A Review in Knowledge Extraction from Knowledge Bases

Fabio Yáñez-Romero
University Institute for Computer Research
University of Alicante
fabio.yanez@ua.es

Andres Montoyo, Rafael Muñoz, Yoan Gutiérrez, Armando Suárez
Department of Computing and Information Systems
University of Alicante
montoyo@dlsi.ua.es, rafael@dlsi.ua.es, ygutierrez@dlsi.ua.es, armando@dlsi.ua.es

Abstract

Generative language models achieve the state of the art in many tasks within natural language processing (NLP). Although these models correctly capture syntactic information, they fail to interpret knowledge (semantics). Moreover, the lack of interpretability of these models promotes the use of other technologies as a replacement or complement to generative language models. This is the case with research focused on incorporating knowledge by resorting to knowledge bases mainly in the form of graphs. The generation of large knowledge graphs is carried out with unsupervised or semi-supervised techniques, which promotes the validation of this knowledge with the same type of techniques due to the size of the generated databases. In this review, we will explain the different techniques used to test and infer knowledge from graph structures with machine learning algorithms. The motivation of validating and inferring knowledge is to use correct knowledge in subsequent tasks with improved embeddings.

1 Introduction

Knowledge bases (KB) are widely used for storage information used in different machine learning tasks. Knowledge bases are generally represented by knowledge graphs (KG), which store information that employ nodes (entities) and edges (relations) creating a network. This way of storing knowledge has been popularized in recent years due to it being a more expressive, versatile and scalable than traditional databases (Hogan et al., 2021).

The efficient use of knowledge stored in a KG with machine learning models is not a trivial task. Traditional machine learning models, including deep neural networks use vectors as input, while the structure of KGs is more complex and can't be simplified in a vector, due to the need for representing nodes, edges, connectivity, global relations inside the graph and features of every element (Sanchez-Lengeling et al., 2021).

Methodologies used for extracting knowledge from KGs focus on creating latent vectors with the graph information (embeddings) or using neural networks specially designed for dealing with the graph structure.

Many KBs are developed using non supervised machine learning techniques, generating massive data in the process. Those methods may cause errors when completing the KB due to false relations between nodes. Large KBs also have problems with not useful information introduced for a specific task which can be considered as noise.

2 Knowledge Graph Structure

KGs are made up of two sets of elements $G = V, E$. Where V is the set of nodes (entities) and E is the set of edges (relations). Where:

$$|V| = N, |E| = R$$

being *N* and *R* the number of entities and relations, respectively.

A knowledge graph can be classified as homogeneous or heterogeneous if nodes are of the same class or different classes, cyclic or acyclic if its possible to reach the initial node traveling between edges or no and directed or undirected if nodes are connected in one direction only or both, respectively.

There are variants from conventional graph, this is the case of hypergraphs which contains hyperedges linking more than two nodes or multigraphs which allow more than one edge between two nodes.

Graph Knowledge can be represented in many ways due to the versatility of the graph structure. A

Proceedings of Recent Advances in Natural Language Processing, pages 109–116
Varna, Sep 4–6, 2023
https://doi.org/10.26615/978-954-452-092-2_012

representation of nodes and edges connecting those nodes is necessary. More complex graphs may consider many features inside each node, nodes containing subnodes with their own relations, features for each edge, different types of edges and global features associated with the entire graph (Sanchez-Lengeling et al., 2021).

For representing the information contained in a KB, consider a head entity e_h and a tail entity e_t sharing a relation r. This is represented by a triple (e_h, r, e_t). If the entities are directly connected with the relation we consider this a "1-hop" relation, otherwise it is called a "multi-hop" relation. Multi-hop relations are more difficult to detect, mainly for entities with a distance of 3 hops or more. There are many tasks involving constructed KGs in NLP, most of the current research focus on entity linking, question answering (QA) and Fact Checking. The structure of triples is generally utilized for representing graph information in the lowest level. This is the general structure in query languages developed for graph databases as Cypher for databases like Neo4j and sparql for data in the Resource Description Framework (RDF) format.

