




Article

# A Two-Stage Framework for Directed Hypergraph Link Prediction

Guanchen Xiao <sup>1</sup>, Jinzhi Liao <sup>1</sup>, Zhen Tan <sup>1</sup>, Xiaonan Zhang <sup>2</sup> and Xiang Zhao <sup>1,\*</sup>

<sup>1</sup> Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China; xiaoguanchen20@nudt.edu.cn (G.X.); liaojinzhi12@nudt.edu.cn (J.L.); tanzhen08a@nudt.edu.cn (Z.T.)  
<sup>2</sup> Harbin Flight Academy, Harbin 150000, China; zhangxiaonan12@nudt.edu.cn  
\* Correspondence: xiangzhao@nudt.edu.cn

**Abstract:** Hypergraphs, as a special type of graph, can be leveraged to better model relationships among multiple entities. In this article, we focus on the task of hyperlink prediction in directed hypergraphs, which finds a wide spectrum of applications in knowledge graphs, chem-informatics, bio-informatics, etc. Existing methods handling the task overlook the order constraints of the hyperlink's direction and fail to exploit features of all entities covered by a hyperlink. To make up for the deficiency, we present a performant pipelined model, i.e., a two-stage framework for directed hyperlink prediction method (TF-DHP), which equally considers the entity's contribution to the form of hyperlinks, and emphasizes not only the fixed order between two parts but also the randomness inside each part. The TF-DHP incorporates two tailored modules: a Tucker decomposition-based module for hyperlink prediction, and a BiLSTM-based module for direction inference. Extensive experiments on benchmarks—WikiPeople, JF17K, and ReVerb15K—demonstrate the effectiveness and universality of our TF-DHP model, leading to state-of-the-art performance.

**Keywords:** hyperlink prediction; hypergraph; Tucker decomposition

**MSC:** 68T07



**Citation:** Xiao, G.; Liao, J.; Tan, Z.; Zhang, X.; Zhao, X. A Two-Stage Framework for Directed Hypergraph Link Prediction. *Mathematics* **2022**, *10*, 2372. <https://doi.org/10.3390/math10142372>

Academic Editor: Aydin Azizi

Received: 2 June 2022

Accepted: 3 July 2022

Published: 6 July 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Link prediction benefits in amplifying the relations in graph-structured data [1], arousing interest from both academia and industries. Existing research mainly focuses on simple graphs where a link (also known as a relation) associates with two entities (also known as an entity), while some real-world relations consist of more than two entities, such as chemical reactions [2], co-authorship relations [3], and social networks [4], etc. As shown in Figure 1, the “Located In” relation contains NYC, New York City, The Big Apple, USA, and The United States, as follows:

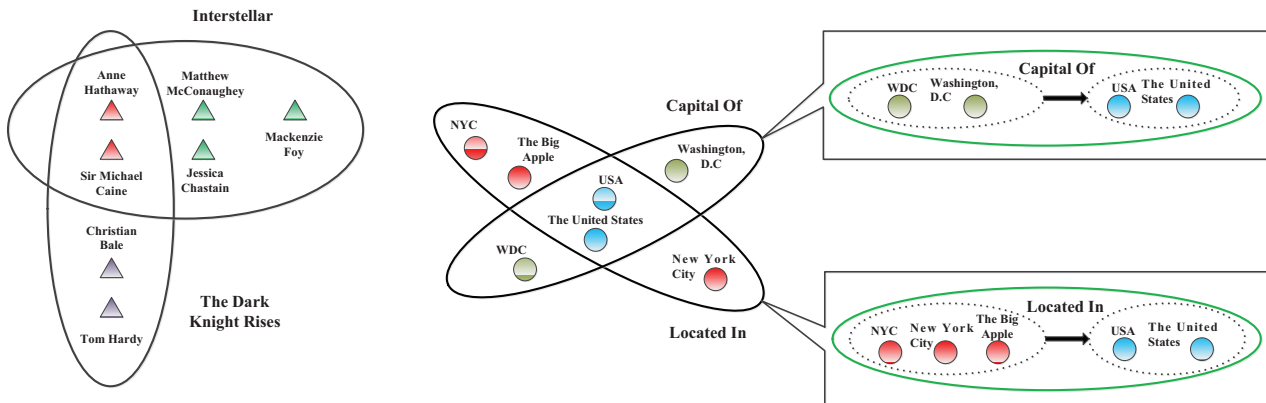
NYC, New York City, The Big Apple  $\xrightarrow{\text{Located In}}$  USA, The United States.

Thus, a hyperlink is coined to model such relations, and the graph comprised of hyperlinks is defined as a *hypergraph* [5].

As the relations among entities are sophisticated, the construct of a hypergraph is time-consuming and hence expensive, making its incompleteness more severe than a simple graph. To mitigate the problem, a *hyperlink prediction task* is introduced to facilitate the research [6]. Similar to the goal of link prediction in simple graphs, the task tries to complete the missing hyperlinks in a given hypergraph.

**Example 1.** Consider the bottom ellipse in green in Figure 1, given several entities, e.g., NYC, New York City, The Big Apple, USA, The United States; the target of the hyperlink prediction is to determine whether there is a hyperlink and what it is (i.e., “Located In”) once existing.

Furthermore, the directivity of hyperlinks also matters in some practical applications. Thus, the machine should also acquire the ability to predict the direction of the hyperlink to form the final answer, i.e., NYC, New York City, The Big Apple  $\xrightarrow{\text{Located In}}$  USA, The United States.



**Figure 1.** Sketch of two types of hypergraphs. The diagram on the left represents an undirected hypergraph while the diagram on the right stands for a directed hypergraph. One ellipse denotes a hyperlink. entities in the same ellipse share the same hyperlink. Arrow denotes the direction of the hyperlink.

To approach this task, current studies mainly fall into two categories: (1) Translation-based models try to generalize the translation constraint in simple graphs to hypergraphs, e.g., m-TransH [7], RAE [8], and NHP [9]. m-TransH directly extends TransH [10] for binary relations to the n-ary case, and RAE further integrates m-TransH with multi-layer perceptron (MLP) by considering the relatedness of entities. Since they use the sum after the projection as the scoring function, when some entities in a hyperlink change, it may not be obvious in the scoring function. (2) Neural-network-based models exploit structural information of hypergraphs, e.g., NaLP [11], HGNN [12], and HyperGCN [13]. These methods design some graph neural networks (GNNs) to absorb neighbouring features to improve entities’ representations. As GNNs usually incorporate a large number of parameters, the sufficient learning process relies on the amount of training samples.

Albeit attracting attention, hyperlink prediction is still notoriously challenging, since existing studies neglect the cores of the task. First, sometimes, the accurate record of facts in a hypergraph necessitates the direction of hyperlinks. For a directed hyperlink, the entities can be divided into two parts—head and tail—based on the hyperlink’s direction. This mandates that the *order* of the two parts matters; in contrast, the specific order in each part is insignificant. As shown in Figure 1, without the arrow pointing, we cannot figure out how these entities construct the relation “Located In”. In addition, NYC, New York City, and The Big Apple (also known as the head) should be in front of USA and The United States (also known as tail), but the order inside the head or tail does not affect the determination. Nevertheless, existing methods mainly focus on undirected hyperlinks. The only method, namely, NHP, tries to average the entity embeddings generated by GCN [14] to calculate a score for inferring the hyperlink direction, which is too rudimentary to embody the direction’s features. Second, as a hyperlink contains more than two entities, each entity contributes to the final existence prediction. In this light, a good representation model needs to consider the representation of all the individual entities involved in a hyperlink when making a determination. However, the current treatment of embedding tends to apply a simple sum or average strategy. This might be insensitive to the number of entities in a hyperlink since an entity with effusive containment could overwhelm other entities’ expressions. Last but not least, as it is sometimes complicated for even a human being to annotate hyperlinks, there is a lack of training data, which can be currently insufficient to train a large number of learnable parameters well.

In order to address these challenges, we propose a simple yet effective model, which is a Two-stage Framework for Directed Hyperlink Prediction, namely, TF-DHP. The model is expected to equally consider the entity's contribution to the form of hyperlinks and emphasize not only the fixed order between two parts but also the randomness inside each part. It conceives a pipeline of two tailored modules: a Tucker decomposition-based module for hyperlink prediction and a BiLSTM-based module for direction inference.

