [8] which is 0.39. MR 23 is best for the Distmult [6] which is 4012.12. Hits@10 25, Hits@3, Hits@1 are 47.52, 42.17, and 33.77 respectively for the ComplEx [8] are the best among the Distmult [6], ComplEx [8], and SimplE [10].

From the table 5.6 we can infer that for the FB15k-237 [18] dataset, TransH [2] is the best embedding technique among the translational-based embeddings techniques TransE [1], TransH [2], TransR [3], and TransD [4]. MRR 24 is 0.41 for TransE [1]. MR 23 is best for the TransD [4] which is 999.11. Hits@10 25, Hits@3 are 65.07 and 49.92 respectively for the TransH [2]. Hits@1 25 is best for TransE [1] which is 29.45. We can make an inference that TransH [2] is best for top-3 and top-10 predictions. TransE [1] is better for top-1 predictions.

From the table 5.7 we can infer that for the YAGO3-10 [19] dataset, ComplEx [8] is the best embedding technique among the matrix-based embeddings techniques Distmult [6], ComplEx [8], and SimplE [10]. MRR 24 is best for ComplEx [8] which is 0.13. MR 23 is best for the Distmult [6] which is 2509.89. Hits@10 25, Hits@3 is 34.06 and 14.73 respectively for the ComplEx [8] are the best among the Distmult [6], ComplEx [8], and SimplE [10].

S.No	Dataset	Embedding Technique	Embedding Dimension	Epochs	MRR	MR	Hits @ (%)		
							10	3	1
1	YAGO3-10 [19]	TransE [1]	50	2000	0.12	1084.60	21.86	12.12	7.13
2			100		0.18	1044.12	33.03	20.19	9.59
3			200		0.35	1703.94	56.87	41.28	23.72
4			300		**0.41**	1729.19	61.93	46.78	**29.45**
5		TransH [2]	300		0.35	1996.88	**65.07**	**49.92**	16.04
6		TransR [3]	300		0.10	2102.40	18.96	10.72	5.28
7		TransD [4]	300		0.18	**999.11**	32.93	20.15	10.03

Table 5.6: **Comparison of the mean reciprocal rank(MRR), mean rank(MR), hits@1, hits@3, and hits@10 using the translational embeddings TransE [1], TransH [2], TransR [3], TransD [4] performed the experiments using embedding dimensions {50, 100, 200, 300}, and epochs of 2000 for the entity-only model. The experiments are performed on the YAGO3-10 [19] dataset.**

S.No	Dataset	Embedding Technique	Embedding Dimension	Epochs	MRR	MR	Hits @ (%)		
							10	3	1
1	YAGO3-10 [19]	Distmult [6]	50	2000	0.05	3363.60	9.28	4.27	2.78
2			100		0.06	3066.05	11.55	4.87	2.82
3			200		0.07	2686.56	15.94	6.60	3.05
4			300		0.08	**2509.89**	19.30	7.86	3.06
5		ComplEx [8]	50		0.06	2932.12	12.99	6.13	2.86
6			100		0.08	2610.49	19.08	8.38	2.95
7			200		0.11	2576.85	27.13	10.76	3.12
8			300		**0.13**	2809.63	**34.06**	**14.73**	3.02
9		SimplE [10]	300		0.09	2929.83	21.24	8.50	**3.15**

Table 5.7: **Comparison of the mean reciprocal rank(MRR), mean rank(MR), hits@1, hits@3, and hits@10 using the matrix factorization-based embeddings Distmult [6], ComplEx [8], and SimplE [10] performed the experiments using embedding dimensions 50, 100, 200, 300, and epochs of 2000 for the entity-only model. The experiments are performed on the YAGO3-10 [19] dataset.**

From the table 5.8 we can infer that for the ConvE [12] embedding technique on FB15k-237 [18], we get the best Hits@10 25, Hits@5, Hist@3 30.20, 21.78, and 16.46 respectively when trained using 100 embedding dimensions and trained using Adagrad [25] optimizer. We can also infer that for the ConvE [12] embedding technique on WN18RR [12], we get the best Hits@10 25, Hits@5, Hist@3 38.61, 33.92, and 28.73 respectively when trained using 200 embedding dimensions and trained using Adam [20] optimizer and trained for 2000 epochs. From the table 5.9 we can infer that R-GCN [11] performs better for FB15k-237 [18] and WN18RR [12] datasets.

From the table 5.10 we can infer that for the FB15k-237 [18] dataset using entity and image data, TransE [1] gives better results when compared to TransH [2]. For TransE [1], Hits@10 25, Hits@5, Hist@3 are 24.55, 17.86, and 14.31 respectively. TransE [1] and TransH [2] both have Hits@1 as 8. We can also infer that for the YAGO3-10 [19] dataset using entity and image data, TransH [2] gives better results when compared to TransE [1]. For TransH [2], Hits@10 25, Hits@5, Hist@3, Hits@1 are 39.81, 32.31, 26.98, and 17.56 respectively.