3 Machine Learning Techniques

ML techniques used on KBs represent the information contained in nodes and edges in a structured format as embeddings, other models act directly over the graph structure, this is the case of Graph Neural Networks (GNNs).

We have considered three different families of models for Entity Linking according to the techniques used to perform the task. There are translational models, which consider that the different relationships between elements can be represented as displacements in space, matrix factorization models represent the relationships between entities as tensors and perform decomposition operations on the tensors to represent each entity and relationship and finally deep neural models are used to obtain the main characteristics of each possible relationship and determine whether they are truthful or encode information from nearby entities.

3.1 Translational models

3.1.1 Euclidean Space Models

Translational models express the existing relation between two entities as a translation in a vector space. Head entity *h* and tail entity *t* have a relation *r* which can translate the first entity to the second, this is the case for the first translational model, TransE (Bordes et al., 2013):

$$h + r \approx t \tag{1}$$

The loss function for creating embeddings with TransE is based on:

$$|h + r - t| \approx 0 \tag{2}$$

TransE does not deal well with complex relations, i.e relations one-to-many (1-N), many-to-one (N-1) or many-to-many (N-N). TransH improves the representation of complex relations creating a unique hyperplane for each relation between two entities (Wang et al., 2014). In this case, the relation is a vector of the hyperplane and entity vectors are translated to the hyperplane by a multiplication with a specific relation matrix(W_r).

TransR considers both entities and relations should be in different spaces. This allow different entity representations according to the relation between them (Lin et al., 2015). In this case, entities *h* and *t* from each triple are proyected in the relation space multiplying with the matrix M_r getting h_r and t_r.

TransD uses less parameters than its predecessor, this can be done using vector multiplications instead of matrices. TransD assumes two vectors for each entity and relation: the first vector (*h, r, t*) represents the meaning of the entity or relation and the second (h_p, r_p, t_p) indicates how the entity must be proyected in the relation space, is utilized to map entities in the relation space (Ji et al., 2015). For each triple there are two matrices M_{rh} (relation-head) and M_{rt} (relation-tail) for proyecting entities in the relation space.

TransD uses the same number of parameters for each specific relation. This may lead to overfitting when using more parameters than necessary (simple relations) or underfitting when there are less parameters (complex relations). In TranSparse(separate) each relation uses a sparse matrix for each entity, with different sparse degrees. This enable the use of more or less parameters depending of the complexity of the relation (Ji et al., 2015).

TransE regularization forces entity embeddings to stay inside a spherical vector space out of the range of the correct triple. The regularization used in TransE is normalization, making the magnitude of each embedding become 1 during each step of

learning. This provoke a violation of equation (2), making the sum of head entity and relation not equal to tail entity. This causes major problems, warping the embeddings obtained. To solve this TorusE creates entity and relation embeddings using the same principles as TransE but in a torus space (Ebisu and Ichise, 2017).

PairRE employs paired vectors for representing complex relations. These vectors proyect entities in the euclidean space where distance is minimized if the relation is right. The main advantage of PairRE is that both paired vectors allow more versatility in the loss function, achieving a better representation of complex relations (Chao et al., 2020).

3.1.2 Complex Space Models

Even if Euclidean space models progressively improve state of the art, they still have difficulties dealing with relations of symmetry, anti-symmetry, inversion and composition. RotatE tries to solve this problem with a complex space in order to represent embeddings using Euler's identity. This way the translation from the head entity to the tail entity is a rotation (Sun et al., 2019). RotatE also changes the loss function introducing self adversarial samples, which improves the training process. The score function employed in RotatE is the the same as equation (2), but using Hadamard product instead of vector sum between head entity and relation.

RotatE is improved with more dimension spaces through relation modeling with orthogonal transformations embeddings OTE (Tang et al., 2019). OTE makes orthogonal transformations with the head and relation vectors to the tail vector, and then from the tail and relation vectors to the head vector.