For predicting the existence of hyperlinks, we exploit *Tucker decomposition* to model hyperlinks, which, to the best of our knowledge, has not been applied to hypergraphs except simple graphs [15]. In particular, instead of applying three-order Tucker decomposition over simple graphs, we employ high-order Tucker decomposition for hypergraphs. It produces a core tensor, which represents the degree of interaction between entities. Then, we devise a scoring function by the mode product of the tensor with each entity representation, which evaluates the existence of hyperlinks. We theoretically show that the score is invariant to the order of mode product with entities, though there is a direction of each hyperlink. In addition, it is noted that the tensors from Tucker decomposition are usually of very high order, which can bring about high computational complexity. To mitigate the issue, we further introduce Tensor Ring (TR) [16] decomposition to decompose higher-order tensors into mode products of several third-order tensors, which effectively reduces the computational cost.

For inferring directions, we first recall that example in Figure 1. Once USA and The United States are determined as the tail entities, the substances in the head entities are implied, and if there is a change in one of the tail entities, the head entities are going to be different. Thus, it is of importance for the model to pass the information between the two parts both forward and backward. This motivates us to design a model that works bidirectionally. In this connection, BiLSTM [17] is utilized to serve as the base model. In addition, the position of entities within the head (or tail) part is insignificant, and hence, it is necessary to train the model to attend only to the order of the two parts. For this characteristic, we keep the order of two parts but randomly shuffle entities within each part to enforce the model to be ignorant of entity positions within head (or tail) part, while being attentive to the order between the two parts. In this way, the data scale is increased as a by-product, alleviating the lack of data.

**Contribution.** In summary, we make the following contributions:

- For existence prediction, we propose, among the first, to generalize Tucker decomposition to a high dimension and introduce a tensor ring algorithm to reduce the model complexity. We theoretically prove that the mode product for scoring a hyperlink is invariant of the order of participating entities.
- For direction inference, we conceive a BiLSTM-based model that can take information into consideration both forward and backward with respect to a hyperlink. A data shuffling strategy is further incorporated to enforce the model to be ignorant of entity positions within the head (or tail) part while being attentive to the order between the two parts.
- The modules constitute a new model, namely, TF-DHP for predicting directed hyperlinks. Through the experiments on several real-world datasets, we confirm the superiority of TF-DHP over state-of-the-art models.

**Organization.** The rest of the article is structured as follows. Section 2 introduces related work and Section 3 provides a detailed account of TF-DHP. Section 4 reports the experimental setup and analyses the experimental results. Section 5 concludes the paper.

## 2. Related Work

In this section, we are going to review related work in link prediction on simple graphs, undirected hypergraphs and directed hypergraphs.

### 2.1. Link Prediction on Simple Graph

Most of the link prediction methods on simple graphs can be divided into three categories—linear mathematics models, non-linear convolutional models, and random walk models.

There were many linear mathematics ways of link prediction created in recent years such as RESCAL [18], DistMult [19], ComplEx [20], and Simple [21]. RESCAL, which is based on tensor factorization, performs collective learning via the latent components of the model and provides an efficient algorithm to compute the factorization. DisMult is a special case of RESCAL with a diagonal matrix per relation which reduces overfitting while ComplEx extends DisMult to the complex domain. Simple is based on Canonical Polyadic (CP) decomposition, in which subject and object entity embeddings for the same entity are independent. TuckER [15] is a straightforward but powerful model based on the Tucker decomposition; it considers the core tensor as the parameter tensor, and the scoring function is defined by taking the modular product between the entities embedding vectors, the relation embedding vector, and the core tensor. Because the information loss in the calculation process is greatly reduced by using high-order tensors to define parameters, TuckER is proved to be the best-performing linear mathematics model to handle the link prediction task on simple graph.

Typical works of non-linear convolutional models are ConvE [22] and HypER [23]. ConvE is a simple multi-layer convolutional architecture for link prediction and is defined by a single convolution layer, a projection layer to the embedding dimension, and an inner product layer. HypER's hypernetwork generates relation-specific filters, and thus extracts relation-specific features from the subject entity embedding. It necessitates no 2D reshaping and allows entity and relation to interact more completely, rather than only around the concatenation boundary.

LRW [24], MIRW [25], and MLRW [26] are random walk-based models for link prediction on complex networks, LRW is conducted using pure random walking and selects the destination entities based on a random manner. To help to improve the LRW, the concept of asymmetric mutual influence of entities is presented, and using this concept, the walker selects the next entity using its effect on the current entity and selects more efficient paths for the next step. Therefore, entities with a more significant structural similarity will obtain a higher score in the proposed algorithm MIRW. MLRW provides a framework to extend the local random walk method to multiplex networks so that we can take advantage of intra-layer and interlayer information presented in the network and increase the accuracy of link prediction properly.

### 2.2. Link Prediction on Undirected Hypergraph

The general work on the undirected hyperlink prediction can be divided into two species, i.e., translation-based models and neural network-based models.

The representative model of the translation-based approaches are m-TransH [7] and RAE [8]. m-TransH generalizes TransH [10] to the case of n-order relations, and it projects entities onto the relation-specific hyperplane and defines the scoring function as the weighted sum of projection results. RAE considers the possibility of common occurrence between entities in n-order relations, establishes the correlation model through MLP, and reflects it in scoring function. Since these models are extended from binary models, restrictions on the representation of relations are also carried to the representation of n-order relations.

NaLP [11], HyperGCN [13], and Hyper-SAGNN [27] are three neural network-based approaches. HGNN is a general hypergraph neural network framework based on hypergraph convolution operation, which can incorporate multi-modal data and complicated data correlations. HyperGCN proposes a new method of training a GCN on hypergraph using tools from the spectral theory of hypergraphs and applying the method to the problems of SSL (hypergraph-based semi-supervised learning) and combinatorial optimization on real-world hypergraphs. Hyper-SAGNN develops a new self-attention based graph neu-

ral network applicable to homogeneous and heterogeneous hypergraphs with variable hyperlink sizes.

### 2.3. Link Prediction on Directed Hypergraphs

The research of link prediction on directed hypergraphs is not very mature, and most methods prefer predicting the direction of the hyperlink after finishing predicting the entities contained in the hyperlink. The NHP [9] model sets up two scoring functions to predict hyperlinks and their directions based on the GCN template, and they divide a hyperlink into two sub-hyperlinks and use their embedding vectors to compute the scoring function for direction. However, as the embedding vectors of hyperlinks are from the average value of entity embedding vectors, information about entities and their positions is lost, which makes the performance of the model barely satisfactory.

## 3. Method

This section formalizes the task of the directed hypergraph link prediction and presents the proposed method, including the framework and module details. Definitions of notations used in the text are shown in the Table 1.

**Table 1.** Descriptions of notations used in the following parts.

Symbol	Definition
$\chi$	a kth-order tensor $\in R^{I_1 \times I_2 \times \dots \times I_{k-1}}$
$\omega$	a kth-order core tensor $\in R^{I_1 \times I_2 \times \dots \times I_{k-1}}$
$\omega_{j_1 j_2 \dots j_{k-1}}$	$(j_1, j_2, \dots, j_{k-1})$ -th element of $\omega$
$U^{(n)}$	n-mode factor matrix $\in R^{I_n \times J_n}$
$u_{j_n}^n$	$j_n$ -th column vector of $U^{(n)}$
$\phi(\cdot)$	scoring function of the existence of hyperlinks
$r$	relation embedding of hyperlink
$v_m$	embedding of entities
$\times_n$	tensor n-mode product
$Z_k(i_k)$	$i_k$ -th lateral slice matrix of TR origin tensor
$Trace(\cdot)$	matrix trace operator
$\oplus$	concatenating operation for hidden layers
$\circ$	vector outer product

