International Journal on Semantic Web and Information Systems
Volume 20 • Issue 1

Improvement of Web Semantic and Transformer-Based Knowledge Graph Completion in Low-Dimensional Spaces

Xiai Yan, Xiangtan University, China & Hunan Police Academy, China*
Yao Yi, Xiangtan University, China
Weiqi Shi, Hunan Police Academy, China
Hua Tian, Hunan Police Academy, China
Xin Su, Hunan Police Academy, China

ABSTRACT

In recent years, knowledge graph completion (KGC) has garnered significant attention. However, noise in the graph poses numerous challenges to the completion of tasks, including error propagation, missing information, and misleading relations. Many existing KGC methods utilize the multi-head self-attention mechanism (MHA) in transformers, which yields favorable results in low-dimensional space. Nevertheless, employing MHA introduces the risk of overfitting due to a large number of additional parameters, and the choice of model loss function is not comprehensive enough to capture the semantic discriminatory nature between entities and relationships and the treatment of RDF indicates that the dataset contains only positive (training) examples, and the error facts are not encoded, which tends to cause overgeneralization.

KEYWORDS

knowledge graph completion, low-dimensional spaces, transformer

INTRODUCTION

The landmark development of the Semantic Web can be traced back to the idea put forward by Berners-Lee et al. that it is desirable to augment the original World Wide Web by treating the Semantic Web as machine-understandable information (2001). In this view, machine-understandable information is accomplished by assigning certain data expressiveness to metadata, which is usually an ontological form of data, has logical semantics, and is amenable to inference. The basic model of the Semantic Web is resource description framework (RDF), a standardized format for representing and exchanging data for describing relationships and properties between resources (d'Amato, 2020).

Knowledge graphs (KGs) consist of structured collections of relationships represented as triples (head entity, relationship, tail entity). They are widely used in various projects, such as intelligent

DOI: 10.4018/IJSWIS.336919 *Corresponding Author

quizzes, comprehensive searches, and recommender systems that rely on substantial data support. Considering the historical issues of semantic networks, especially ontologies, knowledge graphs can be viewed as a richer and more complex semantic network (e.g., RDF), which can contain multiple types of entities and relationships and provide richer semantic information (Breit et al., 2023). Despite containing millions of factual statements, KGs still require a considerable amount of work. For instance, a significant portion of individuals in the Freebase Knowledge Graph, around 71%, lack information regarding their place of birth, while 75% lack information about their nationality (Dong et al., 2014). This highlights the incompleteness of knowledge coverage within KGs. To address this issue, the task of knowledge graph completion (KGC) was introduced.

Knowledge graph completion refers to the task of predicting the presence of connections or edges between two nodes in graph-structured data. In the context of graph theory and network analysis, a graph comprises a collection of nodes and the edges that connect them. The objective of KGC is to infer the existence of unknown edges by leveraging the available partial graph information.

The concept of low-dimensional embedding involves mapping high-dimensional representations of entities and relations into a lower-dimensional vector space. This mapping enables the distances and relationships between entities and relations in the vector space to reflect their semantic similarities and associations within the knowledge graph. Numerous approaches exist for achieving knowledge graph completion through low-dimensional embedding. The challenge lies in maintaining a low embedding dimension while still achieving satisfactory model performance.

In recent years, notable progress has been made in knowledge graph completion using the enhanced generic encoder of the transformer model (Baghershahi et al., 2022; Liu et al., 2022; X. Zhang et al., 2022). The attention mechanism, particularly self-attention, has played a crucial role in achieving these advancements. Self-attention effectively captures the dependencies among linear projections within the model and maps them to the output, allowing important information to be learned and focused automatically. Despite the impressive results obtained with transformer models, they often face challenges related to high-dimensional embeddings, complexity, and scalability. These issues arise from the stacking of multiple encoder layers and the subsequent increase in the number of coding blocks (Baghershahi et al., 2022).

In order to solve some of the these problems, we introduce an improved transformer-based model TFttOM; due to the inclusion of multi-layer self-attention and feed-forward networks in the encoder, we reduce the transformer encoder block instead of stacking and mixing multiple encoders as a way to reduce the overall number of parameters of the model and the computational complexity, thus improving the model computational efficiency. We introduce a separable self-attention method with linear complexity to reduce the free parameters and thus enhance the computational efficiency. For the binary cross-entropy function, we choose to optimize it to improve the model's recognition ability. The contribution of this paper can be summarized in the following three areas. (1) TFttOM was proposed to reduce the overall parameter count and improve computational efficiency by simplifying the encoding and decoding modules. (2) The introduction of the linear separable self-attention module reduces the time complexity of the model. (3) A new loss function is proposed to reduce the risk of semantic discrimination while improving generalization ability.

RELATED WORK

At present, knowledge graph completion (KGC) has evolved along two main directions: model embedding-based approaches and neural network-based approaches.

Knowledge graph embedding (KGE) methods aim to represent entities and relations in a low-dimensional continuous vector space and calculate relations between these vectors. However, encoding all the information of a large knowledge graph can lead to high-dimensional embeddings in KGE. High dimensionality can cause issues such as overfitting, complexity overload, and difficulties in similarity comparison and distance computation between embedded vectors. Common KGE models,

such as TransE (Bordes et al., 2013), TransR (Lin et al., 2015), TransD (Ji et al., 2015), and Complex (Trouillon et al., 2016), can only encode symmetric and asymmetric relations. However, the relatively new model RotatE (Sun et al., 2019) has shown good results in inverse and combined relations but needs to improve in predicting complex relations. To address this limitation, a new KGE model called PairRE (Chao et al., 2021) was introduced, which can encode complex relations and multiple relations, achieving good results in predicting complex relations. Recently, Y. Li et al. proposed a new model called TranSHER (2022) by combining the ideas of TransE and PairRE. TranSHER improves the modeling of complex relations and introduces a new scoring function, resulting in good performance in the field of knowledge graph completion. Despite the progress made by these methods in prediction tasks, they often suffer from complexity due to the large number of non-embedded free parameters, which grows rapidly with the size of the knowledge graph.

Neural network-based approaches often rely on graph neural networks (GNNs), which operate by iteratively propagating node and neighbor information and combining them using aggregation functions (Wu et al., 2021). Wang et al. proposed a semi-supervised deep model with multi-layer nonlinear functions, improving the representation of graph embeddings by capturing highly nonlinear graph structure information (2016). Hamilton et al. introduced the GraphSAGE algorithm, which utilizes graph convolutional neural networks (GCNs) to learn node representations (2017). This approach achieves scalability and high performance in knowledge graph completion (Hamilton et al., 2017). Sultana et al. proposed a unified learnable and line graph-based graph neural network framework (year). This approach mitigates excessive smoothing and information loss caused by stacking layers and pooling operations, respectively, to optimize the results of knowledge graph completion (Morshed et al., 2023). GNNs have the advantage of being less dimensional compared to KGEs due to their high expressiveness. However, they can still face challenges related to stacking modules on top of each other, leading to complexity and scalability issues, as mentioned earlier.

