electronics

MDPI

Article

Efficient Medical Knowledge Graph Embedding: Leveraging Adaptive Hierarchical Transformers and Model Compression

Xuexiang Li, Hansheng Yang, Cong Yang * and Weixing Zhang

School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou 450001, China
* Correspondence: wangyuanyc@zzu.edu.cn

Abstract: Medical knowledge graphs have emerged as essential tools for representing complex relationships among medical entities. However, existing methods for learning embeddings from medical knowledge graphs, such as DistMult, RotatE, ConvE, InteractE, JointE, and ConvKB, may not adequately capture the unique challenges posed by the domain, including the heterogeneity of medical entities, rich hierarchical structures, large-scale, high-dimensionality, and noisy and incomplete data. In this study, we propose an Adaptive Hierarchical Transformer with Memory (AHTM) model, coupled with a teacher–student model compression approach, to effectively address these challenges and learn embeddings from a rich medical knowledge dataset containing diverse entities and relationship sets. We evaluate the AHTM model on this newly constructed "Med-Dis" dataset and demonstrate its superiority over baseline methods. The AHTM model achieves substantial improvements in Mean Rank (MR) and Hits@10 values, with the highest MR value increasing by nearly 56% and Hits@10 increasing by 39%. Furthermore, we observe similar performance enhancements on the "FB15K-237" and "WN18RR" datasets. Our model compression approach, incorporating knowledge distillation and weight quantization, effectively reduces the model's storage and computational requirements, making it suitable for resource-constrained environments. Overall, the proposed AHTM model and compression techniques offer a novel and effective solution for learning embeddings from medical knowledge graphs and enhancing our understanding of complex relationships among medical entities, while addressing the inadequacies of existing approaches.

Keywords: medical knowledge graph; adaptive hierarchical transformer; model compression

Citation: Li, X.; Yang, H.; Yang, C.; Zhang, W. Efficient Medical Knowledge Graph Embedding: Leveraging Adaptive Hierarchical Transformers and Model Compression. *Electronics* **2023**, *12*, 2315. https://doi.org/10.3390/electronics12102315

Academic Editor: Alberto Fernandez Hilario

Received: 6 April 2023
Revised: 5 May 2023
Accepted: 16 May 2023
Published: 20 May 2023

1. Introduction

Knowledge graphs (KGs) have emerged as a powerful tool for organizing and representing structured information in a wide range of domains. By capturing entities and their relationships in a graph-based structure, KGs enable intelligent systems to perform complex reasoning and querying tasks with ease [1]. General-purpose knowledge graphs, such as Freebase [2] and DBpedia [3], have been extensively used for numerous applications, including natural language processing, information retrieval, and recommendation systems [4].

In recent years, the focus has shifted towards domain-specific knowledge graphs, particularly medical knowledge graphs, due to their potential to represent complex relationships among various medical entities, such as diseases, drugs, symptoms, and diagnostic items, in a structured and interpretable format [5]. Medical knowledge graphs enable researchers and practitioners to analyze and extract valuable insights from the vast and ever-growing corpus of medical information available in the scientific literature, electronic health records, and clinical guidelines.

However, constructing and learning from medical knowledge graphs present unique challenges compared to general-purpose KGs. Some of these challenges include:

- **Heterogeneity of medical entities:** Medical knowledge graphs contain a variety of medical entities, such as diseases, drugs, symptoms, diagnostic items, with diverse

and complex relationships between them [6]. Traditional models may struggle to capture these heterogeneous relationships, leading to suboptimal embeddings.

- **Rich hierarchical structure:** Medical entities often exhibit hierarchical structures, such as disease categories or drug classifications. Existing approaches may not fully capture these hierarchical relationships, limiting the quality of the learned embeddings.
- **Noisy and incomplete data:** Medical knowledge graphs are often noisy and incomplete due to the vast and constantly evolving nature of medical information. Traditional models may struggle to handle such inconsistencies, affecting the quality of the learned embeddings [7].
- **Large-scale and high-dimensionality:** Medical knowledge graphs can be large in scale and high in dimensionality, making it challenging for traditional models to efficiently learn embeddings.

Several state-of-the-art knowledge graph embedding (KGE) models have been proposed to address these challenges, including DistMult [8], RotatE [9], ConvE [10], InteractE [11], JointE [12], and ConvKB [13]. These models have achieved remarkable success in general-purpose knowledge graph datasets, such as FB15K-237 [14] and WN18RR [10]. However, their performance in specialized domains such as medical knowledge graphs may not be as satisfactory due to unique challenges associated with the representation and reasoning of medical entities and relationships. Existing methods may not adequately capture the complex relationships, hierarchical structures, and heterogeneity of medical entities, nor address the noisy, incomplete data and high-dimensionality often found in medical knowledge graphs.

To address these challenges, this paper introduces a novel Adaptive Hierarchical Transformer with Memory (AHTM) model that leverages the Transformer architecture [15] and a memory-augmented mechanism, specifically designed for learning embeddings from medical knowledge graphs. Our main contributions are as follows:

- We propose a novel AHTM model to address the issue of medical entity heterogeneity in medical knowledge graphs. The model introduces a JointAttention function. This function effectively integrates self-attention and joint attention mechanisms, enabling the model to better capture and represent various relationships within the medical knowledge graph.
- In order to tackle the challenge posed by the rich hierarchical structure present in medical knowledge graphs, the AHTM model incorporates a hierarchical architecture and residual blocks. These design choices facilitate more effective capture and representation of the intricate hierarchical relationships inherent in medical knowledge graphs.
- To tackle the issue of noisy and incomplete data, the AHTM model incorporates adaptive mechanisms along with a memory storage module. This combination allows the model to effectively handle such data, leading to more robust and accurate embeddings.
- To address the challenge of large-scale and high-dimensional data in medical knowledge graphs, the AHTM model employs a teacher–student model compression approach. This method integrates knowledge distillation [16] and weight quantization [17] techniques to reduce both storage and computational demands of the model. Consequently, the AHTM model becomes capable of effectively learning embeddings from large-scale, high-dimensional medical knowledge graphs.
- We conduct extensive experiments to evaluate the performance of the proposed AHTM model on our newly constructed medical knowledge atlas dataset, as well as on the FB15K-237 and WN18RR datasets. The experimental results demonstrate the superior performance of the AHTM model compared to the baseline methods, with significant improvements in Mean Rank (MR) and Hits@10 values.

The rest of the paper is organized as follows. Section 2 reviews related work in the fields of knowledge graph embedding and model compression. Section 3 describes the methodology, including the proposed AHTM model, teacher–student model compression approach, and a Neural Turing Machine. Section 4 presents the experimental setup and results. Finally, Section 5 concludes the paper and outlines potential future research directions.

2. Related Works

In this section, we discuss related work in the areas of knowledge graph embedding methods, medical knowledge graphs, Transformer attention mechanisms and their applications in the knowledge graph domain, and model compression techniques such as knowledge distillation and weight quantization.

2.1. Knowledge Graph Embedding Methods

Knowledge graph embedding (KGE) models aim to learn low-dimensional representations of entities and relationships in a knowledge graph, which can be used for various tasks, including link prediction, entity resolution, and KG completion [18,19]. Several KGE models have been proposed in recent years, each with its unique strengths and limitations.