Extending the idea of complex spaces, QuatE uses an hypercomplex space with 3 imaginary components i, j, k with the objective of having more degrees of freedom to the obtained embeddings. In this case, the scoring function utilized rotates head entity using the Hamilton product (Zhang et al., 2019).

Previous models interpret relations using only translations or rotations inside a geometric space, but not both types of movements. Whereas translationals models are not capable of represent fundamental aspects of relations as symmetry, inversion or composition, rotational models fail to deal with hierarchical relations or multiples relations between two entities. However, DualE deals with these problems using dual cuaternions (Cao et al., 2021). Dual cuaternions are built with the sum of two cuaternions ($Q = a + \epsilon b$) where a and b represent the two cuaternions. Using dual cuaternions it is possible to model embeddings with translation and rotation relations.

3.1.3 Other Non-Euclidean Space Models

Other models explore the posibility of using mathematical expresions out of the euclidean space.

ManifoldE is a model that uses non-euclidean space. It considers that translational models are algebraically ill-conceived because they generate more equations than variables to solve, leading to approximate calculations for tasks like entity linking, where there are many entity candidates for one relation. In the case of ManifoldE, it uses a principle based on a "manifold" function for expressing the relation between two entities (Xiao et al., 2015). With this approach calculation should be exact, retrieving true candidates for each relation. ManifoldE expands the position of golden triples from one point (TransE) to a manifold using a larger dimension sphere, diminishing noise when detecting true relations between all candidates and improving embedding vectors precision. Considering a head entity and a relation, all possible tail entities are inside a manifold of greater dimension (sphere). Scoring function is obtained as the difference in distance between radius of the sphere and equation (2). ManifoldE improves their results using a hyperplane as a manifold instead of a sphere.

Hyperbolic space is ideal for modeling entities with hierarchical information due to its curvature. The problem with hyperbolic space is representing entities with different hierarchies under different relations. MuRP utilizes a *Poincaré Ball* as a hyperbolic space, creating multi-relational embeddings for each entity and relation (Balažević et al., 2019). The key of MuRP is using a hypersphere in hyperbolic space because it grows exponentially compare to euclidean space, having more space to separate each node. MuRP trains relation-specific parameters used for transforming entity embeddings through Möbius matrix-vector multiplication (in order to obtain the hyperbolic entity embeddings) and Möbius addition. The hyperbolic entity embeddings are obtained by Möbius matrix-vector multiplication projecting the original embeddings to the tangent space of the Poincaré ball transformed by the diagonal relation matrix and then projected back to Poincaré ball.

MuRP cannot encode some logical properties of relationships. It uses a fixed curvature for each relation.Although specific curvature for each relation would represent better hierarchies based on the context, it also uses only translations in the hyperbolic space. By contrast, ATTH creates embeddings in hyperbolic space using reflexions and rotations, enabling RotatE patterns to be captured, as well as considering a relation-specific curvature c_r that allows a variety of hierarchies (Chami et al., 2020). Rotations are created with Givens transformations matrices due to this model does not employ complex numbers. ATTH use entity biases in the scoring function which act as margins for triples.

Previous methods are designed for creating entity and relation representations in Euclidean, Hyperbolic or Hyperspherical space, but no one of them compare results in different spaces. Geometry Interaction Knowledge Graph Embeddings (GIE) (Cao et al., 2022) considers vectors in Euclidean (E), Hyperbolic (H) and Hyperspherical (S) spaces for head and tail entities and uses an attention mechanism over each vector in order to prioritize the space which represents better knowledge from the entity. Vectors in Hyperbolic and Hyperspherical space are logarithmically mapped to tangent space before applying attention and then features are extracted. GIE has an attention vector with a specific component for each different space both for head and tail entities inside a triple.

3.2 Tensor factorization models

Using tensors for expressing entities and relations has some advantages over translational models:

· Tensors can represent multiple relations of any order, you just need to increase tensor dimensionality.

· Previous knowledge from the problem structure is not necessary in order to infer knowledge from data.