In recent years, there has been significant research on methods for achieving low-dimensional knowledge graph completion (KGC). Sachan et al. proposed a quantization-based method where each entity in the knowledge graph is compressed and represented as a discretely encoded vector, which forms the embedding layer (Jurafsky et al., 2020). While this approach reduces the embedding dimension, it often leads to slower inference speed and difficulties in model convergence. Balaževic et al. discovered that embedding entities and relationships into hyperbolic space can enhance the hierarchical modeling of knowledge graphs. They introduced the MuRP model, which leverages hyperbolic space for better representation (2019). Chami et al. further improved on this with the AttH model, which utilizes logistic pairs of equidistant modes in hyperbolic space to model knowledge graphs (2020). However, a major drawback of hyperbolic space embeddings is that the embedded data is challenging to use in downstream tasks. Zhu et al. proposed the DualDE model, which employs knowledge distillation to learn from high-dimensional representations and then embeds them into low dimensions (2022). This approach aims to compress the model size and improve inference speed. However, a drawback of this method is that it requires substantial training time to reduce the dimensionality of its features.

The importance of web semantic optimization is to improve the semantic understanding and processing of data, promote knowledge discovery and reasoning, and improve the knowledge graph. Recently, Li et al. proposed a SPARQL-based reasoning framework that optimizes four semantic ontologies to achieve automated safety checking by combining the safety risk data from BIMs and sensors (2022). However, this framework could be more effective in dealing with similar data (X. Li et al., 2022). Geng et al. (2022) used a genetic algorithm to optimize the alpha network of the Rete algorithm and proposed a lightweight reasoning engine, tiny uke. This method reduces the occupation of storage resources, shortens the reasoning time, and proposes a four-tuple knowledge representation method with a probability factor for uncertain knowledge so that the reasoning engine can realize self-adaptation. However, this engine has yet to be tested on more hardware platforms and lacks certain scalability. P. Li et al. (2022) proposed a high-order information dissemination method

based on the knowledge map, which explored the potential information in the knowledge map by mining the relationship between entities. In addition, it also enriched the entity representation by double entity aggregation. However, this method does not use the semantic relationship information in the knowledge map; that is, it does not deal with the lack of information in place. Manaa et al. proposed a hybrid recommender system using linked open data and ant colony optimization, which solves the lack of recommendation diversity and reduces the complexity of similarity computation (2023). However, this method still needs to solve the problem of slow convergence of the ant colony algorithm, thus falling into local optimal solutions (Manaa et al., 2023).

Indeed, the transformer architecture has demonstrated impressive performance in various coding and decoding tasks (Vaswani et al., 2017). Researchers have also explored the application of transformers in the context of knowledge graph completion. X. Chen et al. proposed a hybrid multimodal perceptual transformer framework that utilizes transformer encoding, leveraging the attention mechanism to achieve hybrid multimodal coding (2022). This approach allows for the effective integration of different modalities through attention mechanisms. Peyman et al. fused transformers into knowledge graph embedding by employing low-dimensional representations and multi-head attention to achieve highly expressive vector representations (Baghershahi et al., 2022). While this approach achieved good results, the large number of free parameters in the multi-head attention module can impact model performance. Chen et al. introduced PatReFormer, a patch refinement model based on the transformer architecture (date). PatReFormer divides the embedded data into a series of patches and employs a cross-attention module for the bidirectional embedding of entities and relations. This approach addresses the limitations of linear variation, deep convolution, and inductive bias, thus improving performance (C. Chen et al., 2023). However, it should be noted that the patch articulation in PatReFormer does not involve encoding at relevant positions, which may introduce certain hidden issues. In summary, transformer-based coding in knowledge graph completion has demonstrated excellent capabilities, especially when combined with its attention mechanism for comprehensive processing of entity and relationship information within patches. However, transformers may face computational and memory challenges when dealing with large-scale knowledge graphs.

In the field of KGC, there has been a proliferation of transformer-based approaches, and the performance of these models has been steadily improving over the years. However, the embedding dimensions of these models have also been increasing, often needing a clear understanding of the optimal dimensionality. For example, ConvE (Dettmers et al., 2018) achieves optimal performance in 200 dimensions, while QuatE (S. Zhang et al., 2019) slightly surpasses ConvE but requires 400 dimensions. Rotate (Sun et al., 2019), which serves as a benchmark among these models, reaches an impressive dimensionality of 1000. Given the desire for both low embedding dimensions and good model performance, there is a need to explore approaches that can strike a balance between these two factors. Research efforts can focus on developing novel techniques or modifications to existing models that enable effective knowledge graph completion with lower-dimensional embeddings without sacrificing performance.

LOW-DIMENSIONAL SPACE TRANSFORMER-BASED KNOWLEDGE GRAPH COMPLETION APPROACH

Utilizing the self-attention mechanism of transformers in the context of knowledge graph completion can be a promising approach. The self-attention mechanism allows the model to capture dependencies and interactions between entities and relationships effectively (Likhosherstov et al., 2021). By incorporating the self-attention mechanism into the model, we can leverage its ability to capture fine-grained interactions and dependencies between the entities and relationships in the knowledge graph. It can improve the model's ability to make accurate predictions and achieve higher performance in knowledge graph completion tasks.

Background and Problem Description

Knowledge graphs can be viewed as factual triples collections $G = \left\{\left(h_i, r_j, t_k\right)\right\} \in \left(\mu \times R \times \mu\right)$ where μ and R denote the set of entities (nodes) and relationships (edges) respectively. h_i and r_j represent the ith entity and the jth relation, and the types of relations are large in either KG. The main task of knowledge graph completion is to replace the true triples in the knowledge graph by replacing, for example $\left(h_i, r_j, t_k\right) \rightarrow \left(h_i, r_j, t_b\right)$, replacing the tail entity t_k with t_b, and then verifying that the new ternary $\left(h_i, r_j, t_b\right)$ continues to be true, and the replacement includes but is not limited to tail entities.

Transformer and Knowledge Graph Completion Generic Models

The generic steps for combining transformer with knowledge graph completion work are as follows: (1) First, the μ and R of each entity in the $e \in \mu$ and the relationship $r \in R$ are represented as d_e and d_r dimensional vectors, and in the model d_e and d_r can be set according to different scenarios, generally for better compatibility of the model, and the entities and relations have the same embedding dimension (i.e., $d_e = d_r = d$). (2) Next, the already embedded vectors are fed into the transformer encoder, where the vectors will first be affected by the self-attention module, which is usually chosen differently depending on the dataset at the time of selecting the attention module. The specific flow of the multi-head attention (MHA) mechanism (Vaswani et al., 2017) is shown in Figure 1. The query, keys, and values are generated by linearly transforming the inputs, then they are used to compute the attention weights individually, and finally, the attention weights are multiplied with the values and weighted and summed to get the final multi-head attention representation. The final output under the encoder is an embedding matrix to represent the entities $\widetilde{e}_t$ and relationships $\widetilde{e}_r$, which then goes to the decoder and obtains the scores by scoring each ternary; the choice of the decoder should be as simple and efficient as possible, so basic decoders such as MLP, TwoMult, and Tucker are enough to solve the problem.

Decoder TwoMult and Tucker

The decoder TwoMult remaps the data-decomposed low-dimensional representation back into the original high-dimensional data space by means of a two-way inner product to enable data reconstruction and recovery so that for the decoder's output relational representation $\widetilde{e}_r$ and target entity e_t, we obtain the score by simply calculating the two-way inner product and the specific score calculation is shown in equation (1):

$$\phi\left(h, r, t\right) = \tilde{e}_r^T \times e_t \tag{1}$$

The decoder Tucker decomposition is shown to have extreme expressiveness in the field of knowledge graph completion. For KGs of binary tensor, TuckER (Balažević et al., 2019) uses the Tucker decomposition proposed by Tucker (1966) to compute the score of each triple, where (term) stands for entities and (term) stands for relations. The calculation steps are shown in equation (2):

$$\phi\left(h, r, t\right) = W_c \times_1 e_h \times_2 e_r \times_3 e_t \tag{2}$$

Where $W_c \in \mathbb{R}^{d \times d \times d}$ is the learnable decomposition core tensor, and $\times_n$ denotes the tensor product of n modes. In the original Tucker decomposition, there exists an exponential growth of the number

of free parameters with respect to the size of the embedding. In contrast, in our case, the number of free parameters of the core tensor can be chosen arbitrarily due to the fact that the model is in a low-dimensional state space.

The TwoMult method, noted for its computational efficiency, deftly encodes the source entity's data into the model's relational representation in a nuanced, implicit way. Upon inspecting the output representation of the source entity, a reduction in the mean reciprocal rank (MRR) (refer to the evaluation metrics below) index was observed, signifying that the relational output representation possesses a more substantial informational payload than that of the source entity itself (Baghershahi et al., 2022).The main advantage of Tucker lies in its powerful expressive power. For example, e_h and e_t represent the d-dimensional representations of the subject entity and object entity, respectively. For each subject entity e_h^i, relationship r^j, and object e_t^k, the corresponding elements of i, j, and k are set to 1, and the remaining elements are set to 0. If the triplet e_h, e_r, e_r holds, then W_c is set to 1, and if it does not hold, then it is set to -1. The product of entity embedding and the relationship embedding with the core tensor, after applying logistic sigmoid, accurately represents the original tensor. Those interested in this verification can refer to the original paper.

After the respective scores are obtained in the above manner, they are finally fed into a logistic S-function to calculate the probability of the target entity pair.

ATTENTION MODULES, LOSS FUNCTIONS, AND IMPROVEMENTS IN WEB SEMANTICS

Improvements to the Multi-Attention Module

Multi-Attention Module

The self-attention module used in the general model is called Multi-Head Attention (MHA). Specifically, MHA uses k d-dimensional label embeddings to form the input module X. The input X is then fed into three branches: query Q, key K, and value V. These three branches are composed of h heads (linear layers). The dot product between the output of the linear layers in query Q and key K is calculated, and then the attention matrix is generated by softmax operation. At this point the mathematical expression of the attention matrix is shown in equation (3):

$$H_i = softmax\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) \tag{3}$$

H_i is specifically represented to each header. Finally the output of the attention module is shown in equation (4):

$$MHA\left(Q,K,V\right) = Concat\left(H_1,\ldots,H_i\right)W^O \tag{4}$$

In KGC, since the existing data is generally in the form of triples, for all triples $\left(h,r,t\right)$, the $h \in \mathbb{R}^{d_{model}}$ and $r \in \mathbb{R}^{d_{model}}$, the embedded matrices $\left(h,r\right)$ will be considered as the query $Q \in \mathbb{R}^{d_{model} \times d}$, the key $K \in \mathbb{R}^{d_{model} \times d}$ and the value $V \in \mathbb{R}^{d_{model} \times d}$. In the above equation, the $Q_i = QW_i^Q$ the $K_i = KW_i^K$ and $V_i = VW_i^V$, here $W_i^Q \in \mathbb{R}^{d_{model} \times d}$, $W_i^K \in \mathbb{R}^{d_{model} \times d}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W^O \in \mathbb{R}^{d_{model} \times hd_v}$ and where h is the number of heads. Attention matrix H_i can be understood

specifically as the amount of attention we should give to our entities compared to the relationship. From the attention mechanism, we can understand that it is dependent on the linear projection of the query acts on the input, and continues to pass information in the feed-forward neural network (Likhosherstov et al., 2021), so it plays a very important role in the fusion of entities and relationships.

Separable Self-Attention

Although the H_i matrix is important the model, in MHA, the batch matrix multiplication is used to calculate the attention; such an approach has a big problem in that the computational resources are limited by the capacity of the hardware facilities at the same time its free parameters are too much, so we chose the separable self-attention (MVT2) based on MHA.

The data embedded in MVT2 is divided into query Q, key K, and value V. Here, it is the same as the unimproved MHA, but in query Q first goes into a single linear layer (compared to MHA, we only set up a single linear layer) to map each d-dimensional token in the input message X to a scalar (a dimensionality reduction operation) and use it as weights $\mathrm{W_i} \in \mathbb{R}^{\mathrm{d}}$ to compute the distance between the potential tokens and X, thus generating a k-dimensional vector, which then goes into a softmax operation to generate the context score $\mathrm{C_s} \in \mathbb{R}^{\mathrm{k}}$. Here only the context score of potential token L is computed, so this reduces the time cost of computing attention from $\mathrm{O}\left(\mathrm{K}^2\right)$ to $\mathrm{O}\left(\mathrm{K}\right)$. The critical branch K in the input message X is similarly linearly projected into a single linear d-dimensional space to produce the output $\mathrm{X_K} \in \mathbb{R}^{\mathrm{k\times d}}$. The context vector $\mathrm{C_v} \in \mathbb{R}^{\mathrm{d}}$ is $\mathrm{C_s}$ and $\mathrm{X_K}$ weighted sum of the inputs, at which point its mathematical expression is shown in equation (5):

$$c_v = \sum_{i=1}^{k} c_s\left(i\right) X_K\left(i\right) \tag{5}$$

This way, by $\mathrm{c_v}$, we get all the information in the input X with low computational cost. Using the weights $\mathrm{W_V} \in \mathbb{R}^{\mathrm{d\times d}}$ as the branch V, after linear activation by RELU, we get $\mathrm{X_V} \in \mathbb{R}^{\mathrm{k\times d}}$. Then, the context vector $\mathrm{c_v}$ is propagated by element-by-element multiplication to $\mathrm{x_V}$. Finally, the resulting output is fed to another weighted $\mathrm{W_O} \in \mathbb{R}^{\mathrm{d\times d}}$ of the linear layer to produce the final output $\mathrm{y} \in \mathbb{R}^{\mathrm{k\times d}}$. The overall expression from a mathematical point of view is shown in equation (6):

$$y = \left\{\sum\left(\sigma\left(xW_I\right) * xW_K\right) * RELU\left(xW_V\right)\right\} W_O \tag{6}$$

Among $\mathrm{c_x}$ is $\tilde{\mathrm{A}}\left(\mathrm{xW_I}\right)$ the $\mathrm{c_v}$ is $\sum\left(\tilde{\mathrm{A}}\left(\mathrm{xW_I}\right)\mathrm{*xW_K}\right)$. Moreover, * and $\sum$ are element-by-element multiplication and summation operations.

$\mathrm{H_o} = \mathrm{FFN}\left(\mathrm{y}\right)$, where the $\mathrm{FFN}\left(\mathrm{x}\right)$ The expression of (term) is shown in equation (7):

$$FFN\left(x\right) = \max\left(0, xW_1 + b_1\right) W_2 + b_2 \tag{7}$$

The token feed-forward network mainly helps us transform the sequence without all of the elements affecting each other; we set it up here as two elements where the relationship represents the $\widetilde{\mathrm{e_r}}$ and target entity $\mathrm{e_t}$. The specific flow of MVT2 is shown in Figure 2.

Figure 1 is the standard multi-headed attention (MHA) joint in transformers. Its complexity is $O\left(K^2\right)$; it is quadratic with the number of tags K and uses batch matrix multiplication to calculate attention, which consumes many computing resources. Figure 2 is linear and separable in complexity; the time complexity is $O\left(K\right)$, and the element-by-element multiplication is used. This method is more flexible and memory-saving in processing.

Improvement of the Loss Function

The loss function is an index to measure the difference between the model output and the actual value; we hope it can also contribute. Therefore, we introduce the L_b loss function and combine and optimize the binary cross-entropy loss function. In L_b, we expect to reach a certain fairness standard for different prediction results, so we add an average strategy to the original binary cross entropy function; at the same time, fairness constraints are added to the total loss function to reduce the risk of semantic discrimination. The original binary cross entropy loss function is shown in equation (8):

$$L = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \times \log\left(y_i\right) + \left(1 - y_i\right) \times \log\left(1 - p\left(y_i\right)\right)\right] \tag{8}$$

Figure 1. Multi-headed attention diagram

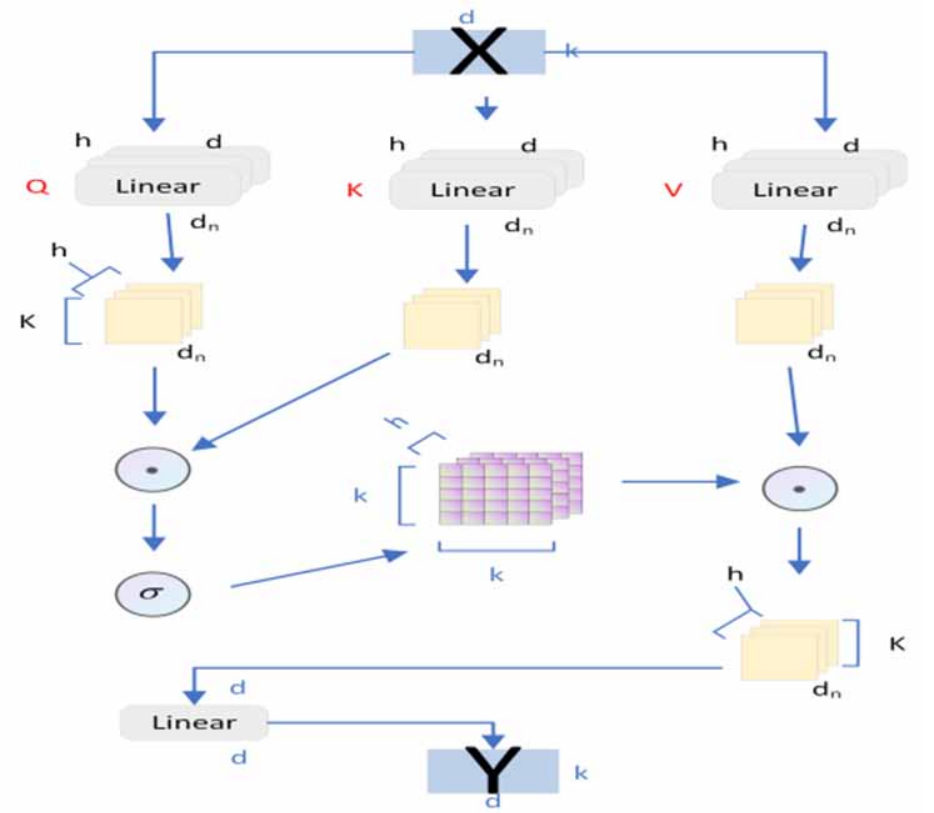
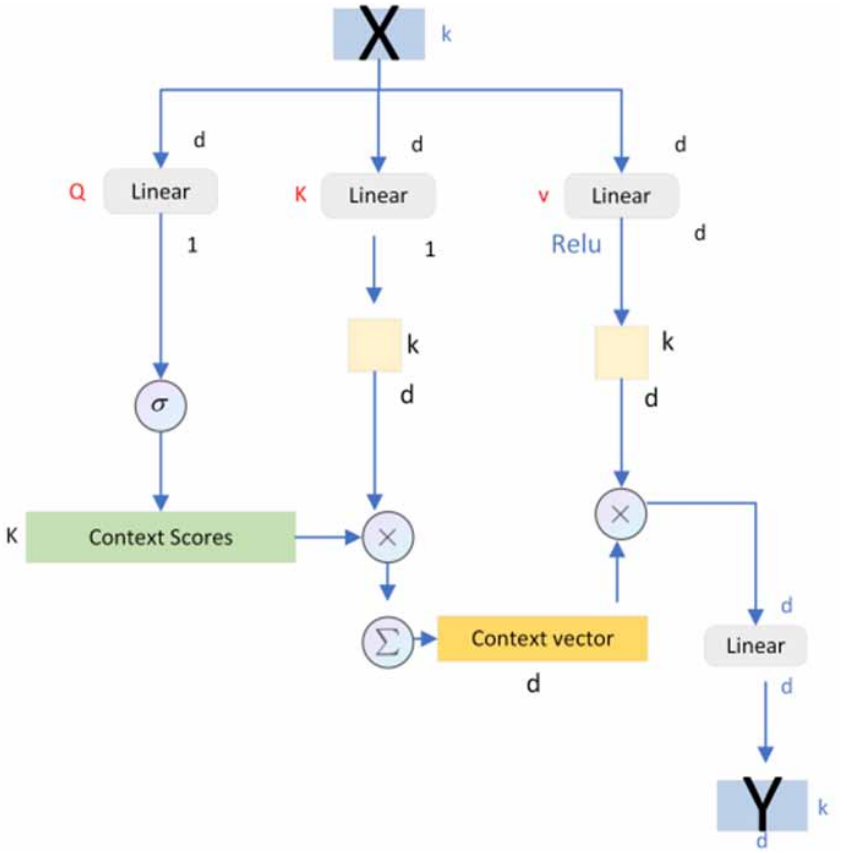


Figure 2. Separable self-attention diagram



Where y_i is the label, whose value is either 0 or 1, commonly understood as binary yes and no. $p(y_i)$ is the probability that the output belongs to y_i.

The introduced L_b loss function is shown in equation (9):

$$L_b = \frac{1}{N}\sum_{i=1}^{N} L\left(\left(p_i\right),\left(g_i\right)\right) \tag{9}$$

where g_i is the ground-truth value, and p_i is the predicted value. To better incorporate into our model, we redefine the binary cross-entropy function as shown in equation (10):

$$L_t = -\frac{1}{\left|N\right|}\sum_{t' \in N}\left[y_{t'} \times \log\left(p_{t'}\right) + \left(1 - y_{t'}\right) \times \log\left(1 - p_{t'}\right)\right] \tag{10}$$

where $p_{t'} = \tilde{A}\left(\phi\left(h, r, t'\right)\right)$ and $y_{t'} = \begin{cases} 1 & if\ t' = t, \\ 0 & otherwise. \end{cases}$

In the above equation, the $\tilde{A}(\cdots)$ is the logistic sigmoid function, where $\phi(h, r, t')$ is the score predicted by the decoder for different tail entities and N is the number of total entities. Then L_b the definition of (term) is shown in equation (11):

$$L_b = \frac{1}{N}\sum_{n=1}^{N} L_t \tag{11}$$

where L_t is the redefined binary cross-entropy function. Finally the L_t function and the L_b function are combined to obtain the total loss function L_S, defined as shown in equation (12):

$$Ls = \alpha L_t + \beta L_b \tag{12}$$

In L_S, we introduce α and β. As the fairness constraint of the loss function, it ensures that the model achieves a certain fairness standard for different prediction results and reduces the risk of semantic discrimination.

Improvements in Web Semantics

In order to avoid the problem of overgeneralization of our model, improve the accuracy of the given triplet real value (t) during the search process, and improve the results of the header entity (h), we adopted a strategy to improve the model. Specifically, we choose to introduce local negative examples into the training; that is, the inverse of each triplet (t, r^{-1}, h) is added to the training data as a negative example so as to achieve the diversity of the data set and reduce the dependence on the specific data distribution.

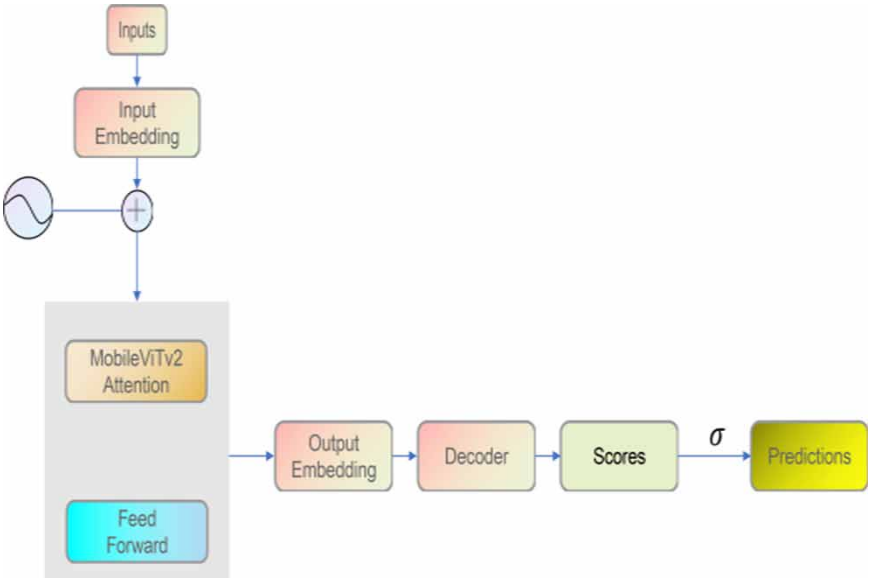
The purpose of introducing local negative cases is to make model learning distinguish the difference between positive cases and negative cases. By taking the inverse value of the triple as a negative example, the model will try to distinguish the positive case from the negative case so that the score of the positive case is higher and the score of the negative case is lower. In this way, the model can better capture the correlation between positive cases and reduce the false matching of negative cases.

The introduction of local negative examples makes the model more robust, provides generalization ability, and enables it to carry out reasoning and prediction more accurately in training. At the same time, we also adopt the early stop strategy. By monitoring the performance indicators of the verification set, we observe that when the performance of the verification set is no longer improved, we end the training in advance. Through the above training strategies, we can improve the generalization ability of the model, reduce overfitting, and make it better able to meet the challenges in the task of knowledge map completion.

The introduction of all the essential components of TFttOM is completed, and its overall flow is shown in Figure 3.

(1) Feed the source entities and relationships in the mapping into the transformer-based encoder after encoding their positions.
(2) Reduce time complexity with separable self-attention mechanisms in encoders.
(3) With the completion of the decoder, calculate and output the resulting score.
(4) Combine the resulting scores with a logistic S-function to make predictions.

Figure 3. The basic flowchart of the TFttOM model



EXPERIMENTAL ANALYSIS

Experimental Data Set and Partial Parameter Setting

We chose the more popular datasets FB15K-237 (Toutanova et al., 2015) and WN18RR (Dettmers et al., 2018), which are subsets of FB15k and WN18, respectively, to evaluate our model (Bordes et al., 2013). Compared to FB15K, the FB15K-237 dataset features a reduced number of relationships yet encompasses a greater number of triples, rendering the evaluation process for models more rigorous. Moreover, the variance in both the quantity and the categories of relationships means that FB15K and FB15K-237 differ in their respective application domains. For instance, FB15k boasts a broader array of relationship types spanning various sectors such as film, television, music, individuals, organizations, and locations. While FB15K-237 also touches upon these domains, it has been meticulously curated to place a stronger emphasis on applications in natural language reasoning, question-answering, and other similar fields. In the same vein, the WN18 and WN18RR datasets differ: WN18RR encompasses a higher count of missing relationships than WN18, which amplifies the complexity of relational reasoning and prediction. Additionally, WN18RR addresses certain inaccuracies and inconsistencies found in the WN18 dataset, thereby enhancing its reliability.

The FB15K and WN18 datasets are plagued by significant issues of test data leakage, which compromises their integrity for effective model evaluation. To mitigate this, a more challenging subset has been adopted as the experimental dataset in the hope of achieving superior performance across various types of knowledge graphs. Table 1 provides a detailed breakdown of these two datasets.

The experiment was carried out on a Linux system and NVIDIA GeForce 3090 GPU. In addition, there are some important parameter settings. In order to meet the requirement that our model can also achieve better performance when embedding in low dimensions, we chose 100 for the embedding dimension of entities and relationships, that is, $d_e = d_r = 100$, and we chose to use the mode of random search and adjustment for the remaining super parameters. First, we chose {128, 512, 1024, 2048} for the batch size. In order to speed up the training process and prevent certain overfitting, we changed the learning rate from {0.0001, 0.005, 0.001} and set the attenuation rate from {1, 0.995}.

Finally, our choice of batch size was 1024, the learning rate was 0.001, and the decay rate was 0.995. The loss function was set to 0.1, where α and β Set to 0.4 and 0.6, respectively.

Evaluation Indicators

In knowledge graph completion, metrics such as mean reciprocal rank (MRR) and Hit@N are commonly used to evaluate the performance of models across various datasets. MRR calculates the average reciprocal rank of the true triples. It considers the rankings of the actual triples and assigns higher scores to those ranked closer to the top. By taking the reciprocal of the rank, MRR focuses on the relative importance and position of the actual triples within the rankings. A higher MRR indicates better performance. Hit@N, however, measures the percentage of actual triples that appear within the top N ranks. It focuses on whether the actual triples are present within a specific range of rankings. For example, Hit@1 calculates the percentage of true triples that occupy the top rank. Hit@10 calculates the percentage of true triples within the top 10 ranks. Higher Hit@N values indicate a better ability to rank the true triples within the top positions. While MRR and Hit@N provide insights into the model's performance, MRR offers a more comprehensive evaluation perspective by considering the reciprocal ranks of all the true triples. It gives a broader assessment of the model's ability to rank the true triples accurately. So, we chose MRR as the primary evaluation metric.

Comparison of Experimental Results With Mainstream Models

We compared our approach with several state-of-the-art models in different categories. Among the path-based methods were Path-Ranking (Lao & Cohen, 2010) and NeuraILP (Yang et al., n.d.); among the embedding-based methods were RotatE (Sun et al., 2019), ConvE (Dettmers et al., 2018), QuatE (S. Zhang et al., 2019), TuckeR (Balažević et al., 2019), McRL (J. Wang et al., 2023), HRGAT (Liang et al., 2023), and GIE (Cao et al., 2022). Since the main framework of our model is based on the transformer, we also picked other transformer-based models, CokE (Q. Wang et al., 2020), HittER (S. Chen et al., 2021), and SattLE (Baghershahi et al., 2022) for comparison. The results of the experiments are shown in Table 2.

The experimental findings affirm the value of our strategies to diminish overgeneralization and error propagation through the incorporation of inverse values in the dataset and the refinement of the loss function. As illustrated in Table 2, the TFttOM model demonstrates superior performance over its contemporaries on the same dataset, particularly in lower dimensions. In the WN18RR dataset, our model outperforms competitors like RotaTE, ConvE, and QuatE with relative ease. Likewise, in the FB15K-237 dataset, our model surpasses other contenders, securing impressive results. Notably, against state-of-the-art models such as McRL, GIE, and HRGAT on the FB15K-237 dataset, our model, utilizing the Tucker decoder, achieves an MRR of 0.378. This represents a marginal yet notable improvement of 0.2% over McRL, 4.4% over GIE, and 3.3% over HRGAT. With the TwoMult decoder, our model attains an MRR of 0.364, which, while not markedly distinct, highlights our model's competitive edge, especially considering its lower embedding dimension. In the WN18RR dataset, our model with the TwoMult decoder outstrips recent models such as McRL, GIE, and HRGAT. In order to more intuitively understand the comparison between us and the latest models, we analyzed the advantages and disadvantages of each model in Table 3.

Table 1. Dataset statistics

Dataset	Entity	Relation	Train	Validation	Test
FB15K-237	14541	237	272115	17535	20466
WN18RR	40943	11	86835	3034	3134

Table 2. Knowledge graph completion results for FB15k-237 and WN18RR

Methods	DoE	FB15k-237				WN18RR			
		MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
Path-Ranking	-	0.174	0.119	0.186	0.285	0.324	0.276	0.360	0.406
NeuraILP	128	0.240	-	-	0.362	0.435	0.371	0.434	0.566
RotatE	1000	0.338	0.241	0.975	0.533	0.476	0.428	0.492	0.566
ConvE	200	0.319	0.232	0.351	0.492	0.462	0.431	0.476	0.525
QuatE	400	0.348	0.248	0.382	0.550	0.488	0.438	0.508	0.582
TuckER	200	0.358	0.266	0.394	0.544	0.470	0.433	0.482	0.526
McRL	400	0.377	0.283	0.376	0.559	0.491	0.477	0.530	0.576
GIE	250	0.362	0.271	0.401	0.552	0.491	0.452	0.505	0.575
HRGAT	200	0.366	0.271	0.404	0.542	0.491	0.454	0.503	0.567
CokE	256	0.364	0.272	0.400	0.549	0.484	0.450	0.496	0.553
HittER	320	0.373	0.279	0.409	0.558	0.503	0.462	0.516	0.584
SAttLE	100	0.360	0.268	0.396	0.545	0.491	0.454	0.508	0.558
TFttOM+TwoMult	100	0.364	0.285	0.425	0.570	0.494	0.458	0.510	0.562
TFttOM+Tucker	64	0.378	0.281	0.423	0.567	0.490	0.457	0.507	0.553

Note. DoE represents the dimension of the embedding. MRR is the main performance reference (larger results are better).

Table 3. Conduct an assessment of the merits and demerits exhibited by distinct models

Methods	Merits	Demerits
CokE	A more comprehensive integration of the intrinsic contextual characteristics endemic to entities and their interrelations	The fusion of contextual information introduces a significant increase in computational complexity, which poses scalability challenges in the processing of large-scale knowledge graphs
HittER	The hierarchical structure is used to help the model capture different levels of semantic information layer by layer	Involves the selection of multiple hyperparameters, such as the number of attention layers and the number of heads
SAttLE	In low-dimensional embedding, a single Transformer encoding can be used to quickly achieve strong scalability.	No regularization method is added to the model, which may lead to overfitting
McRL	Capture the complex conceptual information hidden in entities and relationships by projecting them into multiple vectors	No corresponding improvements have been made to the decoder
GIE	Interactive learning in Euclidean space, hyperbolic space and hyperspherical space to accurately mine the complex structure of KG	The problem of overfitting is not explained, so this problem may exist
HRGAT	More different modal information that integrates entities and relationships	Different modes lead to greater demand for embedding dimensions.

Based on the analysis in the table above, we will briefly discuss KGC's future research directions. First, multimodal fusion, integrating data types such as text and images, improving the richness of knowledge graphs, and better reflecting the real world. Second, further research will be conducted on dynamic or regularly updated knowledge graphs, known as temporal knowledge graph completion (TKGC), to reflect the latest developments in knowledge.

ABLATION EXPERIMENTS

Impact of Attention Mechanisms

In order to fully explore the quality of our model in different situations, we chose to compare SAttLE using the underlying multi-head attention module with the approach used in this paper. Since the focus of multi-head attention is on the number of heads, we chose several different numbers of heads for comparison. The experimental results are shown in Table 4.

As can be seen in Table 4, the performance of SAttLE grows to different degrees with the number of heads of attention replaced, where the MRR improves from 0.35 to 0.36 in dataset FB15K-237 and from 0.48 to 0.491 in dataset WN18RR. In contrast, there is no improvement in the performance when the number of heads is changed from 64 to 128, but the heads The greater the number of attentions, the greater the number of non-embedded free parameters (NFP) of the model grows, which affects the inference speed. Despite increasing the number of heads, our model consistently outperforms SAttLE. Thus, MVT2 achieves very significant results as the attention module of the model.

Impact of Embedding Dimensions

As it was proven in previous experimental results that our model can achieve great results and great scalability with low dimensional embeddings, we chose to continue to challenge the performance case of the model further at lower embedding dimensions. Here again, we chose DoEs of 100, 64, and 32. The results at FB15K-237 are shown in Figure 4.

Impact of the Loss Function

We hope that the improvements to the loss function will benefit the model, so the original loss function was selected to be compared with our improved loss function, replacing only the loss function when other factors that may affect it are the same. The results are shown in Figure 5.

As can be seen in Figure 5, as the number of training rounds increases, using our improved loss function is 3.488E-4 at an Epoch of 200, we can achieve better convergence concerning the original loss function's final convergence number of 3.635E-4, which results in a lower training error or loss value, and better generalization ability.

CONCLUSION AND FUTURE WORK

Completion model: TFttOM reduces computational complexity by improving the self-attention module and using simple and efficient decoding methods, using new loss functions to improve discriminative

Table 4. Comparison results of MRR between TFttOM and SattLE on FB15k-237 and WN18RR

Methods	Heads	FB15k-237	WN18RR
		MRR	MRR
SAttLE	8	0.350	0.480
	32	0.358	0.486
	64	0.360	0.491
	128	0.360	0.492
TFttOM	-	0.364	0.494

Note. Where d = 100 and heads in 8, 32, 64, 128

Figure 4. Experimental results of e model using different DoE cases

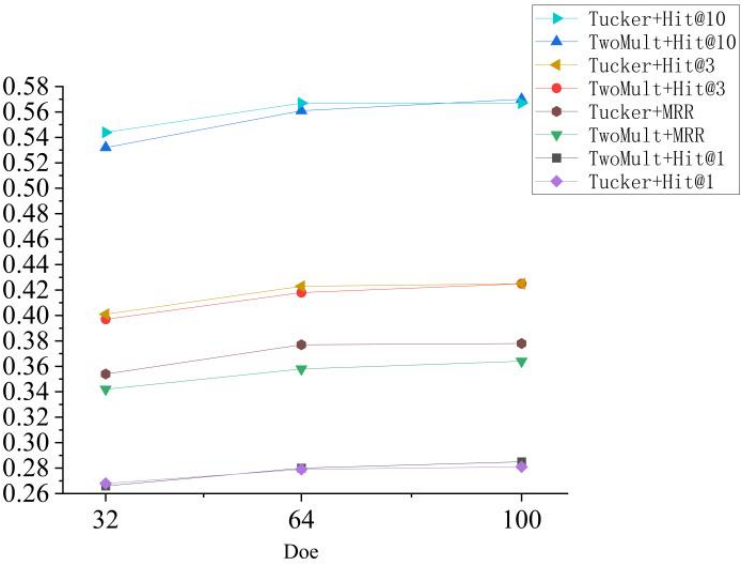
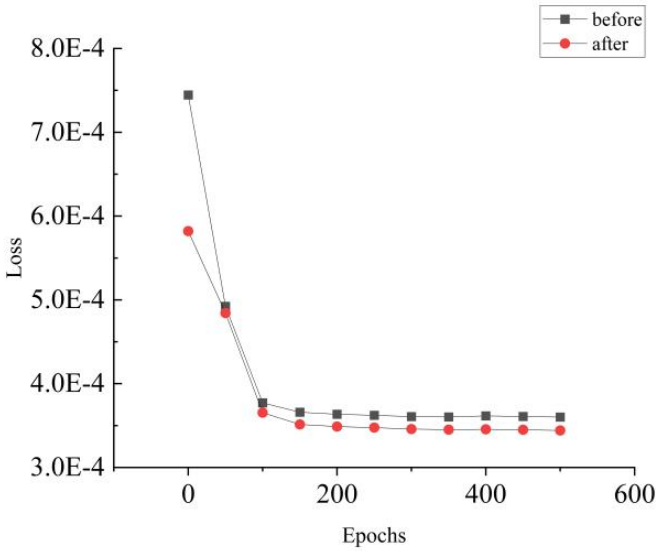


Figure 5. Before and after loss function replacement



features while reducing the risk of semantic discrimination and adding specific inverse values to the data set to reduce the risk of semantic discrimination.

Overfitting: Theoretically, the advantage of TFttOM is that it can achieve good performance in a low-dimensional state. Empirical evaluation of different data sets shows that TFttOM is competitive in knowledge graph completion performance compared with existing advanced models.

By comparing different models, we also found our shortcomings. In the future, we plan to reduce overfitting by adding regularization methods to our models and mining more hidden information in the data to achieve optimal .

AUTHOR NOTE

This study is supported by the National Natural Science Foundation of China (62072170), Key Topics of the Hunan Provincial Department of Education (23A0705), the 13th "Five-year Plan" of Educational Science in Hunan Province (xjk18cxx014), Key R&D Program of Hunan Province under Grant No. 2022SK2109, and Hunan Provincial Department of Education Outstanding Youth Fund under Grant No. 21B0850.

REFERENCES

Balaževic, I., Allen, C., & Hospedales, T. (2019). *Multi-relational poincaré graph embeddings*. Research Gate.

Balažević, I., Allen, C., & Hospedales, T. M. (2019). TuckER: Tensor factorization for knowledge graph completion. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. ACL. doi:10.18653/v1/D19-1522

Berners-Lee, T., Hendler, J., & Lassila, O. (2023). The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. In O. Seneviratne & J. Hendler (Eds.), *Linking the world's information* (1st ed., pp. 91–103). ACM. doi:10.1145/3591366.3591376

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). *Translating embeddings for modeling multi-relational data*. Research Gate.

Breit, A., Waltersdorfer, L., Ekaputra, F. J., Sabou, M., Ekelhart, A., Iana, A., Paulheim, H., Portisch, J., Revenko, A., Teije, A. T., & Van Harmelen, F. (2023). Combining machine learning and semantic web: A systematic mapping study. *ACM Computing Surveys*, *55*(14s), 1–41. doi:10.1145/3586163

Cao, Z., Xu, Q., Yang, Z., Cao, X., & Huang, Q. (2022). *Geometry interaction knowledge graph embeddings*. Research Gate.

Chen, X., Zhang, N., Li, L., Deng, S., Tan, C., Xu, C., Huang, F., Si, L., & Chen, H. (2022). Hybrid transformer with multi-level fusion for multimodal knowledge graph completion. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. doi:10.1145/3477495.3531992

d'Amato, C. (2020). Machine learning for the semantic web: Lessons learnt and next research directions. *Semantic Web*, *11*(1), 195–203. doi:10.3233/SW-200388

Dettmers, T., Minervini, P., Stenetorp, P., & Riedel, S. (2018). Convolutional 2D knowledge graph embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1). Advance online publication. doi:10.1609/aaai.v32i1.11573

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., & Zhang, W. (2014). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. doi:10.1145/2623330.2623623

Geng, D., Li, H., & Liu, C. (2022). Tiny-UKSIE: An optimized lightweight semantic inference engine for reasoning uncertain knowledge. *International Journal on Semantic Web and Information Systems*, *18*(1), 1–23. doi:10.4018/IJSWIS.300826

Hamilton, W., Ying, Z., & Leskovec, J. (2017). *Inductive representation learning on large graphs*. Research Gate.

Ji, G., He, S., Xu, L., Liu, K., & Zhao, J. (2015). *Knowledge graph embedding via dynamic mapping matrix.* Research Gate.

Lao, N., & Cohen, W. W. (2010). Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, *81*(1), 53–67. doi:10.1007/s10994-010-5205-8

Li, P., Li, T., Wang, X., Zhang, S., Jiang, Y., & Tang, Y. (2022). Scholar Recommendation Based on High-Order Propagation of Knowledge Graphs. *International Journal on Semantic Web and Information Systems*, *18*(1), 1–19. doi:10.4018/IJSWIS.313190

Li, X., Yang, D., Yuan, J., Donkers, A., & Liu, X. (2022). BIM-enabled semantic web for automated safety checks in subway construction. *Automation in Construction*, *141*, 104454. doi:10.1016/j.autcon.2022.104454

Liang, S., Zhu, A., Zhang, J., & Shao, J. (2023). Hyper-node relational graph attention network for multi-modal knowledge graph completion. *ACM Transactions on Multimedia Computing Communications and Applications*, *19*(2), 1–21. doi:10.1145/3545573

Lin, Y., Liu, Z., Sun, M., Liu, Y., & Zhu, X. (2015). Learning Entity and Relation Embeddings for Knowledge Graph Completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, *29*(1). Advance online publication. doi:10.1609/aaai.v29i1.9491

Manaa, N., Seridi, H., Beldjoudi, S., & Hadjadji, I. (2023). Enhancing decision making with semantic-based hybrid recommender system. *2023 International Conference on Decision Aid Sciences and Applications (DASA)*. IEEE. doi:10.1109/DASA59624.2023.10286649

Morshed, M. G., Sultana, T., & Lee, Y.-K. (2023). LeL-GNN: Learnable edge sampling and line based graph neural network for link prediction. *IEEE Access : Practical Innovations, Open Solutions*, *11*, 56083–56097. doi:10.1109/ACCESS.2023.3283029

Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., & Gamon, M. (2015). Representing text for joint embedding of text and knowledge bases. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. ACL. doi:10.18653/v1/D15-1174

Trouillon, T., Welbl, J., & Riedel, S. (2016). *Complex embeddings for simple link prediction*.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31*(3), 279–311. doi:10.1007/BF02289464 PMID:5221127

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*.

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACL. doi:10.1145/2939672.2939753

Wang, J., Wang, B., Gao, J., Hu, Y., & Yin, B. (2023). Multi-concept representation learning for knowledge graph completion. *ACM Transactions on Knowledge Discovery from Data*, *17*(1), 1–19. doi:10.1145/3533017

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24. doi:10.1109/TNNLS.2020.2978386 PMID:32217482

Yang, F., Yang, Z., & Cohen, W. W. (n.d.). *Differentiable learning of logical rules for knowledge based reasoning*.

Zhang, S., Tay, Y., Yao, L., & Liu, Q. (2019). *Quaternion knowledge graph embeddings*.

Zhang, X., Fang, Q., Hu, J., Qian, S., & Xu, C. (2022). TCKGE: Transformers with contrastive learning for knowledge graph embedding. *International Journal of Multimedia Information Retrieval*, *11*(4), 589–597. doi:10.1007/s13735-022-00256-3

Zhu, Y., Zhang, W., Chen, M., Chen, H., Cheng, X., Zhang, W., & Chen, H. (2022). DualDE: Dually distilling knowledge graph embedding for faster and cheaper reasoning. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. ACM. doi:10.1145/3488560.3498437