S.No	Dataset	Embedding Technique	Embedding Dimension	Epochs	MR	Hits @ (%)			
						10	5	3	1
1	WN18RR [12]	ConvE [12]	100	100	11650.91	0.48	0.27	0.12	0.07
2				500	8270.34	12.07	8.89	6.29	1.54
3				1000	7423.65	24.66	19.84	16.04	**5.20**
4			200	500	8774.75	22.71	18.50	14.50	2.15
5				1000	7109.11	33.92	29.00	24.04	3.45
6				2000	**6155.72**	**38.61**	**33.92**	**28.73**	4.20
7	FB15k-237 [18]		100	100	762.82	8.09	5.54	3.50	1.46
8				100*	562.53	**30.20**	**21.78**	**16.46**	8.37
9			200	100*	370.09	29.37	20.65	16.25	**8.66**
10				500*	**362.37**	18.94	12.06	8.86	4.63
11	YAGO3-10 [19]		100	100	2873.56	1.4	0.9	0.64	0.18
12				1000	3345.63	4.98	4.29	3.85	2.5
13			200	100	2213.10	2.85	1.57	0.90	0.32
14				1000	5131.43	0.78	0.48	0.38	0.08

Table 5.8: **Comparison of the mean rank(MR), hits@1, hits@3, hits@5, and hits@10 using the ConvE [12] embedding technique and performed the experiments using embedding dimensions {100, 200}, and epochs {100, 500, 1000, 2000} for entity only model. The experiments are performed on the WN18RR [12] dataset, FB15k-237 [18] dataset and YAGO3-10 [19] dataset using Adam optimizer [20].**

* indicates that the results of that row are trained on Adagrad [25] optimizer.

S.No	Dataset	Embedding Technique	Embedding Dimension	Epochs	MRR	Hits @ (%)		
						10	3	1
1	FB15k-237 [18]	R-GCN [11]	100	6000	0.24	41.85	26.07	15.29
2	WN18RR [12]				0.38	43.97	40.03	34.56
3	YAGO3-10 [19]				0.09	16.04	8.57	4.99

Table 5.9: **Comparison of the mean reciprocal rate(MRR), hits@1, hits@3, and hits@10 using the R-GCN [11] embedding techniques performed the experiments using embedding dimensions of 100 and epochs 6000 for the entity only model. The experiments are performed on the FB15k-237 [18] dataset, WN18RR [12] dataset and YAGO3-10 [19] dataset.**

S.No	Dataset	Embedding Technique	Embedding Dimension	Epochs	Hits @ (%)			
					10	5	3	1
1	YAGO3-10 [19]	TransE [1]	50	100	22.34	16.63	13.50	7.95
2			350	100	24.19	17.36	14.13	7.57
3				207	**24.55**	**17.86**	**14.31**	7.79
5		TransH [2]	300	100	24.03	17.36	13.74	**8.02**
6	FB15k-237 [18]	TransE [1]	50	100	38.59	30.65	25.65	17.12
7			350	820	38.23	30.55	25.80	17.45
8		TransH [2]	200	100	**39.81**	**32.31**	**26.98**	**17.56**
9			350	100	39.04	31.51	26.36	17.51

Table 5.10: **Comparison of the mean rank(MR), hits@1, hits@3, hits@5, and hits@10 using the TransE [1] and TransH [2] embedding techniques performed the experiments using embedding dimensions {50, 200, 300, 350} and epochs of {100, 207, 820} for the entity and image model. The experiments are performed on the YAGO3-10 [19] dataset and FB15k-237 [18] dataset.**

6 Conclusion

In this work, I did a detailed survey on knowledge graph embedding techniques and implemented four translational embedding techniques TransE [1], TransH [2], TransR [3], and TransD [4], three matrix factorization-based embedding techniques Distmult [6], ComplEx [8], and SimplE [10], two neural network-based embedding techniques ConvE [12], and R-GCN [11] for the entity only data. For the entity and image data, I implemented TransE [1] and TransH [2] embedding techniques.

References

[1] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[2] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.

[3] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[4] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "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)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 687–696. [Online]. Available: https://aclanthology.org/P15-1067

[5] H. Xiao, M. Huang, and X. Zhu, "TransG : A generative model for knowledge graph embedding," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 2316–2325. [Online]. Available: https://aclanthology.org/P16-1219

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

[7] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Icml*, 2011.

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

[9] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[10] S. M. Kazemi and D. Poole, "Simple embedding for link prediction in knowledge graphs," *Advances in neural information processing systems*, vol. 31, 2018.

[11] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

[12] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[13] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou, "End-to-end structure-aware convolutional networks for knowledge base completion," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3060–3067.

[14] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," *arXiv preprint arXiv:1712.02121*, 2017.

[15] D. Q. Nguyen, T. Vu, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A capsule network-based embedding model for knowledge graph completion and search personalization," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 2180–2189. [Online]. Available: https://aclanthology.org/N19-1226

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[18] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1499–1509.

[19] F. Mahdisoltani, J. Biega, and F. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *7th biennial conference on innovative data systems research*. CIDR Conference, 2014.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] Y. Dai, S. Wang, N. N. Xiong, and W. Guo, "A survey on knowledge graph embedding: Approaches, applications and benchmarks," *Electronics*, vol. 9, no. 5, p. 750, May 2020. [Online]. Available: http://dx.doi.org/10.3390/electronics9050750

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[23] P. Pezeshkpour, L. Chen, and S. Singh, "Embedding multimodal relational data for knowledge base completion," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3208–3218. [Online]. Available: https://aclanthology.org/D18-1359

[24] D. Oñoro-Rubio, M. Niepert, A. García-Durán, R. Gonzalez-Sanchez, and R. J. López-Sastre, "Answering visual-relational queries in web-extracted knowledge graphs," in *AKBC*, 2019.

[25] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.