3.2.1 Euclidean Space Models

RESCAL is the first tensor factorization model created to represent relations between entities. In this model, each matrix is constructed representing the relation between two entities, like a confusion matrix, and each matrix indicates a specific relation. The data is given as a $(n \cdot n \cdot m)$ tensor where *n* is the number of entities and *m* is the number of relations (Nickel et al., 2011). RESCAL employs the following factorization over each slice of tensor X_k:

$$X_k \approx AR_kA^T, \text{for } k = 1, ..., m \quad (3)$$

Where A is a *n* x *r* matrix containing latent-component representation of entities an R_k is an *r* x *r* matrix that models the interactions between latent components for relation *k*.

Matrix R_k is asymmetric, which is useful for considering whether a latent component acts as a subject or object, given that each entity has a unique latent-component representation even if it is a subject or object in a relation. Matrices A and R_k are computed solving the following minimization problem:

$$min\ f(A, R_k) + g(A, R_k) \quad (4)$$

Where:

$$f(A, R_k) = \frac{1}{2}(\sum ||X_k - AR_kA^T||_F^2) \quad (5)$$

and *g* is a regularization term included to avoid overfitting:

$$g(A, R_k) = \frac{1}{2}\lambda(||A||_F^2 + \sum ||R_k||_F^2) \quad (6)$$

In order to reduce training parameters in RESCAL, DistMult uses a diagonal matrix W_r instead of an asymmetric relation matrix (Yang et al., 2014). This leads to a more expressive model than transE with the same number of parameters, being as scalable as previously mentioned models but less expressive than RESCAL.

Holographic embeddings use vector circular correlation to represent entity embeddings. HolE creates holographic embeddings for represent pairs of entities(Nickel et al., 2015). Correlation makes HolE efficient to compute and scalable to large datasets. This operation can be considered as a compression of the tensor product, in circular correlation each component is a sum of a fixed partition of pairwise interactions. HolE can store and retrieve information via circular convolution and circular correlation, respectively and it also learns the embeddings of the data.

SimplE is a tensor factorization method based con *Canonical Polyadic*(CP) decomposition (Kazemi and Poole, 2018). It uses two vectors for each entity (h_e, t_e) and relation $(v_r, v_{r^{-1}})$. SimplE uses a similarity function for each triple which is

the average of the CP scores for the current triple and its inverse relation triple.

TuckER is a lineal model for tensor factorization which generalizes previous tensor factorization models like RESCAL, DistMult, ComplEx and SimplE based on *Tucker decomposition*. It makes a decomposition from the binary tensor of triplets. It factorizes a tensor into a core smaller tensor multiplying one matrix for each dimension in the original tensor (Balazevic et al., 2019). In the case of TuckER, the decomposition creates a smaller tensor *W*, and matrices e_h, w_r and e_t for head entity, relation and tail entity, respectively.

3.2.2 Other Non-Euclidean Space Models

As RotatE, ComplEx uses imaginary numbers in the complex space, in this case it performs tensor factorization using Hermitian dot product, which involves the conjugate-transform on one of the two vectors multiplied. With this type of dot product, we obtain a non symmetric matrix being able to represent antisymmetric relations while maintaining linearity and low time complexity (Trouillon et al., 2016).

3.3 Deep Neural Models

Graph neural networks can encode information about neighbours from each specific node, introducing context during processing in the neural network.

3.3.1 Graph Convolutional Networks (GCNs)

The first GCN introduced generates hidden states for each node processed taking into consideration each neighbour and relation. For each GCN layer, the processed node adds information from each neighbour equally as shown in the next equation:

$$h_i^{(l+1)} = \sigma\left(\sum_{m \in M_i} g_m(h_i^{(l)}, h_j^{(l)})\right) \quad (7)$$

Where h_i^l is the hidden state of node *i* for the layer *l* in the GCN. M_i is the set of neighbours considered for node *i*, g_m is a linear transformation which uses a weight matrix *W* and σ is an element-wise activation function.