TransE [20] is one of the pioneering KGE methods, which models relationships as translations in the embedding space. However, TransE struggles to model complex relationships and capture symmetry, antisymmetry, and inversion patterns. DistMult [8] models relationships using element-wise multiplications of entity embeddings but is limited in modeling asymmetric relationships. RotatE [9] models relationships as rotations in the complex embedding space, which captures symmetry, antisymmetry, and inversion patterns but has a high computational cost. ConvE [10] employs convolutional neural networks to model relationships, while InteractE [11] extends ConvE with a more expressive interaction mechanism. JointE [12] leverages joint learning of entity and relationship embeddings, ConvKB [13] and DyConvNE [21] integrates convolutional neural networks into knowledge base completion tasks.

While these models have achieved remarkable success in general-purpose knowledge graph datasets, they may not adequately address the aforementioned research questions and challenges specific to medical knowledge graphs. Therefore, our study aims to develop an Adaptive Hierarchical Transformer with Memory (AHTM) model tailored for medical knowledge graphs, which effectively tackles the heterogeneity of medical entities, rich hierarchical structures, large-scale and high-dimensionality, and noisy and incomplete data. By doing so, we seek to significantly improve the performance of KGE models in the medical domain and contribute to a deeper understanding of the complex relationships among medical entities.

2.2. Medical Knowledge Graphs

Medical knowledge graphs have been developed to represent structured medical information in a graph-based format. It has been shown to be effective for various medical applications, including clinical decision support [22,23], drug repurposing [24], and symptom-disease inference [25]. Several medical knowledge graphs have been proposed in the literature, such as the UMLS Metathesaurus [26], DrugBank [27], and Hetionet [28].

Despite the growing interest in Medical knowledge graphs, the development of KGE models specifically designed for the medical domain remains an open research problem. In light of the growing interest in medical knowledge graphs and the open research problem of developing KGE models specifically designed for the medical domain, our work aims to investigate the limitations of existing KGE models in the context of medical knowledge graphs and identify the unique challenges associated with the representation of medical entities and relationships. We will evaluate the performance of the AHTM model on a newly constructed "Med-Dis" dataset and compare its performance against existing state-of-the-art KGE models. By addressing these objectives, we hope to significantly advance the development of KGE models for medical knowledge graphs and contribute to a deeper understanding of the complex relationships among medical entities.

2.3. Transformer Attention Mechanism and Its Applications in Knowledge Graphs

The Transformer model, proposed by Vaswani et al. [15], has revolutionized natural language processing with its self-attention mechanism, which allows the model to capture long-range dependencies and contextual information efficiently. Transformers have been

successfully applied to a variety of NLP tasks, such as machine translation, text summarization [29], and question-answering [30]. Recently, researchers have started exploring the application of Transformer-based models in the knowledge graph domain.

Graph Attention Networks (GAT) [31,32] adapt the attention mechanism for graph-structured data, allowing nodes to selectively focus on their neighbors. However, GAT suffers from scalability issues due to its complexity. Transformer-KGE [33] integrates the Transformer architecture into KGE tasks, leveraging its attention mechanism for better relational reasoning. Graph Transformer Networks (GTN) [34,35] generalize the Transformer model for graph-structured data, enabling efficient representation learning for graphs.

These models demonstrate the potential of leveraging the Transformer's attention mechanism for knowledge graph embedding and reasoning tasks, although their scalability and complexity remain open challenges. In this work, our research objectives involve developing an Adaptive Hierarchical Transformer with Memory (AHTM) model that effectively leverages attention mechanisms while addressing scalability and complexity challenges. We aim to evaluate the performance of the AHTM model on a newly constructed medical knowledge graph dataset, comparing its performance against existing Transformer-based models and other state-of-the-art KGE models. Furthermore, we will explore the potential real-world applications of the learned embeddings in the medical domain.

2.4. Model Compression Methods

Model compression techniques aim to reduce the storage and computational requirements of deep learning models while maintaining their performance. Two popular model compression methods are knowledge distillation [16,36] and weight quantization [17,37].

Knowledge distillation involves training a smaller student model using the knowledge acquired by a larger teacher model, allowing the student model to achieve competitive performance with a reduced model size. However, the effectiveness of knowledge distillation depends on the quality of the teacher model and the choice of the student model's architecture.

Weight quantization, on the other hand, reduces the numerical precision of the model parameters, leading to significant reductions in both model size and computational requirements. Various weight quantization techniques have been proposed, including binary [38], ternary [39], and vector quantization [40]. Despite the advantages of weight quantization, it may introduce quantization errors that can affect the model's performance, especially when extreme quantization levels are applied.

These model compression techniques have been widely used in various deep learning domains, such as computer vision and natural language processing. However, their application in the context of knowledge graph embeddings and medical knowledge graphs remains relatively unexplored, with ample opportunities for further investigation and improvement.

In the realm of model compression techniques for knowledge graph embeddings, our research objectives include examining the effectiveness of knowledge distillation and weight quantization methods in reducing the storage and computational requirements of deep learning models while preserving their performance, particularly in the context of medical knowledge graphs. We will develop a teacher–student model compression approach for our Adaptive Hierarchical Transformer with Memory (AHTM) model, utilizing knowledge distillation and weight quantization techniques to create a more resource-efficient model. Our research questions involve identifying the potential challenges and trade-offs associated with applying these model compression techniques to knowledge graph embeddings.

3. Methodology

In this section, we describe the methodology employed for knowledge graph embedding using the Med-DiseaseKG dataset, which is developed based on the format of the FB15k-237 dataset. Our proposed approach combines a novel data input module, which incorporates a convolution operation and a residual network, with the Adaptive Hierarchical Transformer with Memory (AHTM) architecture. Furthermore, we employ a

teacher–student model using knowledge distillation and weight quantization methods for model compression. The overall pipeline consists of four main components: (1) Data Input Module, (2) AHTM Module, (3) Residual Block, and (4) Compression Model.

3.1. Data Input Module

The Data Input Module plays a crucial role in processing the knowledge graph triples and generating suitable input representations that can be effectively utilized by the AHTM module. Considering a triple (h, r, t), where h denotes the head entity, r represents the relation, and t corresponds to the tail entity, the module carries out the subsequent operations, as illustrated in Figure 1, and we perform the following operations:

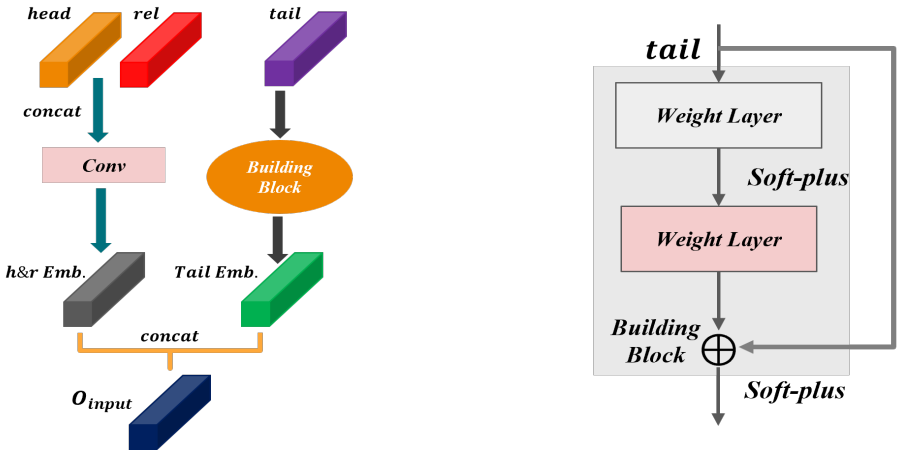

Figure 1. Input Layer: composite embedding of header entities and relationships and tail entity embedding using Building Block.