### 3.1. Task Description

A directed hypergraph is an ordered pair  $H = (V, E)$ , where  $V = \{v_1, \dots, v_l\}$  denotes a set of entities and  $l$  is the number of entities.  $E$  comprises a set of directed hyperlinks, formally:

$$E = \{(h_1, t_1), (h_2, t_2), \dots, (h_m, t_m)\} \tag{1}$$

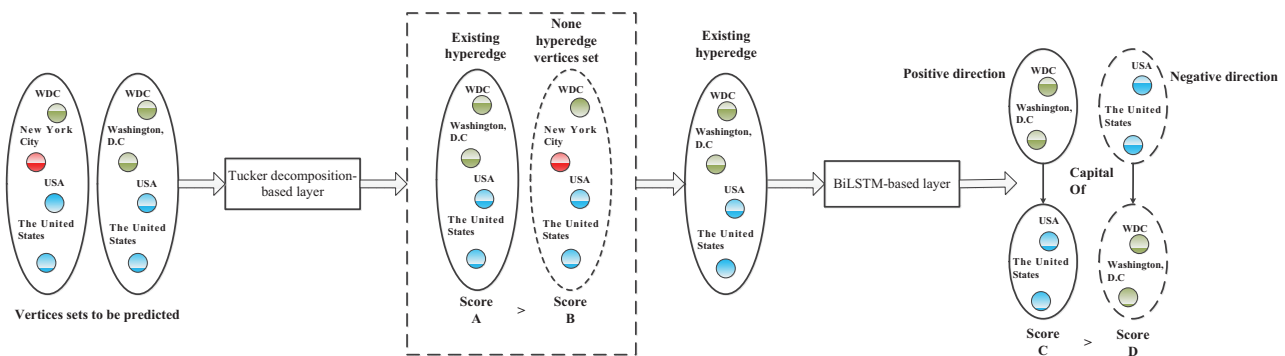
Each element in  $E$  can be divided into two components, where  $h$  (resp.  $t$ ) serves as the *head* (resp. *tail*), with the direction being from the head to the tail.

The directed hyperlink prediction aims to predict the missing hyperlinks, including the existence and associated direction, based on the relevance of the given entities. Take relation knowledge in Figure 1 as an instance. Entities in each relation build the  $V$ , and their corresponding relation forms the directed hyperlinks  $E$ . Every sample in the dataset will contain an uncertain number of substances. We have to determine whether they can support a relation knowledge and which component each entity belongs to.

### 3.2. Framework

TF-DHP consists of a Tucker decomposition-based hypergraph link prediction model and a BiLSTM-based direction prediction model to predict directed hyperlinks among entities sets in a directed hypergraph. It is then optimized by a ranking objective in which scores of existing hyperlinks are ranked higher than those of non-existing entity subsets and scores of positive directions are higher than those of negative directions. The framework is shown in Figure 2.

We generalize TuckerER [15] to the high dimension and regard it as a scoring function. We use the scoring function after obtaining the embedding vectors of every entity in an entity set to evaluate whether the hyperlink exists or not. If the hyperlink does exist, we divide the entities set into two groups based on the direction label of each entity and then use the BiLSTM model [17] to evaluate the direction between the groups which can be defined as the direction of the hyperlink. Meanwhile, we also randomly sort the entities in each group to increase training data according to the characteristic that the order of entities in each group does not influence the direction.



**Figure 2.** A sketch of TF-DHP directed hypergraph prediction model. The embedding of entity sets to be predicted are fed into the Tucker-decomposition-based layer to calculate the score. The target of model training is to make the score of existing hyperlinks larger than the score of entities set without hyperlinks. Then, the embeddings of entities in the existing hyperlink are sent to the BiLSTM layer to calculate the direction score. The target of model training is to make the score in the positive direction larger than the score in the negative direction.

### 3.3. Tucker Decomposition-Based Hyperlink Prediction Module

To predict hyperlinks of the entity set, we propose a Tucker decomposition-based scoring function and provide mathematical proof of its irrelevance with the order of inputs.

#### 3.3.1. Tucker Decomposition-Based Scoring Function

Tucker decomposition is a tensor decomposition algorithm that decomposes higher-order tensors into a core tensor and several factor matrices. The core tensor reflects the degree of interaction between different factor matrices. The formal expression is as follows:

$$\mathcal{X} = \omega \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_{k-1} U^{(k-1)} = \sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} \dots \sum_{j_{k-1}=1}^{J_{k-1}} \omega_{j_1 j_2 \dots j_{k-1}} u_{j_1}^{(1)} u_{j_2}^{(2)} \dots u_{j_{k-1}}^{(k-1)} \quad (2)$$

where  $\mathcal{X} \in R^{I_1 \times I_2 \times \dots \times I_{k-1}}$  denotes the original tensor,  $\omega \in R^{I_1 \times I_2 \times \dots \times I_{k-1}}$  denotes the core tensor and  $J_1 J_2 \dots J_{k-1}$  are much smaller than  $I_1 I_2 \dots I_{k-1}$ ,  $k$  denotes the order of  $\mathcal{X}$ ,  $U^{(1)}, \dots, U^{(k-1)}$  denotes the set of factor matrices, and the mathematical symbol  $\times_k$  denotes the tensor product along with the  $k$ th mode. The dimensions of the core tensor are smaller than those of the original tensor in each order, so the core tensor can be regarded as the dimensionality reduction in the original tensor.

Based on the Tucker decomposition of the representation tensor, we design the scoring function to score each hyperlink. Specifically, if a hyperlink contains  $m$  entities, we

first select the corresponding entity and relation embeddings. Then, a parameter tensor is designed as the core tensor containing learnable parameters shared by entities and relations [15]. Our goal is to optimize these parameters to fully exploit the relevance among entities and the associated relations based on their embeddings. The scoring function can be expressed as below:

$$\phi(r, v_1, v_2, \dots, v_m) = \omega \times_1 r \times_2 v_1 \times_3 \dots \times_{m+1} v_m, \tag{3}$$

where  $m$  changes with the number of entities contained in the hyperlink, and the order of the tensor  $\mathcal{Z}$  is equal to one plus the number of entities.  $r$  denotes the relation embedding of the hyperlink to be predicted, and  $v_1, v_2, \dots, v_m$  are the embeddings of entities contained by the hyperlink. Since the tensor product of a tensor with a vector will change the dimension of its corresponding order to 1, we can repeat the process  $m + 1$  times to acquire a real number. This real number is further regarded as the score of this hyperlink.

As every entity in the hyperlink and the relation embedding are computed simultaneously, Equation (3) reduces information loss. Nevertheless, the computational complexity becomes enormous with the increase in the number of entities because of the inner computation of the high-order tensor product. To address the issue, we use the TR [16] decomposition algorithm. It represents a high-order tensor by a sequence of third-order tensors multiplied circularly, mathematically:

$$T(i_1, i_2, \dots, i_n) = Trace\{Z_1(i_1)Z_2(i_2) \dots Z_n(i_n)\} = Trace\{\prod_{k=1}^d Z_k(i_k)\} \tag{4}$$

where  $T$  denotes the original tensor of size  $n_1 \times n_2 \times \dots \times n_d$ ,  $Z_k$  denotes a set of third-order tensors whose dimensions are  $r_k \times n_k \times r_{k+1}$ ,  $i_k$  denotes  $i_k$ -th layer matrix in the second-order of the tensor, and  $Tr$  denotes the trace of the product of matrices. The tensor ring decomposition makes the third dimension of the last decomposed tensor the same as the first dimension of the first decomposed tensor. The advantage is that when we make a circular shifting of the decomposed tensor, the results will not be changed because of the matrix trace operation. Tensor ring decomposition dramatically reduces the computational load of the model when the tensor order is large by decomposing higher-order tensors into products of third-order tensors.

The computational complexity grows sharply when the order of the core tensor grows, so we use the TR decomposition on the core tensor to decompose the high-order tensor into several three-order tensors multiplied circularly. Based on the definition of TR decomposition, every single parameter in the core tensor can be computed by the trace of the matrices product. It can be expressed in the tensor form [16], given by:

$$Z = \sum_{\alpha_1, \dots, \alpha_n=1}^{r_1, \dots, r_n} Z_1(\alpha_1, \alpha_2) \circ Z_2(\alpha_2, \alpha_3) \circ \dots \circ Z_d(\alpha_d, \alpha_1) \tag{5}$$

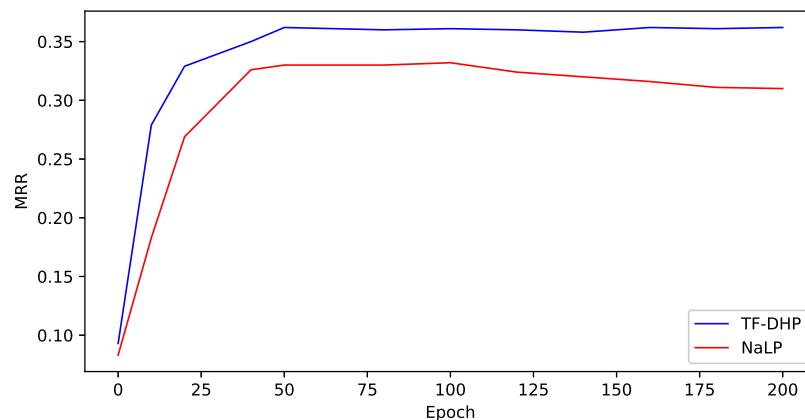
where  $Z_i(\alpha_k, \alpha_{k+1})$  denotes the vector corresponding to the index in the tensor and the symbol  $\circ$  denotes the outer product of vectors,  $r_1, \dots, r_n$  correspond to the dimension of the first and 3rd order of the tensor. We use the simplified form  $Z = Tr(Z_1, Z_2, \dots, Z_n)$  to represent the decomposition of the core tensor. Combining with Equation (3), we can rewrite the scoring function as:

$$\phi(r, v_1, v_2, \dots, v_n) = Trace(Z_1, Z_2, \dots, Z_n) \times_1 r \times_2 v_1 \times_3 \dots \times_{n+1} v_n \tag{6}$$

This scoring function not only considers all the entities and relation information contained in a hyperlink but also controls the model complexity within an acceptable range. As shown in Table 2, the scoring function above has fewer parameters than NaLP and is not easy to overfit in the datasets which are not large enough, concretely shown in Figure 3.

**Table 2.** Scoring functions of several models for undirected hypergraph link prediction tasks, with the significant terms of their model complexity.  $n_e$  and  $n_r$  are the number of entities and relations, while  $d_e$  and  $d_r$  are the dimensionalities of entity and relation embeddings respectively.  $n$  is the number of entities in a hyperlink and  $d_{max}$  is the maximum size of TR latent tensors.  $maxmin$  is the element-wise difference of maximum and the minimum values of the vectors.

Model	Scoring Function	Model Complexity
RAE	$\ \sum_{j=1}^n a_j(e_{i_j} - w_{ir}^T e_{i_j} w_{ir}) + r_{ir}\ _p$	$O(n_e d_e + n_r d_r)$
NaLP	$FCN_2(\min(FCN_1(\text{Conv}([W_r, [e_{i_1}; e_{i_2}; \dots; e_{i_n}] ]))))$	$O(n_e d_e + n_r d_r)$
NHP-U-mean	$\sigma(\frac{1}{ e } W \cdot \sum_{v \in e} h_v^{(e)} + b)$	$O(n_e d_e)$
NHP-U-maxmin	$\sigma(W \cdot \maxmin\{h_v^{(e)}\}_{v \in e} + b)$	$O(\sum_{e \in E} \frac{1}{2} \cdot  n_e  \cdot ( n_e  - 1))$
TF-DHP	$\text{Trace}(Z_1, Z_2, \dots, Z_n) \times_1 r \times_2 v_1 \times_3 \dots \times_{n+1} v_n$	$O(n_e d_e + n_r d_r + n d_{max}^3)$



**Figure 3.** MRR results on NaLP and TF-DHP with training epoch growing, evaluated on WikiPeople.

This model is based on the scoring function of Tucker decomposition, and because the model needs to determine the order of the core tensor, the model cannot process the hyperlinks with different number of nodes in one time. For datasets with such hyperlinks, we need to classify them before predicting, which increases the workload to a certain extent.

As the order of the core tensors increases, the number of third-order tensors required by TR decomposition increases accordingly, which will increase the amount of computation to a certain extent. The machine used in this paper can deal with the prediction task of hyperlinks with up to six nodes.

### 3.3.2. Proof of Sequence Independence

As illustrated above, the Tucker decomposition processes the inputs sequentially, while the order of entities contained in one hyperlink does not influence the determination, which requires the invariance property of our scoring function. We prove that the order of entities' and relations' embeddings in the tensor product makes no difference to the result. We first rewrite the scoring function in the tensor-wise form:

$$\phi(r, v_1, v_2, \dots, v_n) = \sum_{\alpha_1, \dots, \alpha_n=1}^{r_1, \dots, r_n} Z_1(\alpha_1, \alpha_2) \circ \dots \circ Z_d(\alpha_d, \alpha_1) \times_1 r \times_2 v_1 \times_3 \dots \times_{n+1} v_n \quad (7)$$

In the mentioned TR decomposition, the matrix trace operation and the same dimensions of the input and output ensure the invariance of circular shifting. When it comes to the hypergraph, the dimensions of entities and relations are set to a fixed value, which makes the invariance not only in circular shifting but also in order changing between every single entity. It means the change in the order of the product does not change the result. So,



we just need to prove that the order of the tensor product in the Tucker decomposition has no effect on the result. The element-wise form of the tensor product is as follows:

$$\begin{aligned} \chi_{i_1 i_2 \dots i_n} &= (\omega \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_n U^{(n)})_{i_1 i_2 \dots i_n} \\ &= \sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} \dots \sum_{j_n=1}^{J_n} \omega_{j_1 j_2 \dots j_n} u_{i_1 j_1}^{(1)} u_{i_2 j_2}^{(2)} \dots u_{i_n j_n}^{(n)} \end{aligned} \tag{8}$$

On the right-hand side of the equation, if we regard the indices  $j_1, \dots, j_n$  as a set of integer-independent variables and their variation range is from 1 to  $J_1, \dots, J_n$ ,  $(u_{i_1 j_1}^{(1)}, u_{i_2 j_2}^{(2)}, \dots, u_{i_n j_n}^{(n)})$  can be regarded as the functions of these independent variables, the meaning of the function value is the value of the element at the corresponding position in the entity embedding vector indexed by the independent variable. We use  $f_1(j_1), f_2(j_2), \dots, f_n(j_n)$  (in Equation (9)) to represent the functions. The expression  $\omega_{j_1 j_2 \dots j_n}$  can be regarded as a multivariate function whose form is  $g(j_1, j_2, \dots, j_n)$ , and the value of the function means the parameter on the corresponding position of the core tensor.

Then, we find that if we make the independent variables take the value of all real numbers from 1 to  $J_n$  instead of being integers, we can transform Equation (8) into a multiple definite integral:

$$\iiint \dots \int_D g(j_1, j_2, \dots, j_n) f_1(j_1) f_2(j_2) \dots f_n(j_n) dj_1 dj_2 \dots dj_n \tag{9}$$

The integral domain  $D$  of this multiple integrals is an n-order tensor that has the same size as the core tensor. Changing the order of independent variables in  $g(j_1, j_2, \dots, j_n)$  does not change the corresponding parameter; thus, the order of  $j_1, \dots, j_n$  has no influence of the function  $g(j_1, j_2, \dots, j_n) f_1(j_1) f_2(j_2) \dots f_n(j_n)$ .

Since the functions  $f_1(j_1), \dots, f_n(j_n)$  are all unary function, the integral can be rewritten as:

$$\iiint \dots \int_D g(j_1, j_2, \dots, j_n) dj_1 dj_2 \dots dj_n \int_1^{J_1} f_1(j_1) dj_1 \int_1^{J_2} f_2(j_2) dj_2 \dots \int_1^{J_n} f_n(j_n) dj_n \tag{10}$$

For the multiple definite integrals  $\iiint \dots \int_D g(j_1, j_2, \dots, j_n) dj_1 dj_2 \dots dj_n$ , the limit of integration for each order are finite constants, and the order of  $j_1, \dots, j_n$  makes no difference to the function, so changing the order of integration does not change the value of the definite integral. Therefore, the whole integral has the invariance property. Because Equation (8) is a special case of Equation (9), the scoring function is proven to have the invariance property.