The context given by graphs improves many tasks when dealing with relational data, this is the case for R-GCN, an encoder that produces a hidden state for each node considering neighbours but also specific relations (Schlichtkrull et al., 2017), in contrast with original GCN, being suitable for processing heterogeneous graphs.

3.3.2 Graph Attention Networks (GATs)

GCNs make convolutions considering equal importance among all edges in the processed graph, which may be a shallow approach for tasks where specific nodes and edges have more important information than others (Kipf and Welling, 2017). In order to solve this issue, Graph Attention Networks are introduced. GATs make a convolution considering different weights for each edge connected to a specific node and can have multiple weights associated for each edge equal to the number of attention heads (Veličković et al., 2018).

A2N uses attention mechanism with specific queries in order to generate conditioned embeddings taking into account each query with the neighborhood of a source entity (Bansal et al., 2019). A scalar attention score is generated for each neighbour and then their embeddings are aggregated generating a new source embedding $\hat{s}$. Lastly, concatenate the new source embedding with the initial and projecting it to obtain the final source embedding, **s** . In the original paper, DistMult is utilized as an attention scoring function as it allows the projection of neighbors in the same space as target entities.

The use of non-Euclidean spaces has been extended to graph neural networks as in the case of M2GNN (Wang et al., 2021). Previous models using non-Euclidean spaces only considered homogeneous relations, so they lack expressiveness in this respect. M2GNN creates a non-constant heterogeneous curvature space using new parameters in the network called curvature coefficients. The proposed architecture also makes use of attention heads to improve the accuracy obtained.

3.4 Convolutional Neural Networks (CNNs)

CNNs utilized broadly in computer vision have recently been used for entity linking. The main reason is that CNNs can solve entity linking tasks with far less parameters than previous mentioned models like DistMult. CNNs are also considered a very expressive way of representing entities and relations comparing to translational models, due to the number of features extracted with the CNN filters (Kipf and Welling, 2017).

ConvE is the first convolutional model achieving good results with entity linking tasks. It is simple, as it uses only one convolutional layer with 2D

convolutions, a proyection layer to the embedding dimension and an inner product to make the entity linking prediction (Dettmers et al., 2017). The convolution is made by first concatenating the 2D vectors from the head entity and relation embeddings. Score function used for training the model is the following:

$$\phi(\mathbf{e_h}, \mathbf{e_t}) = f(vec(f(\overline{e_h}; \overline{r_r}) * w))\mathbf{W})\mathbf{e_t} \quad (8)$$

Where $\overline{e_h}$ and $\overline{r_r}$ are the 2D representations of embeddings $\mathbf{e_h}$ and $\mathbf{r_r}$ respectively. w are filters used in the convolutional layer, $\mathbf{W}$ is the matrix for projecting the data into another dimensional space for matching $\mathbf{e_o}$.

ConvKB uses a convolutional layer with 3-column matrices, where each matrix is made of the concatenation of the triple vectors (e_h, r, e_t). The features obtained after convolution are concatenated and score is obtained performing a multiplication with a weight vector $\mathbf{w}$ (Nguyen et al., 2018).

Filters used for convolution in previous models are designed arbitrarily, which can lead to a poor performance. In order to solve this problem, HypER uses a hypernetwork for determining the right filter for each relation (ević et al., 2019). A fully connected layer is used for obtaining embeddings representing head entity and relation, then the hypernetwork creates the filters of each relation embedding which will be utilized during convolution of entity embeddings. The hypernetwork proposed is a single fully connected layer. HypER uses a weight matrix that projects the results to another dimensional space in order to make the dot product between head entity and tail entity.

4 Discussion and future directions

The representation of knowledge bases as embedding vectors can be seen as a way to obtain contextualised embeddings of any knowledge base with a graph structure, such as ontologies. Furthermore, contextualized embeddings can be used beyond tasks such as entity linking or knowledge base completion by representing the latent knowledge of the knowledge bases used in the form of vectors.