1. **Concatenation-1:** We concatenate the embedding vectors of the head entity and the relation, resulting in a combined representation $\boldsymbol{z} = [h, r]$, where $[\cdot;\cdot]$ denotes the concatenation operation.
2. **Convolution:** we apply a convolution operation on the concatenated representation z to obtain a complex embedding phasor for the head entity and the relation. The convolution operation can be mathematically defined as:

$$\boldsymbol{p}_{hr} = \boldsymbol{W}_c * \boldsymbol{z} + \boldsymbol{b}_c \tag{1}$$

   where $\boldsymbol{W}_c$ is the convolutional kernel, $*$ denotes the convolution operation, and $\boldsymbol{b}_c$ is the bias term.
3. **Tail Entity Embedding:** we employ a residual network with a softplus activation function $\sigma(x) = \log(1 + e^x)$ to obtain the embedding vector of the tail entity:

$$\boldsymbol{t}' = \text{BuildingBlock}(\boldsymbol{t}) \tag{2}$$

   where BuildingBlock$(\cdot)$ represents the residual network.
4. **Concatenation-2:** we concatenate the complex embedding phasor of the head entity and the relation, $\boldsymbol{p}_{hr}$, with the embedding vector of the tail entity, t', to form the output of the data input module:

$$\boldsymbol{O}_{\text{input}} = [\boldsymbol{P}_{hr}; \boldsymbol{t}'] \tag{3}$$

3.2. AHTM Module

The AHTM module receives the output of the Data Input Module, denoted as O_{input}, as its input, and processes it through a streamlined Adaptive Hierarchical Transformer with Memory architecture. Comprising key components such as Tree-based Encoding, Joint Attention, Neural Turing Machine (NTM), and Adaptive Hierarchical Transformer Layers,

the AHTM module is designed to effectively capture and process complex relationships within the knowledge graph. A comprehensive depiction of the AHTM module and its components can be found in Figure 2. The following sections provide a detailed exploration of each constituent element.

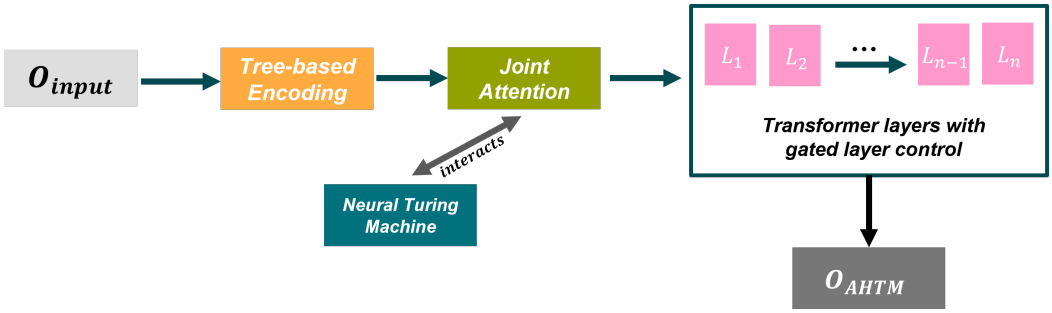

Figure 2. Adaptive-Hierarchical Transformer with Memory architecture.

3.2.1. Tree-Based Encoding

The Tree-based Encoding component, denoted as TreeEnc, is designed to capture the hierarchical structure of the input data. Given the input representation o_{input}, TreeEnc generates a tree-structured representation, which can be mathematically expressed as:

$$\boldsymbol{T} = \text{TreeEnc}(\boldsymbol{O}_{\text{input}}) \tag{4}$$

where $\boldsymbol{T}$ is the tree-structured representation of the input data. The TreeEnc uses the correlation function of the Tree-LSTM model. Given an input node x and its children nodes $c_1, c_2, \ldots, c_n$, we have the following equations for Tree-LSTM:

$$i_x = \sigma(W^{(i)}x + U^{(i)}h_{c1} + \cdots + U^{(i)}h_{cn} + b^{(i)}) \tag{5}$$

$$f_{xk} = \sigma(W^{(f)}x + U^{(f)}h_{ck} + b^{(f)}) \tag{6}$$

$$o_x = \sigma(W^{(o)}x + U^{(o)}h_{c1} + \cdots + U^{(o)}h_{cn} + b^{(o)}) \tag{7}$$

$$u_x = \tanh(W^{(u)}x + U^{(u)}h_{c1} + \cdots + U^{(u)}h_{cn} + b^{(u)}) \tag{8}$$

$$c_x = i_x \odot u_x + \sum_{k=1}^{n} f_{xk} \odot c_k \tag{9}$$

$$h_x = o_x \odot \tanh(c_x) \tag{10}$$

In the Tree-LSTM model, we compute the input gate i_x, forget gates f_{xk}, output gate o_x, and cell update gate u_x using the input node x and its children nodes $c_1, c_2, \ldots, c_n$. The cell state c_x is updated using the input, forget, and cell update gates. Finally, the hidden state h_x is computed using the output gate and the updated cell state.

3.2.2. Joint Attention

The Joint Attention component processes the hierarchical representations generated by the Tree-based Encoding module. It combines multi-head self-attention and joint attention mechanisms to incorporate information from different levels of the tree structure. The output of the Joint Attention module, denoted as $\boldsymbol{J}$, can be calculated as:

$$\boldsymbol{J} = \text{JointAttention}(\boldsymbol{T}), \tag{11}$$

where JointAttention$(\cdot)$ represents the Joint Attention module. The Joint Attention module combines self-attention mechanisms and joint attention mechanisms to enhance the model's ability to capture both extra and inter relationships among entities. The JointAttention function is formulated as follows:

1. **Self-attention mechanism:** Given a set of input embeddings $X = x_1, x_2, \ldots, x_n$, the self-attention mechanism computes the attention scores between each pair of input embeddings as follows:

$$\alpha_{ij} = \frac{exp(\text{score}(x_i, x_j))}{\sum_{k=1}^{n} exp(\text{score}(x_i, x_k))} \tag{12}$$

where $\text{score}(x_i, x_j)$ is a function that computes the attention score between x_i and x_j. The scoring function is formulated as follows:

$$\text{score}(x_i, x_j) = x_i^T W x_j \tag{13}$$

where W is a learnable weight matrix.

2. **Joint attention mechanism:** in addition to self-attention, we incorporate a joint attention mechanism that leverages external context information $C = c_1, c_2, \ldots, c_m$ to guide the attention process. The joint attention mechanism computes the attention scores as follows:

$$\beta_{ij} = \frac{exp(\text{score}(x_i, x_j, C))}{\sum_{k=1}^{n} exp(\text{score}(x_i, x_k, C))} \tag{14}$$

where $\text{score}(x_i, x_j, C)$ is a function that computes the joint attention score between x_i and x_j considering the external context information C.

3. **Combining self-attention and joint attention mechanisms:** the JointAttention function combines the self-attention and Joint Attention mechanisms using a weighted sum:

$$\text{JointAttention}(X, C) = \lambda \cdot \alpha + (1 - \lambda) \cdot \beta \tag{15}$$

where λ is a learnable scalar that balances the contribution of self-attention and joint attention mechanisms.

By combining self-attention and Joint Attention mechanisms, the JointAttention function allows the model to effectively capture both intra- and inter-relationships among the entities, while considering the external context information.

3.2.3. Neural Turing Machine (NTM)

The Neural Turing Machine (NTM) is an external memory mechanism that augments the AHTM module by allowing it to store and retrieve information. The NTM interacts with the Joint Attention output through read and write operations. The memory state of the NTM, denoted as $\boldsymbol{M}$, can be updated as:

$$\boldsymbol{M}_{t+1} = \text{NTM}(\boldsymbol{J}_t, \boldsymbol{M}_t), \tag{16}$$

where t denotes the current time step and $\text{NTM}(\cdot, \cdot)$ represents the Neural Turing Machine.

The NTM module is based on the Neural Turing Machine. Given a controller state h_t, the read and write operations are defined as follows:

$$k_t^r, k_t^w, \beta_t^r, \beta_t^w, g_t^r, g_t^w, s_t^r, s_t^w, \gamma_t^r, \gamma_t^w = \text{Controller}(h_t) \tag{17}$$

$$w_t^r = \text{ReadHead}(k_t^r, \beta_t^r, g_t^r, s_t^r, \gamma_t^r) \tag{18}$$

$$w_t^w = \text{WriteHead}(k_t^w, \beta_t^w, g_t^w, s_t^w, \gamma_t^w) \tag{19}$$

$$M_t = \text{Write}(M_{t-1}, w_t^w) \tag{20}$$

$$r_t = \text{Read}(M_t, w_t^r) \tag{21}$$

The Neural Turing Machine module consists of a controller, read and write heads, and a memory matrix M. The controller takes the current state h_t as input and computes the parameters for the read and write heads. The read and write weights w_t^r and w_t^w are

computed by the ReadHead and WriteHead functions. The memory matrix M_t is updated using the write weights, and the read vector r_t is computed using the read weights.

3.2.4. Adaptive Hierarchical Transformer Layers

The Adaptive Hierarchical Transformer Layers consist of a stack of Transformer layers with gated layer control. Each layer in the stack processes the output from the NTM and the previous layer. The output of the i-th layer, denoted as $\boldsymbol{H}_i$, can be computed as:

$$\boldsymbol{H}_i = \text{TransformerLayer}_i(\boldsymbol{H}_{i-1}, \boldsymbol{M}), \tag{22}$$

where TransformerLayer$i(\cdot,\cdot)$ represents the i-th Transformer layer, and $\boldsymbol{H}_0 = \boldsymbol{J}$ is the initial input to the stack.

The TransformerLayer combines standard Transformer layers with gating mechanisms. Given an input matrix $X \in \mathbb{R}^{n\times d}$, the layer is computed as follows:

$$M_1 = \text{LayerNorm}(X + \text{MultiHeadAttention}(X, X, X)) \tag{23}$$

$$M_2 = \text{LayerNorm}(M_1 + \text{FFN}(M_1)) \tag{24}$$

$$Z = \sigma(W_z M_2 + b_z) \tag{25}$$

$$O = Z \odot M_2 + (1 - Z) \odot X \tag{26}$$

The TransformerLayer starts with a multi-head attention operation on the input matrix X. The output of the attention operation is added to the input and normalized using LayerNorm, resulting in the intermediate matrix M_1. A feed-forward network (FFN) is applied to M_1, and the output is added to M_1 and normalized, resulting in M_2. A gating mechanism is applied using a sigmoid activation function (σ) to produce the gate values Z. Finally, the output O is computed as a linear combination of M_2 and X, controlled by the gate values Z.

The gated layer control mechanism dynamically adjusts the contribution of each layer based on the input data. The final output of the Adaptive Hierarchical Transformer Layers, denoted as $\boldsymbol{H}_{\text{final}}$, is a weighted sum of the outputs from all layers:

$$\boldsymbol{H}_{\text{final}} = \sum_{i=1}^{N} \alpha_i \boldsymbol{H}_i, \tag{27}$$

where N is the total number of Transformer layers, and α_i is the gating weight for the i-th layer, calculated using a gating mechanism:

$$\alpha_i = \sigma(\boldsymbol{W}_\alpha \boldsymbol{H}_i + \boldsymbol{b}_\alpha), \tag{28}$$

where $\sigma(\cdot)$ is the sigmoid activation function, $\boldsymbol{W}_\alpha$ is the weight matrix, and $\boldsymbol{b}_\alpha$ is the bias term.

Finally, the output of the AHTM module, denoted as $\boldsymbol{O}_{\text{AHTM}}$, is computed by applying a linear transformation followed by a softmax activation function on $\boldsymbol{H}_{\text{final}}$:

$$\boldsymbol{O}_{\text{AHTM}} = \text{softmax}(\boldsymbol{W}_o \boldsymbol{H}_{\text{final}} + \boldsymbol{b}_o), \tag{29}$$

where $\boldsymbol{W}_o$ and $\boldsymbol{b}_o$ are the weight matrix and bias term for the output layer, respectively.

In summary, the AHTM module processes the input representations generated by the Data Input Module, leveraging a hierarchical structure and external memory to capture complex relationships in the knowledge graph. The Tree-based Encoding, Joint Attention, Neural Turing Machine, and Adaptive Hierarchical Transformer Layers work in conjunction to produce the final output, $\boldsymbol{O}_{\text{AHTM}}$, which is subsequently used in the residual block and teacher–student model components.

3.3. Residual Block

The residual block component is introduced between the Data Input Module and the final output of the AHTM module to facilitate more efficient learning and better gradient flow. The rationale behind using a residual connection is to allow the model to learn a more direct mapping between the input and output, which can alleviate the vanishing gradient problem that may occur in deep networks, as demonstrated by He et al. (2016) [41] in the context of residual networks (ResNets) for image classification.

Given the output of the Data Input Module, $\boldsymbol{O}_{\text{input}}$, and the output of the AHTM Module $\boldsymbol{O}_{\text{AHTM}}$, the Residual Block computes the final output of the model, $\boldsymbol{O}_{\text{final}}$, as follows:

$$\boldsymbol{O}_{\text{final}} = \boldsymbol{O}_{\text{AHTM}} + F(\boldsymbol{O}_{\text{input}}), \tag{30}$$

where $F(\cdot)$ is a learnable function that transforms $\boldsymbol{O}_{\text{input}}$ to match the dimensions and the latent feature space of $\boldsymbol{O}_{\text{AHTM}}$. The function $F(\cdot)$ can be defined as:

$$F(\boldsymbol{O}_{\text{input}}) = \boldsymbol{W}_F \boldsymbol{O}_{\text{input}} + \boldsymbol{b}_F \tag{31}$$

where $\boldsymbol{W}_F$ is the weight matrix and $\boldsymbol{b}_F$ is the bias term associated with the transformation function.

The residual connection allows the gradients to flow more easily through the network during backpropagation, which can help the model learn more effectively, especially in deeper architectures. By combining the Data Input Module and the AHTM module with a residual connection, the model can leverage both the initial representations and the higher-level features captured by the AHTM module, resulting in improved overall performance.

3.4. Compression Model

Model compression seeks to decrease the storage and computational demands of deep learning models without compromising their performance. In the proposed approach, we implement a teacher–student model incorporating knowledge distillation and weight quantization methods to achieve effective model compression. A schematic representation of the model compression process, including its key components, can be found in Figure 3. A comprehensive examination of the individual techniques employed is provided in the subsequent sections.

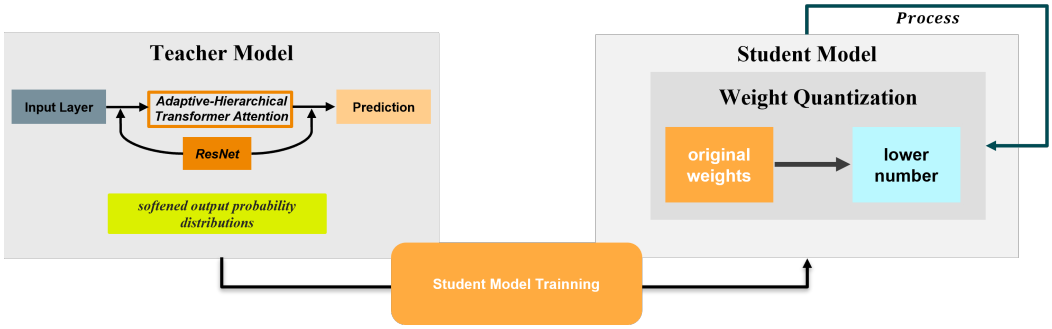

Figure 3. Teacher–student architecture.

3.4.1. Knowledge Distillation

The teacher–student model, also known as model distillation, involves training a smaller student model to mimic the behavior of a larger, more accurate teacher model. The basic idea is to transfer the knowledge from the teacher model to the student model, allowing the student model to achieve comparable performance with reduced complexity.

Given the output of the teacher model, $\boldsymbol{O}_{\text{teacher}}$, and the output of the student model, $\boldsymbol{O}_{\text{student}}$, we compute the knowledge distillation loss, L_{KD} as the Kullback–Leibler (KL) divergence between the softened probability distributions of the two models:

$$L_{\text{KD}} = \text{KL}\left(\frac{\text{softmax}(\boldsymbol{O}_{\text{teacher}}/T)}{\text{softmax}(\boldsymbol{O}_{\text{student}}/T)}\right), \tag{32}$$

where $\boldsymbol{T}$ is the temperature parameter that controls the softness of the probability distributions. A higher temperature value results in softer distributions, which can facilitate better knowledge transfer from the teacher model to the student model.

During training, the student model is optimized to minimize a weighted combination of the knowledge distillation loss and the original task loss, L_{task}:

$$L_{\text{total}} = \alpha L_{\text{KD}} + (1-\alpha)L_{\text{task}}, \tag{33}$$

where α is a hyperparameter that controls the trade-off between the two loss terms.

3.4.2. Weight Quantization

Weight quantization is a model compression technique that reduces the precision of the model's weights, thereby reducing the storage and computational requirements. Given a full-precision weight matrix, $\boldsymbol{W}$, the quantized weight matrix, $\boldsymbol{W}_{\text{quant}}$, can be computed as:

$$\boldsymbol{W}_{\text{quant}} = Q(\boldsymbol{W}, b), \tag{34}$$

where $Q(\cdot,\cdot)$ is a quantization function that maps the full-precision weights to a lower-precision representation with b bits. To maintain the accuracy of the model during the quantization process, a set of scale factors, $\boldsymbol{S}$, is used to rescale the quantized weights:

$$\boldsymbol{W}_{\text{rescaled}} = \boldsymbol{S} \odot \boldsymbol{W}_{\text{quant}}, \tag{35}$$

where $\odot$ denotes the element-wise multiplication operation.

The scale factors, $\boldsymbol{S}$, and the quantized weights, $\boldsymbol{W}_{\text{quant}}$, are learned during the training process by minimizing the quantization error, which is the difference between the full-precision weights and the rescaled quantized weights:

$$L_{\text{quant}} = |\boldsymbol{W} - \boldsymbol{W}_{\text{rescaled}}|^2 \tag{36}$$

By combining knowledge distillation and weight quantization, we can effectively compress the model, reducing its storage and computational requirements while maintaining its performance. The compressed student model can then be used in resource-constrained environments or for faster inference.

In summary, the proposed model compression technique integrates a teacher–student model with knowledge distillation and weight quantization to create a compact version of the original model. This compressed model can achieve comparable performance with reduced complexity, making it more suitable for deployment in various practical scenarios with limited computational resources or strict latency requirements.

4. Experiments and Results

4.1. Experimental Setting

The experimental setup was carefully designed to ensure the reliability and validity of the results. The experiments were conducted using Python 3.8 as the programming language, while the deep learning framework PyTorch 1.12.0 was utilized for model implementation and training. The configuration of the model parameters was chosen based on empirical studies and best practices in the literature. Specifically, the following settings were adopted for the experiments:

Training epochs: The model was trained for a total of 2000 rounds (epochs) to ensure sufficient exposure to the data and adequate convergence of the model's parameters. This choice was informed by the literature and our preliminary experiments, which indicated that 2000 rounds were adequate to achieve a stable performance.

Batch size: A batch size of 256 was selected to balance the trade-off between computational efficiency and convergence speed. This choice allowed for effective parallelization of the training process, thereby reducing the training time while maintaining the model's performance.

Embedding size: The size of the entity and relation embeddings was set to 200, providing an adequate representation capacity for capturing the complex semantics of the medical knowledge graph. This choice was informed by the literature and previous experiments, which have demonstrated that an embedding size of 200 offers a suitable trade-off between model complexity and expressiveness.

Learning rate: A learning rate of 0.002 was chosen to optimize the model's convergence rate while minimizing the risk of overshooting the optimal solution. This learning rate value allowed the model to adapt quickly to the training data while retarding potential issues such as oscillations or divergence.

Label smoothing: To address the issue of overfitting and improve the model's generalization capability, label smoothing with a parameter value of 0.2 was employed. This technique has been shown to enhance the performance of deep learning models by encouraging the model to assign non-zero probabilities to incorrect class labels, thereby mitigating overconfidence in its predictions.

4.2. Experimental Dataset

We evaluate the performance of our proposed model using a diverse set of datasets, including the self-constructed medical knowledge graph dataset "Med-Dis" and the widely used FB15K-237 and WN18RR datasets. This approach allows us to assess not only the model's effectiveness in the specific medical domain but also its generalization capabilities across different knowledge graph structures and domains.

The data in our self-constructed "Med-Dis" dataset are derived from various medical encyclopedia health websites and relevant resources. Following our curation process, a relatively comprehensive and structured dataset has been assembled. The dataset is formatted similarly to the FB15K-237 and WN18RR datasets, using structured triplets to represent entities and relationships. The "Med-Dis" dataset comprises five types of entities: Disease, Drug name, Symptom name, Diagnostic item, and Department name. The number of entities in each category is as follows: 8372 Diseases, 3729 Drug names, 6203 Symptom names, 3201 Diagnostic items, and 52 Department names.The following provides our brief description of each entity type:

- **Disease:** This entity type represents a different disease name, such as emphysema, gastric ulcer, liver cancer, heart disease, etc.
- **Drug name:** This entity type represent various drugs for the treatment of corresponding diseases, such as aspirin, acetaminophen, metronidazole, norfloxacin, etc.
- **Symptom names:** This entity type represents the names of various symptoms associated with different diseases, such as headache, fever, nausea, vomiting, chest tightness, etc.
- **Diagnostic items:** This entity type represents diagnostic items associated with various diseases, such as blood tests, B-mode ultrasounds, magnetic resonance imaging (MRI), computed tomography (CT) scans, etc.
- **Department names:** This entity type represents common medical department names, such as Respiratory Medicine, Ophthalmology, Urology, Endocrinology, etc.

The dataset contains five types of relationships: Disease symptom, Concurrent disease, Regular medication, Required inspection, and Co-department. There are 6011 Disease symptom relationships, 11,829 Concurrent disease relationships, 58,934 Regular medication relationships, 38,706 Required inspection relationships, and 8752 Co-department

relationships. This comprehensive dataset provides a rich and diverse knowledge base for modeling complex relationships among various medical entities.The following provides our brief description of each relation type:

- **Disease symptom:** This relationship type connects the "Disease" class entities to the "Symptom names" class entities, establishing a link between a disease name and its corresponding symptom. This allows a clear understanding of the associations between diseases and their related symptoms.
- **Concurrent disease:** This relationship type is employed to establish connections between specific diseases and their associated complications within the "Disease" class entities. By associating a disease with its corresponding complications, we can accurately represent the association between a disease and its complications.
- **Regular medication:** This relationship type connects the "Disease" class entity to the "Drug name" class entity, establishing a link between a specific disease and its corresponding therapeutic agent. This connection enables us to gain a clearer understanding of the association between a particular disease and the drugs used for its treatment.
- **Required inspection:** This relationship type connects the "Disease" class entity with the "Diagnostic Items" class entity, establishing a link between a specific disease and its corresponding required diagnostic items. This connection enables a clear understanding of the association between a disease and the necessary tests for its identification or evaluation.
- **Co-department:** This relationship type connects the "Disease" class entity with the "Department names" class entity, establishing a link between the disease name and its corresponding diagnostic department. This association elucidates the relationship between a specific disease and the medical department responsible for its diagnosis and treatment.

In addition to the "Med-Dis" dataset, we also employ the FB15K-237 and WN18RR datasets, which are widely used in the knowledge graph embedding research community. These datasets provide a common benchmark to evaluate the performance of our proposed model and compare it with existing methods.

Table 1 presents a summary of the statistical information for the "Med-Dis" dataset, highlighting the distribution of entities and relationships across different categories. Similarly, Table 2 provides a summary of the statistical information for the FB15K-237 and WN18RR datasets, showcasing their diversity and complexity.

Table 1. Summary of Med-Dis Dataset Statistics.

Entity Type	Quantity	Relation Type	Quantity
Disease	8342	Disease symptom	6011
Drug name	3729	Concurrent disease	11,829
Symptom name	6203	Regular medication	58,934
Diagnostic item	3201	Required inspection	38,706
Department name	52	Co-Department	8752
Total	**21,527**	**Total**	**124,232**

Table 2. Summary of FB15K-237 and WN18RR datasets.

Benchmarks	Entities	Relations
WN18RR	40,943	11
FB15k-237	14,541	237

By utilizing these datasets, we aim to demonstrate the effectiveness and robustness of our proposed model in handling a variety of knowledge graph structures and challenges, both within the medical domain and beyond.

4.3. Evaluation Indicators

To evaluate the performance of our proposed model, we employ two widely used evaluation metrics: Mean Rank (MR) and Hits@10. These metrics allow us to assess the model's effectiveness in ranking correct entities and its ability to retrieve relevant entities within the top-ranked predictions.

Mean Rank (MR) is the average rank of the correct entities in the ranked list of entities predicted by the model. A lower MR value indicates better performance, as it suggests that the correct entities are ranked closer to the top of the predictions. Mathematically, MR is defined as follows:

$$MR = \frac{1}{N}\sum_{i=1}^{N} rank_i \tag{37}$$

where N is the total number of test triples and $rank_i$ denotes the rank of the correct entity for the i-th triple.

Hits@10, on the other hand, measures the proportion of correct entities ranked within the top 10 predictions. A higher Hits@10 value indicates better performance, as it demonstrates the model's ability to retrieve relevant entities among the top-ranked predictions. Hits@10 is mathematically defined as:

$$Hit@10 = \frac{1}{N}\sum_{i=1}^{N}[rank_i \leq 10] \tag{38}$$

where $[\cdot]$ represents the indicator function, which takes a value of 1 if the condition inside the brackets is true and 0 otherwise.

By employing these evaluation metrics, we can effectively gauge the performance of our model in various knowledge graph settings and compare it with existing state-of-the-art methods.

4.4. Ablation Experiment

In order to evaluate the individual contributions of various components within our proposed model, we perform a series of ablation experiments. Specifically, we assess the impact of removing the residual blocks and replacing the AHTM module with a standard Transformer module on the model's overall performance. By comparing the results of our full model to those of the ablated versions, we can better understand the significance of each component. The results of the ablation experiments for the Med-Dis, FB15K-237, and WN18RR datasets are presented in Tables 3–5, respectively.

Table 3. Ablation experiments of Our Model on the Med-Dis dataset.

Method	MR	Hit@10
InputLayer+Transformer	2981	0.568
InputLayer+AHTM	2537	0.624
InputLayer+Transformer+ResNet	2893	0.586
InputLayer+AHTM+ResNet	2265	0.632

Table 4. Ablation experiments of Our Model on the FB15K-237 dataset.

Method	MR	Hit@10
InputLayer+Transformer	226	0.487
InputLayer+AHTM	193	0.526
InputLayer+Transformer+ResNet	215	0.501
InputLayer+AHTM+ResNet	173	0.582

Table 5. Ablation experiments of Our Model on the WN18RR dataset.

Method	MR	Hit@10
InputLayer+Transformer	3091	0.481
InputLayer+AHTM	2739	0.513
InputLayer+Transformer+ResNet	2998	0.496
InputLayer+AHTM+ResNet	2364	0.562

These ablation results shed light on the importance of incorporating both the AHTM module and residual blocks into our model. From the tables, it is evident that the full model with InputLayer+AHTM+ResNet consistently achieves the best performance across all three datasets, in terms of both MR and Hit@10. Comparatively, the performance of the model with only the InputLayer+Transformer is inferior, highlighting the efficacy of the AHTM module in enhancing the model's capability to capture complex medical knowledge graph relationships. Furthermore, the results indicate that the addition of residual blocks contributes to the model's performance improvements, as the models with residual blocks generally outperform their counterparts without them.

In addition, the ablation experiments demonstrate the effectiveness of integrating the AHTM module and residual blocks into our model, as they contribute to the superior performance observed across the Med-Dis, FB15K-237, and WN18RR datasets. These results further emphasize the importance of carefully designing and combining model components to effectively tackle the challenges posed by the medical knowledge graph completion task.

4.5. Comparison Experiment

In this study, we present the results of the comparison experiments conducted between our proposed model and several baseline models, including TransE, DistMult, RotatE, ConvE, InteractE, JointE, and ConvKB. We evaluate the performance of these models on three datasets: Med-Dis, FB15K-237, and WN18RR. Our model's performance is assessed using two key evaluation metrics: Mean Rank (MR) and Hits@10. A lower MR value and a higher Hits@10 value indicate better performance. The experimental results are presented in Tables 6–8.

Table 6. Comparison experiments of different algorithms on the FB15K-237 dataset.

Method	MR	Hit@10
TransE	347	0.465
DistMult	254	0.419
RotatE	177	0.533
InteractE	172	0.535
JointE	177	0.543
ConvE	244	0.501
ConvKB	257	0.517
Our Model	173	0.582

Table 7. Comparison experiments of different algorithms on the WN18RR dataset.

Method	MR	Hit@10
TransE	3385	0.501
DistMult	5110	0.491
RotatE	3340	0.571
InteractE	5202	0.528
JointE	4655	0.537
ConvE	5277	0.479
ConvKB	2554	0.525
Our Model	2364	0.562

Table 8. Comparison experiments of different algorithms on the Med-Dis dataset.

Method	MR	Hit@10
TransE	3079	0.529
DistMult	4897	0.534
RotatE	3184	0.615
InteractE	5013	0.573
JointE	4236	0.558
ConvE	5162	0.497
ConvKB	2541	0.561
Our Model	2265	0.632

The experimental results demonstrate that our proposed model consistently outperforms the baseline models across all datasets. Specifically, on the Med-Dis dataset, compared to the performance baseline approach, our model improves up to 56% in MR and 27% in Hits@10. On the FB15K-237 dataset, our model improves the MR by nearly 51% and the Hits@10 value by 39%. In the WN18RR dataset, the MR value increases by nearly 55% and the Hits@10 value by 18%. These results serve to validate the effectiveness of our model and highlight the improvements offered by the integration of the AHTM module, residual blocks, and model compression techniques.

The comparison experiments indicate the comprehensive performance advantage of our proposed model over the baseline models. Our model's superior performance can be attributed to the careful design and combination of its components, which enable it to capture complex relationships in medical knowledge graphs effectively.

5. Conclusions

In this paper, we have presented a novel knowledge graph embedding model based on the Adaptive Hierarchical Transformer with Memory (AHTM) architecture, specifically tailored for the medical domain. Our model effectively tackles the challenges presented by the complex and heterogeneous nature of medical knowledge graphs. We introduced the data input module, which leverages convolution and residual networks to generate embeddings for head entities, relations, and tail entities. The AHTM module integrates tree-based encoding, joint attention, Neural Turing Machines (NTM), and adaptive hierarchical transformer layers to effectively process the input representations.

The learned embeddings generated by the AHTM model encode the relationships and hierarchical structures within medical knowledge graphs. These embeddings can be interpreted as continuous representations of medical entities and relationships in a high-dimensional space. By analyzing the distance and patterns in this space, one can identify meaningful connections between entities and gain insights into the underlying structure of the medical domain. The embeddings can be utilized in various real-world medical applications, some potential applications include:

- **Auxiliary diagnosis of disease:** The embeddings can be used to develop diagnostic support systems that help healthcare professionals identify diseases based on patient symptoms, medical history, and other relevant factors.
- **Treatment Recommendation:** By examining the embeddings, treatment recommendations can be generated based on the relationships between diseases, drugs, and diagnostic items. This can assist healthcare professionals in selecting appropriate treatments.
- **Drug Repurposing:** The embeddings can be used to discover new therapeutic applications for existing drugs by identifying similarities between drug entities and their relationships with diseases. This can potentially expedite the drug development process and reduce costs.
- **Medical Knowledge Discovery:** The learned embeddings can facilitate the discovery of previously unknown relationships between medical entities. By analyzing the embeddings, researchers can identify potential correlations, causative factors, or patterns

that may warrant further investigation, ultimately contributing to the expansion of medical knowledge.

Moreover, we implemented a model compression technique using knowledge distillation and weight quantization methods to reduce the storage and computational requirements of our model while maintaining its performance. Our proposed model was evaluated on a self-constructed medical knowledge graph dataset, "Med-Dis", as well as the widely used FB15K-237 and WN18RR datasets. Experimental results demonstrated the superior performance of our model compared to several baseline methods, with substantial improvements in MR and Hits@10 values.

Despite the promising results, our model has some limitations. First, our model primarily focuses on capturing both intra- and inter-relationships among entities in the medical knowledge graph but does not explicitly consider temporal information that may be present in the data. This limitation may lead to an incomplete understanding of the dynamics underlying the relationships among entities, as they may evolve over time. Second, our model assumes a fixed structure of the medical knowledge graph, which may not hold true in real-world scenarios where new entities and relationships are constantly being discovered. This assumption might result in a less adaptive model when dealing with dynamic changes in medical knowledge. Finally, the model's interpretability could be further improved to facilitate a better understanding of the complex relationships captured by the model. Enhancing interpretability would enable users to better leverage the learned embeddings for practical medical applications and contribute to a deeper comprehension of the underlying medical phenomena.

In conclusion, our proposed model advances the state-of-the-art in medical knowledge graph embedding by effectively capturing complex relationships and incorporating the AHTM architecture and model compression techniques. For future work, we plan to address the following priorities:

- We plan to incorporate temporal information in the AHTM model by exploring approaches such as temporal features, attention mechanisms, recurrent neural networks, T-GCNs, and temporal edge prediction. Addressing these aspects will enable the model to capture complex temporal dynamics in medical knowledge graphs, enhancing its applicability to real-world medical applications like disease diagnosis, treatment planning, and drug discovery.
- We aim to address the challenge of the evolving structure of medical knowledge graphs, which occurs in real-world scenarios with the constant discovery of new entities and relationships. Our plan is to create a dynamic AHTM model that adapts to graph structure changes over time. This can be achieved using techniques such as online, incremental, or continual learning, allowing the model to update its embeddings and knowledge base in real-time with new information.
- We aim to enhance the AHTM model's interpretability, enabling users to better comprehend learned embeddings for real-world medical applications and deepen understanding of underlying medical phenomena. We plan to incorporate explainable AI techniques, such as attention mechanisms, feature visualization, and local interpretable model-agnostic explanations into our model. These techniques will provide insight into the complex relationships captured by the model and reveal significant contributing features.

Author Contributions: Conceptualization, X.L. and H.Y.; methodology, H.Y.; formal analysis, W.Z.; investigation, W.Z.; resources, H.Y.; data curation, C.Y.; writing—original draft preparation, H.Y.; writing—review and editing, W.Z.; supervision, X.L.; project administration, W.Z.; funding acquisition, X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Zhengzhou collaborative innovation major project 20XTZX06013 and the Key scientific research project of colleges and universities in Henan Province 22A520042.

Data Availability Statement: Not applicable.

Acknowledgments: We express our sincere appreciation to Nan Lin and Lipeng Xie for their constructive feedback and insightful discussions throughout the course of this work. With their contributions, we have achieved satisfactory results on the both datasets.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Yu, P.S. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 494–514. [CrossRef]
2. Bollacker, K.D.; Evans, C.; Paritosh, P.K.; Sturge, T.; Taylor, J. Freebase: A collaboratively created graph database for structuring human knowledge. In Proceedings of the SIGMOD Conference, Vancouver, BC, Canada, 10–12 June 2008.
3. Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P.N.; Hellmann, S.; Morsey, M.; van Kleef, P.; Auer, S.; et al. DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia. *Semant. Web* **2015**, *6*, 167–195. [CrossRef]
4. Wu, C.; Wu, F.; Qi, T.; Huang, Y. FeedRec: News Feed Recommendation with Various User Feedbacks. In Proceedings of the ACM Web Conference 2022, Lyon, France, 25–29 April 2022.
5. Li, X.; Zhang, K.; Li, G.; Zhu, B. A Chinese Knowledge Graph for Cardiovascular Disease. In Proceedings of the 9th International Conference on Communications, Signal Processing, and Systems, Quebec City, QC, Canada, 16–20 June 2020; Liang, Q., Wang, W., Liu, X., Na, Z., Li, X., Zhang, B., Eds.; Springer: Singapore, 2021; pp. 1816–1826.
6. Xu, K.; Wang, L.; Yu, M.; Feng, Y.; Song, Y.; Wang, Z.; Yu, D. Cross-lingual Knowledge Graph Alignment via Graph Matching Neural Network. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; Association for Computational Linguistics: Florence, Italy, 2019; pp. 3156–3161. [CrossRef]
7. Nunes, S.; Sousa, R.T.; Pesquita, C. Predicting Gene-Disease Associations with Knowledge Graph Embeddings over Multiple Ontologies. *arXiv* **2021**, arXiv:2105.04944.
8. Yang, B.; tau Yih, W.; He, X.; Gao, J.; Deng, L. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *arXiv* **2015**, arXiv:1412.6575.
9. Sun, Z.; Deng, Z.H.; Nie, J.Y.; Tang, J. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. *arXiv* **2019**, arXiv:1902.10197.
10. Dettmers, T.; Minervini, P.; Stenetorp, P.; Riedel, S. Convolutional 2D Knowledge Graph Embeddings. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
11. Vashishth, S.; Sanyal, S.; Nitin, V.; Agrawal, N.; Talukdar, P. InteractE: Improving Convolution-based Knowledge Graph Embeddings by Increasing Feature Interactions. *arXiv* **2020**, arXiv:1911.00219.
12. Zhou, Z.; Wang, C.; Feng, Y.; Chen, D. JointE: Jointly utilizing 1D and 2D convolution for knowledge graph embedding. *Knowl.-Based Syst.* **2022**, *240*, 108100. [CrossRef]
13. Nguyen, D.Q.; Nguyen, T.D.; Nguyen, D.Q.; Phung, D. 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), New Orleans, LA, USA, 1–6 June 2018; Association for Computational Linguistics: New Orleans, LA, USA, 2018. [CrossRef]
14. Toutanova, K.; Chen, D. Observed versus latent features for knowledge base and text inference. In Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality, Beijing, China, 26–31 July 2015; Association for Computational Linguistics: Beijing, China, 2015; pp. 57–66. [CrossRef]
15. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.
16. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.
17. Courbariaux, M.; Bengio, Y. BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or −1. *arXiv* **2016**, arXiv:1602.02830.
18. Choudhary, S.; Luthra, T.; Mittal, A.; Singh, R. A Survey of Knowledge Graph Embedding and Their Applications. *arXiv* **2021**, arXiv:2107.07842.
19. Wang, M.; Qiu, L.; Wang, X. A Survey on Knowledge Graph Embeddings for Link Prediction. *Symmetry* **2021**, *13*, 485. [CrossRef]
20. Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; Yakhnenko, O. Translating Embeddings for Modeling Multi-relational Data. In Proceedings of the NIPS 2013, Lake Tahoe, NV, USA, 5–10 December 2013.
21. Peng, H.; Wu, Y. A Dynamic Convolutional Network-Based Model for Knowledge Graph Completion. *Information* **2022**, *13*, 133. [CrossRef]
22. Sutton, R.; Pincock, D.; Baumgart, D.; Sadowski, D.; Fedorak, R.; Kroeker, K. An overview of clinical decision support systems: Benefits, risks, and strategies for success. *npj Digit. Med.* **2020**, *3*, 17. [CrossRef] [PubMed]
23. Jain, S.; Naicker, D.; Raj, R.; Patel, V.; Hu, Y.C.; Srinivasan, K.; Jen, C.P. Computational Intelligence in Cancer Diagnostics: A Contemporary Review of Smart Phone Apps, Current Problems, and Future Research Potentials. *Diagnostics* **2023**, *13*, 1563. [CrossRef] [PubMed]

24. Savva, K.; Zachariou, M.; Kynigopoulos, D.; Fella, E.; Vitali, M.I.; Kosofidou, X.; Spyrou, M.; Sargiannidou, I.; Panayiotou, E.; Dietis, N.; et al. Preliminary In Vitro and In Vivo Insights of In Silico Candidate Repurposed Drugs for Alzheimer's Disease. *Life* **2023**, *13*, 1095. [CrossRef]
25. Du, N.; Chen, K.; Kannan, A.; Tran, L.; Chen, Y.; Shafran, I. Extracting Symptoms and their Status from Clinical Conversations. *arXiv* **2019**, arXiv:1906.02239.
26. Wijesiriwardene, T.; Nguyen, V.; Bajaj, G.; Yip, H.Y.; Javangula, V.; Mao, Y.; Fung, K.W.; Parthasarathy, S.; Sheth, A.P.; Bodenreider, O. UBERT: A Novel Language Model for Synonymy Prediction at Scale in the UMLS Metathesaurus. *arXiv* **2022**, arXiv:2204.12716.
27. Wishart, D.S.; Feunang, Y.D.; Guo, A.C.; Lo, E.J.; Marcu, A.; Grant, J.R.; Sajed, T.; Johnson, D.; Li, C.; Sayeeda, Z.; et al. DrugBank 5.0: A major update to the DrugBank database for 2018. *Nucleic Acids Res.* **2017**, *46*, D1074–D1082.
28. Himmelstein, D.S.; Lizee, A.; Hessler, C.; Brueggeman, L.; Chen, S.L.; Hadley, D.; Green, A.; Khankhanian, P.; Baranzini, S.E. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLife* **2017**, *6*, e26726. [CrossRef]
29. Bao, H.; Dong, L.; Wang, W.; Yang, N.; Wei, F. s2s-ft: Fine-Tuning Pretrained Transformer Encoders for Sequence-to-Sequence Learning. *arXiv* **2021**, arXiv:2110.13640.
30. Mavi, V.; Jangra, A.; Jatowt, A. A Survey on Multi-hop Question Answering and Generation. *arXiv* **2022**, arXiv:2204.09140.
31. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. *arXiv* **2018**, arXiv:1710.10903.
32. Xin, X.; Wumaier, A.; Kadeer, Z.; He, J. SSEMGAT: Syntactic and Semantic Enhanced Multi-Layer Graph Attention Network for Aspect-Level Sentiment Analysis. *Appl. Sci.* **2023**, *13*, 5085. [CrossRef]
33. Bi, Z.; Cheng, S.; Chen, J.; Liang, X.; Zhang, N.; Chen, Q.; Xiong, F.; Guo, W.; Chen, H. Relphormer: Relational Graph Transformer for Knowledge Graph Representations. *arXiv* **2023**, arXiv:2205.10852.
34. Yun, S.; Jeong, M.; Kim, R.; Kang, J.; Kim, H.J. Graph Transformer Networks. *arXiv* **2020**, arXiv:1911.06455.
35. Schuler, A.H.; Praschl, C.; Pointner, A. Analysing and Transforming Graph Structures: The Graph Transformation Framework. *Software* **2023**, *2*, 218–233. [CrossRef]
36. Pezzat-Morales, M.; Perez-Meana, H.; Nakashika, T. Fast Jukebox: Accelerating Music Generation with Knowledge Distillation. *Appl. Sci.* **2023**, *13*, 5630. [CrossRef]
37. Shen, W.; Wang, W.; Zhu, J.; Zhou, H.; Wang, S. Pruning- and Quantization-Based Compression Algorithm for Number of Mixed Signals Identification Network. *Electronics* **2023**, *12*, 1694. [CrossRef]
38. Courbariaux, M.; Bengio, Y.; David, J.P. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. *arXiv* **2016**, arXiv:1511.00363.
39. Alemdar, H.; Leroy, V.; Prost-Boucle, A.; Pétrot, F. Ternary Neural Networks for Resource-Efficient AI Applications. *arXiv* **2017**, arXiv:1609.00222.
40. Dai, S.; Venkatesan, R.; Ren, H.; Zimmer, B.; Dally, W.J.; Khailany, B. VS-Quant: Per-vector Scaled Quantization for Accurate Low-Precision Neural Network Inference. *arXiv* **2021**, arXiv:2102.04503.
41. Alemdar, H.; Leroy, V.; Prost-Boucle, A.; Pétrot, F. European Conference on Computer Vision. *arXiv* **2016**, arXiv:1603.05027.