### 3.4. BiLSTM-Based Direction Prediction Module

In the directed hyperlink prediction problem, the embedding of each entity further determines the existence of a hyperlink and its direction. However, different from the existence prediction, the direction of a hyperlink emphasizes the order of entities. For example, in the related knowledge “WDC, Washington D.C  $\xrightarrow{\text{Capital Of}}$  USA, The United States”, the direction comes from WDC and Washington D.C (also known as head entities) to USA and The United States (also known as tail entities). Once a substance is placed in the wrong component, the reaction might not even exist. In addition, the interaction between two components, e.g., conservation of materials, indicates that the model cannot individually determine the components. Therefore, we apply BiLSTM in our module to encode all entities sequentially to achieve the information passing both forward and backward.

As shown in the Figure 4 The BiLSTM consists of several LSTM hidden layers. These hidden layers are divided into two groups that meet end-to-end in opposite directions. The entities’ embeddings in the hyperlink are calculated in the hidden layer of the corresponding position one by one. Meanwhile, the state of the previous hidden layer is calculated in the next hidden layer together with the embedding of the entities fed into the corresponding layer. After all hidden layers have been calculated, embedding containing all sequential

information is generated. The same process occurs in the backward hidden layer group, which means we can obtain two embeddings of the hyperlink. We concatenate them into one vector and then send it to a Softmax layer to obtain the direction score. The specific expression of the process is as follows:

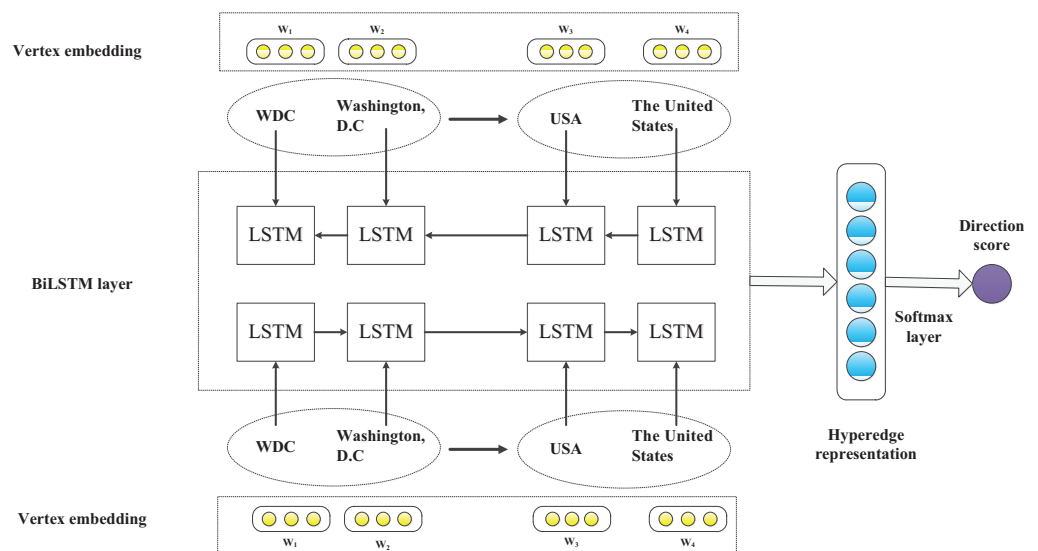
$$\overleftarrow{h}_t = \overleftarrow{LSTM}(\overleftarrow{h}_{t+1}, w_t) \tag{11}$$

$$\overrightarrow{h}_t = \overrightarrow{LSTM}(\overrightarrow{h}_{t-1}, w_t) \tag{12}$$

$$h_t = \overrightarrow{h}_t \oplus \overleftarrow{h}_t \tag{13}$$

$$p = \text{Softmax}(h_t) \tag{14}$$

where  $h_t$  denotes the concatenated embedding of the sequential representation,  $\overrightarrow{h}_t$  and  $\overleftarrow{h}_t$  are calculated by two hidden layers in opposite directions,  $w_t$  denotes the embedding for the  $t$ th entity, and the symbol  $\oplus$  means the concatenating operation.



**Figure 4.** A sketch of the BiLSTM-based hyperlink direction prediction model, the entities in the directed hyperlink are divided into *head* and *tail* parts according to the label and are input into the BiLSTM layer in a specific order. The hyperlink representation is obtained by splicing the representation vectors obtained from each direction of the BiLSTM layer. Finally, the direction score is obtained through a Softmax layer.

As the inner order of entities in one component does not change the elements, it also has no effect on the direction, e.g., “WDC, Washington D.C  $\xrightarrow{\text{Capital Of}}$  USA, The United States” and “Washington D.C, WDC  $\xrightarrow{\text{Capital Of}}$  The United States, USA” are the same relation knowledge. However, they might be regarded as two different instances when fed into BiLSTM concentrating only on the specific sequence. In other words, if “WDC, Washington D.C  $\xrightarrow{\text{Capital Of}}$  USA, The United States” is annotated as the positive instance, BiLSTM cannot naturally and directly determine the correctness of “Washington D.C, WDC  $\xrightarrow{\text{Capital Of}}$  The United States, USA” without other guidance. Therefore, we enlighten BiLSTM to focus on the order of two components and ignore the order of entities in the same component through a data shuffling strategy. Specifically, we maintain the order of two components and randomly shuffle the entities in the same component. The number of generated instances relies on how many entities every component owns. For “WDC, Washington D.C  $\xrightarrow{\text{Capital Of}}$  USA, The United States”, there will be  $2 \times 2 = 4$  different sequences. We then

give all generated instances a correct label to enforce BiLSTM to exploit features of the direction. The strategy can enlarge the data scale without introducing external manual efforts, which also contributes to tackling the low-data regime problem.

### 3.5. Training

TF-DHP is a pipeline model, which means that we predict the hyperlink's existence in the first stage and judge the direction of the hyperlink in the second stage. If we use the data of undirected hypergraphs to train the first stage of the model separately, we can obtain a model that can perform link prediction of undirected hypergraphs. If the whole model is trained on the data of the directed hypergraph, the trained model can have the ability to predict directed hyperlinks.

The TF-DHP is trained in two stages, which keeps the same pace with the framework. The training goal of the first stage is to provide the existing hyperlink with a higher score while decreasing the score of entities that cannot comprise a hyperlink. With the initial embeddings of entities and their labels as input, we use the Tucker decomposition-based scoring function to obtain two kinds of the score, and a binary cross-entropy loss function is designed to maximize their gap.

After the first stage of the model is trained, we acquire the updated core tensor and embeddings and use these embeddings to initialize the second stage of the model. Two kinds of scores are calculated in the BiLSTM. One is the score of the correct direction, and the other is the score of the wrong direction. The specific expression of the loss function is as follows:

$$L = f_{mean}(\log(1 + e^{f_{mean}(\sigma(\phi_{d_n})) - \sigma(\phi_{d_p})})) \quad (15)$$

where  $f_{mean}$  denotes an average function,  $\sigma$  denotes the sigmoid function,  $\phi_{d_n}$  denotes the score of each negative hyperlink, and  $\phi_{d_p}$  denotes the score of each positive hyperlink. Finally, the BiLSTM-based model updates the model parameters and embeddings of entities and relations based on the loss gradients.

## 4. Experiment

This section reports the experiments.

### 4.1. Experimental Setup

We detail the adopted datasets, evaluation metrics, parameters, and baselines.

#### 4.1.1. Datasets

We use two public relational datasets in our experiment for undirected hypergraph link prediction and one open KB canonicalized dataset for directed hypergraph link prediction. We brief these datasets below.

- **WikiPeople** [11]: WikiPeople is a public  $n$ -ary relational dataset concerning entities of type *human* extracted from Wikidata. WikiPeople is an incomplete hypergraph with many hyperlinks missing [11]. In WikiPeople, each set of entities has one kind of relationship. We use this dataset to train the undirected hyperlink prediction model.
- **JF17K** [8]: JF17K is a public  $n$ -ary relational dataset that has high-quality facts. It is filtered from Freebase while having multi-fold relational structures preserved. The same as WikiPeople, each set of entities has one kind of relationship, and we use this dataset to train the undirected hyperlink prediction model.
- **ReVerb15K** [9,28]: ReVerb45K is an open KB canonicalization dataset [28], and it is constructed by intersecting information from ReVerb Open KB [29], Freebase entity linking information from [30], and Clueweb09 corpus [31]. In triples of the original dataset, there may be different subjects or objects having the same meaning. Based on the Freebase entity linking information, we cluster the synonyms of the subjects or objects in one set, and use each cluster to represent the new subject or object. In this way, a canonicalized directed hypergraph dataset is obtained. Since it contains about 15 K

entities, we call it ReVerb15K. The treated subject entities represent head hyperlinks, and the treated object entities represent the corresponding tails; the direction is from head to tail.

The specific size of datasets are shown in the Table 3.

**Table 3.** Statistics of the hypergraph datasets used in the experiments.

Datasets	WikiPeople	JF17K	ReVerb15K
Direction	undirected	undirected	directed
Number of Entities	12,270	11,541	14,798
Number of Relations	66	104	382

#### 4.1.2. Metrics And Parameters

We test the effectiveness of the model in two parts. One is the Tucker-decomposition-based model for predicting the undirected hyperlinks, the other is the whole framework for predicting the directed hyperlinks. The total hyperlinks in datasets are divided into three parts: 20% for training, 10% for validation, and 70% for testing. We evaluate the link prediction performance via two standard metrics: MRR and Hits@k (k is top ranking). MRR is the mean of the inverse of rankings over all testing facts, while Hits@k measures the proportion of top k rankings. The aim of the training is to achieve high MRR and Hits@k.

The reported results are given for the best set of hyper-parameters evaluated on the validation set for each model, after grid search on the following values: embedding size  $\in \{15, 20, 25, 30, 35\}$ , learning rate  $\in \{1, 0.6, 0.06, 0.006\}$ , and TR-ranks  $\in \{5, 10, 20, 30, 40\}$ , with TR-ranks the size of the tensor decomposed by TR decomposition.

#### 4.1.3. Baselines

We compare TF-DHP with the following  $n$ -ary hyperlink prediction baselines:

- RAE [8]: RAE is a translational distance model which considers the possibility of common occurrence between entities in  $n$ -order relations, establishes a correlation model through MLP, and reflects it in the scoring function.
- NaLP [11]: NaLP is a neural network model that achieves the state-of-the-art  $n$ -ary hypergraph link prediction performance.
- HGNN [12]: This is a general hypergraph neural network framework for data representation learning based on hypergraph convolution operation, which can incorporate multi-modal data and complicated data correlations. We use  $maxmin_+$  as a scoring layer and a direction scoring layer [9] for directed hyperlink prediction with HGNN.
- HyperGCN [13]: This is a new method of training a GCN on hypergraph using tools from spectral theory of hypergraphs. Since it is not directly proposed for hyperlink prediction, we use the same scoring layers as used on HGNN.
- NHP-U-mean and NHP-U-maxmin [9]: These two methods are both based on the GCN layer. NHP-U-mean uses  $mean$  as the scoring layer while NHP-U-maxmin uses  $maxmin_+$  as the scoring layer to predict hyperlinks. These two methods are proposed for undirected hyperlink prediction.
- NHP-D-mean and NHP-D-maxmin [9]: These two methods use a direction scoring layer on NHP-U-mean and NHP-U-maxmin to predict directed hyperlinks.

#### 4.2. Experiment on Undirected Hypergraphs

Tables 4 and 5 show the undirected hyperlink prediction results on two datasets. The highest scores are set in bold. As shown in the tables, we can find out that our proposed TF-DHP can achieve optimal results under various measurement standards, consistently. For both datasets, graph neural networks NHP combining the  $mean$  or  $maxmin$  scoring functions cannot have comparable performances in link prediction problems. For example,

on WikiPeople, compared with our proposed model, TF-DHP, the MRR of the first four methods is only about a third, and Hits@10 is about a half. The large improvement of TF-DHP can strongly confirm that scoring functions such as *mean* or *maxmin* largely ignore the influence of the representation of each entity in the hyperlink on the predicted results, which also reflects the advantage of Tucker-decomposition-based model taking every entity embedding into the computation.

**Table 4.** Undirected hyperlink prediction results on WikiPeople dataset.

Model	MRR	Hits@10	Hits@3	Hits@1
HGNN	0.132	0.285	0.152	0.117
HyperGCN	0.137	0.289	0.158	0.115
NHP-U-mean	0.122	0.283	0.147	0.119
NHP-U-maxmin	0.143	0.302	0.144	0.139
RAE	0.153	0.273	0.152	0.146
NaLP	0.332	0.537	0.403	0.334
TF-DHP	<b>0.362</b>	<b>0.574</b>	<b>0.440</b>	<b>0.368</b>

**Table 5.** Undirected hyperlink prediction results on JF15K dataset.

Model	MRR	Hits@10	Hits@3	Hits@1
HGNN	0.649	0.722	0.640	0.526
HyperGCN	0.654	0.743	0.652	0.538
NHP-U-mean	0.632	0.710	0.639	0.509
NHP-U-maxmin	0.686	0.783	0.670	0.573
RAE	0.707	0.837	0.751	0.629
NaLP	0.714	0.805	0.737	0.673
TF-DHP	<b>0.751</b>	<b>0.873</b>	<b>0.786</b>	<b>0.686</b>

As for the translational distance model RAE, although RAE achieves slightly better results than the four methods, its results are still unsatisfying. On WikiPeople, TF-DHP improves MRR by 0.21 and Hits@1 by 0.15, which is a considerable improvement. The main reason for the unsatisfying performance of RAE is the restriction on relations of the translational distance model. Such restriction does not exist in the Tucker-decomposition-based model. Tucker decomposition can accurately represent any ground truth over a set of entities and relations by its full expressiveness [15].

The performance of NaLP is much better than the aforementioned methods due to the enormous amount of model parameters. It uses a neural network to greatly reduce the restriction on relations existing in the translational distance model. However, a large number of parameters makes it easy to over-fit, especially when training datasets are not big enough. According to the network structure and scoring function of NaLP, the model complexity of NaLP is  $O(n_e d_e + n n_r d_r)$ , with  $n_e$  and  $d_e$  representing the number and dimension of entities, respectively.  $n$  is the number of entities in one relation.  $n_r$  and  $d_r$  stand for the number and the dimension of relations, respectively. However, the model complexity of the first stage of TF-DHP is only  $O(n_e d_e + n_r d_r + n d_{max}^3)$ , where  $d_{max}$  is the maximum dimension of the third-order tensors in TR decomposition. Since the number of relations is much larger than the dimension of the decomposed tensor in hypergraphs, the model complexity of NaLP is apparently larger than TF-DHP. As shown in Figure 3, with the training epoch growing, NaLP requires more training epochs than TF-DHP to achieve the optimal result. Moreover, because too many NaLP parameters lead to an over-fitting issue, the results decrease when the epoch is larger than 100. However, due to relatively few parameters, the results of TF-DHP are relatively stable after reaching the optimal result during training.

#### 4.3. Experiment on Directed Hypergraphs

Table 6 shows the results of several directed hyperlink prediction models. The highest scores are set in bold. To the best of our knowledge, there are few models dealing with the hyperlink prediction problem in directed hypergraphs. As shown in Table 6, TF-DHP obtains considerable improvement compared with other methods. For example, for the best baseline NHP-D-maxmin, TF-DHP improves MRR by 0.056 and Hits@10 by 0.026.

**Table 6.** Directed hyperlink prediction results of Reverb15K dataset.

Model	MRR	Hits@10	Hits@3	Hits@1
HGNN	0.276	0.422	0.336	0.226
HyperGCN	0.316	0.443	0.347	0.238
NHP-D-mean	0.288	0.435	0.352	0.219
NHP-D-maxmin	0.442	0.560	0.438	0.348
TF-DHP	<b>0.498</b>	<b>0.586</b>	<b>0.474</b>	<b>0.353</b>

We believe that there are two main reasons for the better prediction performance of TF-DHP on directed hypergraphs. First, when testing the directed hypergraph prediction model, we put the weighted average of scores computed in two stages of the model as the final score, which means we regard an entity set as positive only if there exists a directed hyperlink among the entities set with the direction also being correct. So, the accuracy of the first stage of the model will inevitably affect the performance of the whole model. Second, the NHP-D-maxmin and other methods in Table 6 use the average value of entities' embedding vectors to represent the embedding of the hyperlink and consider the product of embedding vectors of *head* and *tail* parts of the hyperlink as a scoring function. As mentioned above, these methods ignore the influence of each entity embedding on the direction of the hyperlink and the relationship between an entity and its adjacent entities. The improvement of experimental results proves that considering the representation information of each entity separately and the information of the adjacent entities (from forward to backward) can improve the accuracy of directed hypergraph prediction.

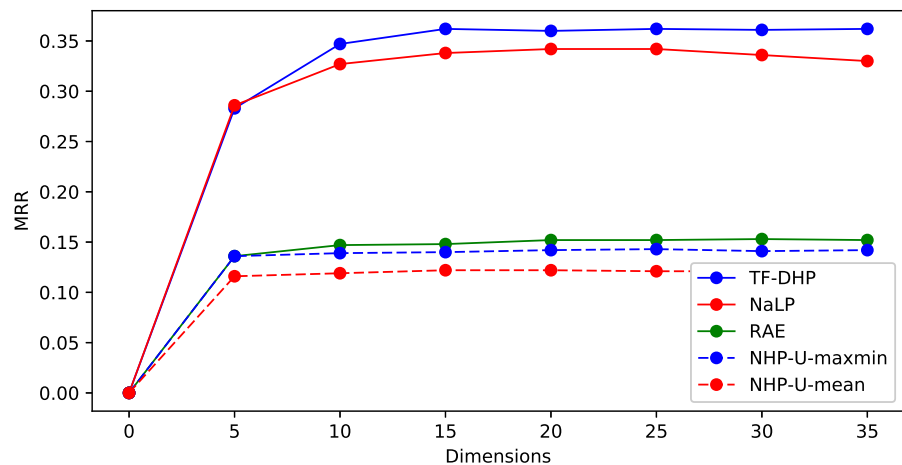
#### 4.4. Parameter Analysis

Embedding size is a significant factor in hyperlink prediction models, determining the performance of the model to a large extent. Hence, we will analyze the results obtained by the model in different embedding sizes to investigate its impact.

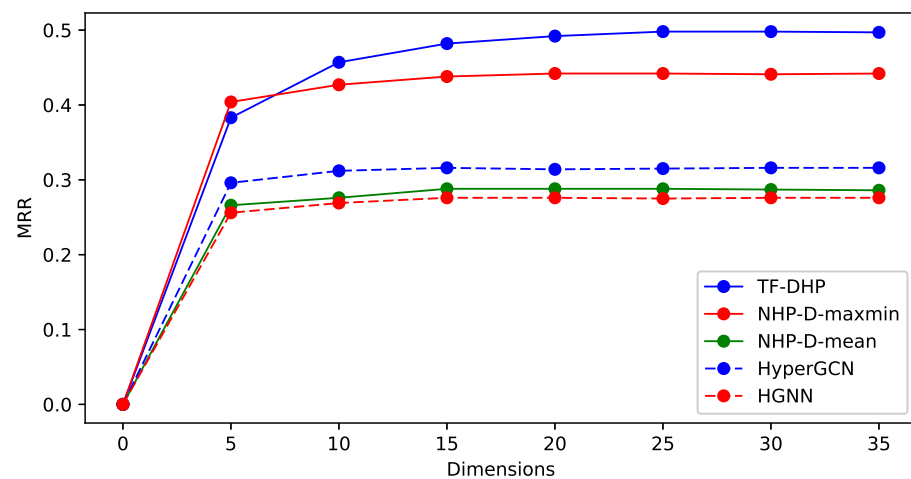
First, according to Figure 5a, TF-DHP outperforms other methods on each embedding size. The MRR of TF-DHP increases sharply with the early stage of increasing the embedding size and becomes smooth after the embedding size increases to 15. The MRR of NaLP is almost identical to TF-DHP's from the start; however, due to a large number of parameters, it cannot remain smooth like TF-DHP when the embedding size increases. After the embedding size increases to a certain extent, NaLP's MRR will decrease. For other methods, the change in embedding size has less influence on the experimental results due to their smaller number of parameters.

Figure 5b shows the impacts of embedding size on directed hyperlink prediction. The same as undirected hyperlink prediction, TF-DHP always outperforms other methods. As BiLSTM is added, the optimal embedding size of the model increases to 25, after which the increase in MRR becomes smooth. As for other methods, the addition of the direction scoring function also increases the optimal number of parameters and shares the similar tendency as TF-DHP.

It proves the stability of TF-DHP on the choice of the dimension size. In addition the reasonable amount of parameters of TF-DHP allows it to be more stable, as other models' performances may decrease with the increasing dimensions, suffering from the over-fitting issue.



(a)



(b)

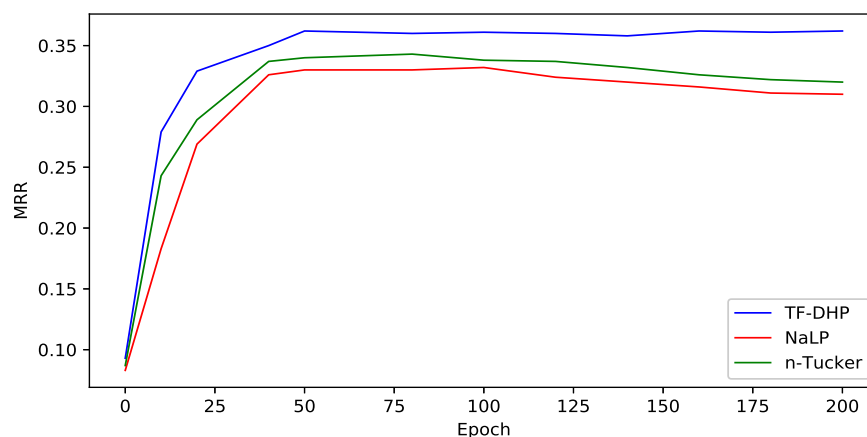
**Figure 5.** MRR over different embedding sizes of undirected hyperlink prediction models and directed hyperlink prediction models, evaluated on WikiPeople and Reverb15k: (a) undirected hyperlink prediction models; (b) directed hyperlink prediction models.

#### 4.5. Approximate Training Time Comparison

On the two undirected datasets WikiPeople and JF15K, TF-DHP takes around 45 min of training time, while NaLP and RAE take around 3 h and 1 h, respectively. On the directed dataset Reverb15K, TF-DHP takes around 1 h of training time, while NHP-D-maxmin and NHP-D-mean take around 15 min each due to their oversimplified scoring function. All were run on a GeForce GTX 1080 super GPU machine.

#### 4.6. Ablation Study

Since experiments on the directed hypergraph dataset have proved the effectiveness of the BiLSTM model, we designed an ablation study to prove the influence of TR decomposition in Tucker decomposition. We designed a variant on WikiPeople of TF-DHP which does not use TR decomposition on Tucker decomposition, and we call it *n-Tucker*. As shown in Figure 6, without TR decomposition, the computational complexity of the model greatly increases, which will result in an over-fitting issue. Similar but better than NaLP, *n-Tucker* reaches the optimal value of MRR and then gradually decreases due to the over-fitting issue. This kind of experiment not only proves the superiority of the Tucker decomposition-based model but also proves the necessity of the TR decomposition.



**Figure 6.** MRR under different training epochs of undirected hyperlink prediction models. Evaluated on WikiPeople.

## 5. Conclusions and Future Work

In this paper, we introduce TF-DHP, a novel model for hyperlink prediction for both undirected and directed hypergraphs. We use a tensor-decomposition-based method to handle the undirected part and add a BiLSTM model to predict the direction of the hyperlink. Our model TF-DHP is a pipelined model, which is flexible to deal with not only directed hypergraphs but also undirected hypergraphs. The experimental results verify the advantages of TF-DHP in both settings across multiple datasets.

In the future, we plan to further look into heterogeneous hypergraphs where there are multiple types of high-order relations, such as inclusion relations and produce relations, and to see how directed hypergraphs can be used on reaction prediction in chemical or biological domains.

**Author Contributions:** Conceptualization, G.X. and J.L.; methodology, G.X.; software, G.X. and J.L.; validation, G.X., J.L. and X.Z. (Xiang Zhao); formal analysis, G.X.; investigation, G.X., J.L., Z.T. and X.Z. (Xiaonan Zhang); data curation, G.X.; writing—original draft preparation, G.X.; writing—review and editing, G.X., J.L. and X.Z. (Xiang Zhao); visualization, G.X. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was partially supported by NSFC under grants Nos. 61872446, 61902417, and The Science and Technology Innovation Program of Hunan Province under grant No. 2020RC4046.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

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

## References

- Zhang, M.; Chen, Y. Link Prediction Based on Graph Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018 (NeurIPS 2018), Montréal, QC, Canada, 3–8 December 2018; pp. 5171–5181.
- Ning, X.; Shen, L.; Li, L. Predicting High-Order Directional Drug-Drug Interaction Relations. In Proceedings of the 2017 IEEE International Conference on Healthcare Informatics, ICHI 2017, Park City, UT, USA, 23–26 August 2017; IEEE Computer Society: Washington, DC, USA, 2017; pp. 556–561.
- Jin, T.; Wu, Q.; Ou, X.; Yu, J. Community detection and co-author recommendation in co-author networks. *Int. J. Mach. Learn. Cybern.* **2021**, *12*, 597–609. [[CrossRef](#)]
- Zhang, Z.; Liu, C. Hypergraph model of social tagging networks. *arXiv* **2010**, arXiv:1003.1931.
- Bollobás, B.; Daykin, D.E.; Erdős, P. Sets of Independent edges of a hypergraph. *Q. J. Math.* **1976**, *27*, 25–32. [[CrossRef](#)]
- Li, D.; Xu, Z.; Li, S.; Sun, X. Link prediction in social networks based on hypergraph. In Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, Brazil, 13–17 May 2013.



7. Wen, J.; Li, J.; Mao, Y.; Chen, S.; Zhang, R. On the Representation and Embedding of Knowledge Bases beyond Binary Relations. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016; Kambhampati, S., Ed.; IJCAI/AAAI Press: Palo Alto, CA, USA, 2016; pp. 1300–1307.
8. Zhang, R.; Li, J.; Mei, J.; Mao, Y. Scalable Instance Reconstruction in Knowledge Bases via Relatedness Affiliated Embedding. In Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, 23–27 April 2018; Champin, P., Gandon, F., Lalmas, M., Ipeirotis, P.G., Eds.; ACM: New York, NY, USA, 2018; pp. 1185–1194.
9. Yadati, N.; Nitin, V.; Nimishakavi, M.; Yadav, P.; Louis, A.; Talukdar, P.P. NHP: Neural Hypergraph Link Prediction. In Proceedings of the CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, 19–23 October 2020; ACM: New York, NY, USA, 2020; pp. 1705–1714.
10. Wang, Z.; Zhang, J.; Feng, J.; Chen, Z. Knowledge Graph Embedding by Translating on Hyperplanes. In Proceedings of the AAAI, Quebec City, QC, Canada, 27–31 July 2014.
11. Guan, S.; Jin, X.; Wang, Y.; Cheng, X. Link Prediction on N-ary Relational Data. In Proceedings of the The World Wide Web Conference (WWW 2019), San Francisco, CA, USA, 13–17 May 2019; Liu, L., White, R.W., Mantrach, A., Silvestri, F., McAuley, J.J., Baeza-Yates, R., Zia, L., Eds.; ACM: New York, NY, USA, 2019; pp. 583–593.
12. Feng, Y.; You, H.; Zhang, Z.; Ji, R.; Gao, Y. Hypergraph Neural Networks. In Proceedings of the the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019), The Thirty-First Innovative Applications of Artificial Intelligence Conference (IAAI 2019), The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI 2019), Honolulu, HI, USA, 27 January–1 February 2019; AAAI Press: Palo Alto, CA, USA, 2019; pp. 3558–3565.
13. Yadati, N.; Nimishakavi, M.; Yadav, P.; Nitin, V.; Louis, A.; Talukdar, P.P. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019; pp. 1509–1520.
14. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations (ICLR 2017), Toulon, France, 24–26 April 2017.
15. Balazevic, I.; Allen, C.; Hospedales, T.M. Tucker: Tensor Factorization for Knowledge Graph Completion. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019), Hong Kong, China, 3–7 November 2019; Inui, K., Jiang, J., Ng, V., Wan, X., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 5184–5193.
16. Zhao, Q.; Zhou, G.; Xie, S.; Zhang, L.; Cichocki, A. Tensor Ring Decomposition. *arXiv* **2016**, arXiv:1606.05535.
17. Tamburini, F. A BiLSTM-CRF PoS-tagger for Italian tweets using morphological information. In Proceedings of the Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016), Napoli, Italy, 5–7 December 2016; Volume 1749.
18. Nickel, M.; Tresp, V.; Krieger, H. A Three-Way Model for Collective Learning on Multi-Relational Data. In Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, WA, USA, 28 June–2 July 2011; Getoor, L., Scheffer, T., Eds.; Omnipress: Madison, WI, USA, 2011; pp. 809–816.
19. Yang, B.; Yih, W.; He, X.; Gao, J.; Deng, L. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015.
20. Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; Bouchard, G. Complex Embeddings for Simple Link Prediction. In Proceedings of the 33rd International Conference on Machine Learning (ICML 2016), New York, NY, USA, 19–24 June 2016; Volume 48, pp. 2071–2080.
21. Kazemi, S.M.; Poole, D. Simple Embedding for Link Prediction in Knowledge Graphs. In Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018 (NeurIPS 2018), Montréal, QC, Canada, 3–8 December 2018; pp. 4289–4300.
22. Dettmers, T.; Minervini, P.; Stenetorp, P.; Riedel, S. Convolutional 2D Knowledge Graph Embeddings. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, LA, USA, 2–7 February 2018; McIlraith, S.A., Weinberger, K.Q., Eds.; AAAI Press: Palo Alto, CA, USA, 2018; pp. 1811–1818.
23. Balazevic, I.; Allen, C.; Hospedales, T.M. Hypernetwork Knowledge Graph Embeddings. In Proceedings of the Artificial Neural Networks and Machine Learning—ICANN 2019—28th International Conference on Artificial Neural Networks, Munich, Germany, 17–19 September 2019; Proceedings—Workshop and Special Sessions; Lecture Notes in Computer Science; Tetko, I.V., Kurková, V., Karpov, P., Theis, F.J., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11731, pp. 553–565.
24. Liu, W.; Lu, L. Link prediction based on local random walk. *EPL* **2010**, *89*, 58007. [[CrossRef](#)]
25. Berahmand, K.; Nasiri, E.; Forouzandeh, S.; Li, Y. A Preference Random Walk Algorithm for Link Prediction through Mutual Influence Nodes in Complex Networks. *arXiv* **2021**, arXiv:2105.09494.
26. Nasiri, E.S.; Berahmand, K.; Li, Y. A new link prediction in multiplex networks using topologically biased random walks. *Chaos Solitons Fractals* **2021**, *151*, 111230. [[CrossRef](#)]
27. Zhang, R.; Zou, Y.; Ma, J. Hyper-SAGNN: A self-attention based graph neural network for hypergraphs. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.

28. Vashishth, S.; Jain, P.; Talukdar, P.P. CESI: Canonicalizing Open Knowledge Bases using Embeddings and Side Information. In Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, 23–27 April 2018; Champin, P., Gandon, F., Lalmas, M., Ipeirotis, P.G., Eds.; ACM: New York, NY, USA, 2018; pp. 1317–1327.
29. Fader, A.; Soderland, S.; Etzioni, O. Identifying Relations for Open Information Extraction. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, Edinburgh, UK, 27–31 July 2011; A meeting of SIGDAT, a Special Interest Group of the ACL; ACL: Stroudsburg, PA, USA, 2011; pp. 1535–1545.
30. Gabrilovich, E.; Ringgaard, M.; Subramanya, A. FACC1: Freebase Annotation of ClueWeb Corpora, Version 1 (Release date 2013-06-26, Format Version 1, Correction Level 0). 2013.
31. Callan, J.; Hoy, M.; Yoo, C.; Zhao, L. Clueweb09 Data Set 2009.