Contextualised embedding vectors commonly used in natural language processing (NLP) are usually obtained from a corpus with unsupervised techniques such as Word2Vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014), using Long-Short Term Memory neural networks (LSTMs) on the text as in the case of ELMo (Peters et al., 2018) or using language models with Transformers-type architecture as BERT (Devlin et al., 2019).

With the methods explained in this paper, contextual vectors can be created taking into account as context only the graph itself and the relations existing in it without taking into consideration any corpus. This becomes even more important taking into account the current research trends within NLP focused on combining knowledge from ontologies with the latent language of language models, creating synergies with the aim of improving the state of the art in different NLP tasks, achieving explainable models or reaching competitive results with lighter models (Pan et al., 2023). It is expected that the improvement of embeddings obtained from knowledge graphs will be useful to achieve a better integration between language models and knowledge graphs.

5 Conclusions

Both in the case of translational models and in tensor factorization, there is a tendency to represent increasingly complex spaces, to the point of combining different types of spaces into one (euclidean, hyperspherical and hyperbolic) or to represent increasingly complex vector spaces (complex space, quaternions, etc.). However, in some cases it is observed that the state of the art is surpassed without necessarily increasing the complexity of the space represented; this is the case of SimplE (which achieves results similar to ComplEx) or Tucker.

Alternative spaces to the euclidean with positive or negative curvature tend to better represent some properties of entities with a smaller number of features, such as circular relations in hyperspherical spaces and hierarchies in hyperbolic spaces, allowing the creation of embeddings at a lower computational cost.

In the case of deep neural models, tests have also been carried out with positive and negative curvature spaces. In these cases, curvature is a parameter to be trained within the network.

The current state of the art is led by models that combine different vector spaces (GIE, M2GNN).

References

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, Hong Kong, China. Association for Computational Linguistics.

Ivana Balažević, Carl Allen, and Timothy Hospedales. 2019. Multi-relational poincaré graph embeddings.

Trapit Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. 2019. A2N: Attending to neighbors for knowledge graph inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4387–4392, Florence, Italy. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2021. Dual quaternion knowledge graph embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6894–6902.

Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2022. Geometry interaction knowledge graph embeddings.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings.

Linlin Chao, Jianshan He, Taifeng Wang, and Wei Chu. 2020. Pairre: Knowledge graph embeddings via paired relation vectors.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Takuma Ebisu and Ryutaro Ichise. 2017. Toruse: Knowledge graph embedding on a lie group.

Ivana Balaž ević, Carl Allen, and Timothy M. Hospedales. 2019. Hypernetwork knowledge graph embeddings. In *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*, pages 553–565. Springer International Publishing.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge graphs. *ACM Comput. Surv.*, 54(4).

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. 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)*, pages 687–696, Beijing, China. Association for Computational Linguistics.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2181–2187. AAAI Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 327–333, New Orleans, Louisiana. Association for Computational Linguistics.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2015. Holographic embeddings of knowledge graphs.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 809–816, Madison, WI, USA. Omnipress.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B. Wiltschko. 2021. A gentle introduction to graph neural networks. *Distill*. Https://distill.pub/2021/gnn-intro.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *CoRR*, abs/1902.10197.

Yun Tang, Jing Huang, Guangtao Wang, Xiaodong He, and Bowen Zhou. 2019. Orthogonal relation transforms with graph context modeling for knowledge graph embedding.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks.

Shen Wang, Xiaokai Wei, Cicero Nogueira Nogueira dos Santos, Zhiguo Wang, Ramesh Nallapati, Andrew Arnold, Bing Xiang, Philip S. Yu, and Isabel F. Cruz. 2021. Mixed-curvature multi-relational graph neural network for knowledge graph completion. In *Proceedings of the Web Conference 2021*, WWW '21, page 1761–1771, New York, NY, USA. Association for Computing Machinery.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 1112–1119. AAAI Press.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2015. From one point to a manifold: Knowledge graph embedding for precise link prediction.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings.