



**EVALUATION OF GRAPH EMBEDDING BASED REASONING
OVER KNOWLEDGE BASES**



BETÜL BAYRAK

AUGUST 2020

**EVALUATION OF GRAPH EMBEDDING BASED REASONING
OVER KNOWLEDGE BASES**

**A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF
NATURAL AND APPLIED SCIENCES**

**OF
ÇANKAYA UNIVERSITY**

BY

BETÜL BAYRAK

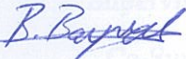
**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING**

AUGUST 2020

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this study has been obtained and bestowed in conformance with educational rules and ethics provided in educational medium. I can also state that, all materials and outcomes used in this thesis are referenced and cited in accordance with the educational rules.

Name, Last Name: Betül BAYRAK

Signature: 

Date: 28.08.2020

ABSTRACT

EVALUATION OF GRAPH EMBEDDING BASED REASONING OVER KNOWLEDGE BASES

BAYRAK, Betül

M.Sc., Computer Engineering Department

Supervisor: Asst. Prof Dr. Roya CHOUPANI

Co-Supervisor: Prof Dr. Erdoğan DOĞDU

AUGUST 2020, 47 pages.

Knowledge graphs (KG) include large amounts of structured data in many different domains. Knowledge or information is captured by entities and relationships among them in KG .

One of the open problems in the knowledge graphs area is “link prediction”, that is predicting new relationships or links among entities, given the existing entities and links in KG . A recent approach in graph-based learning problems is “graph embedding”, in which graphs are represented as low-dimensional vectors. It is easier to make link predictions using these vector representations using this method. We also use graph embedding for graph representations. A sub-problem of link prediction in KG is link prediction in the presence of literal values, and specifically numeric values, on the receiving end of links. This creates a difficult situation as the numeric literal values take arbitrary values. There are several studies in this area, but they are all complex approaches.

In this study, we propose a novel approach for link prediction in the presence of numerical values. We cluster the numerical values in graphs to enhance the prediction rates. We evaluated our method on Freebase knowledge graph, which includes entities, relations, and numeric literals. Test results show that a considerable increase in link prediction rate can be achieved in comparison to the other work.

Keywords: Link Prediction, Knowledge Graph Completion, Knowledge Graph Embedding, LP with Numeric Literals.



ÖZ

BİLGİ TABANLARI ÜZERİNDE BİLGİ ÇİZGESİ GÖMME MUHAKEMESİ.

BAYRAK, Betül

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Danışman: Dr. Öğr. Üyesi Roya CHOUPANI

Ortak Danışman: Prof Dr. Erdoğan DOĞDU

AĞUSTOS 2020, 47 sayfa.

Bilgi çizgeleri, bir çok alanda, büyük miktarda yapılı veri içerir. Bilgi çizgelerinde bilgiler, varlıklar ve varlıklar arasındaki ilişkiler aracılığıyla tutulur.

Bilgi çizgeleri alanında geliştirilmeye açık problemlerden biri de “bağlantı tahmini”dir. Bağlantı tahmini bilgi çizgelerinde varlıklar arasındaki var olmayan yeni ilişkileri tahmin etme işidir. Çizge tabanlı öğrenme problemlerindeki yeni bir yaklaşım “çizge gömme”dir. Çizge gömme, çizgelerin düşük boyutlu vektörler olarak temsil edilmesidir. Bu sayede, bu vektör gösterimleri kullanarak bağlantı tahminleri yapmak daha kolaydır. Bilgi çizgelerindeki bağlantı tahmininin bir alt problemi, bağlantıların alıcı ucundaki sabit değerlerin ve özellikle sayısal değerlerin varlığında bağlantı tahminidir. Bu, rastgele değerler alan sayısal değişmez değerler nedeniyle daha zor bir sorundur. Bu alanda birkaç çalışma var, ancak hepsi karmaşık yaklaşımlardır.

Bu çalışmada, sayısal değerlerin varlığında bağlantı tahmini için yeni bir yaklaşım öneriyoruz. Tahmin doğruluk oranlarını artırmak için sayısal değerleri kümelendiriy-

oruz. Önerdiğimiz yöntemi, varlıkları, ilişkileri ve sayısal sabitleri içeren FreeBase bilgi çizgeleri üzerinde değerlendirdik. Test sonuçları, diğer çalışmalara kıyasla bağlantı tahmin oranında önemli bir artış sağlanabileceğini göstermektedir.

Anahtar Kelimeler: Bağlantı Tahmini, Bilgi Çizgesi Tamamlama, Bilgi Çizgesi Sayısallaştırma, Nümerik Değerlerle Link Tahmini.



ACKNOWLEDGEMENT

Firstly, I would like to express my sincere appreciation and gratitude to Asst. Prof. Dr. Roya CHOUPANI and Prof. Dr. Erdoğan DOĞDU as my supervisor and co-supervisors respectively, for their continuous guidance and counseling, supervision, suggestions, and immense knowledge throughout the process of this masters study. Without your precious supports, it would not be possible to conduct this research and make this thesis reach its conclusion.

My sincere acknowledgment also goes to my thesis committee members Assoc. Prof. Dr. Ibrahim Alper DOGRU and Asst. Prof. Dr. Abdul Kadir GORUR for their motivation as well as guidance towards the end of writing this thesis. Your insightful, motivational comments encouraged me throughout this process.

To my friends and colleagues, thank you for listening, offering me advice, and supporting me through this entire process.

Finally, I must express my profound, sincere, and gratitude to my family, especially my parents and siblings, for their unparalleled love, encouragement, prayers, help, and continuous support both financially and emotionally throughout my years of study and the process of this thesis. This attainment would not have been possible without them. I proudly dedicate this accomplishment to them.

TABLE OF CONTENTS

STATEMENT OF NON-PLAGIARISM	iii
ABSTRACT	iv
ÖZ	vi
ACKNOWLEDGEMENT	viii
TABLE OF CONTENTS	x
LIST OF FIGURES	xi
LIST OF TABLES	xii
LIST OF ABBREVIATIONS	xiii
1 INTRODUCTION	1
1.1 Background	1
1.1.1 Knowledge Graphs	1
1.1.2 Applications of Knowledge Graph Usage	4
1.1.3 Knowledge Graph Embedding	5
1.1.4 Link Prediction on Knowledge Graphs	6
1.2 Problem Statement	7
1.3 Thesis Contribution	10
1.4 Thesis Structure	10
2 LITERATURE	12
2.1 Knowledge Graph Link Prediction	12
2.2 Knowledge Graph Link Prediction Using Literal Values	17
3 LINK PREDICTION WITH CLUSTERED LITERALS	19
3.1 Our Approach	19
3.2 Models Compared	23
3.2.1 TransE	24
3.2.2 DistMult	24
3.2.3 ComplEx	24
3.2.4 HolE	24

3.2.5	ConvE	25
3.2.6	ConvKB	25
4	EVALUATION	26
4.1	Experimental Setup	26
4.2	Dataset	27
4.3	Evaluation Metrics	27
4.3.1	Ranking	28
4.3.2	MRR	28
4.3.3	MR	28
4.3.4	HIT@N	28
4.4	Experimental Results	29
5	CONCLUSIONS AND FUTURE WORK	32
	REFERENCES	33
A	NOMENCLATURE	39
B	DATA SET DESCRIPTION	41
C	DETAILED EXPERIMENTS ON TRANSE MODEL	44

LIST OF FIGURES

1.1	Screen Shot of Google Search for 'Vietnam' Keyword. Right Side Structured Box.	2
1.2	Screen Shot of Google Search for 'Asia' Keyword. Right Side Structured Box.	2
1.3	A Sub-graph To Show Triple Structure on KGs.	3
1.4	A Sub-graph To Show Link Prediction on KGs.	6
1.5	A Sub-graph To Show Link Prediction Results on KGs.	7
1.6	A Sub-graph To Show Instances For URL And Numerical (l_u and l_n) Literals in KGs.	8
1.7	A Sub-graph To Show Numerical Literals in KGs.	9
3.1	The approach of the state-of-the-art studies on knowledge graph link prediction using literal values.	20
3.2	A Sub-graph To Show Perception Problems For KG LP with Literals.	21
3.3	Our approach: Knowledge Graph Link Prediction Using Clustered Literal Values.	22

LIST OF TABLES

1.1	Triple Instances From a Sub-graph To Show Triple Structure on KGs	3
3.1	Triple Instances To Show Perception Problems For KG LP with Literals	21
3.2	Compared Models' LP Results Comparison Table	23
4.1	Datasets Details	27
4.2	Result Table	31
A.1	Nomenclature Table.	39
B.1	(e, r, e) Triples From Dataset	41
B.2	(e, r, l_n) Triples From Dataset	42
B.3	(e, r, l_n) Triples From Dataset After Clustering Process	43
C.1	Detailed Experiment Result Table For TransE Model	47

LIST OF ABBREVIATIONS

- KG:** Knowledge Graph
- KGE:** Knowledge Graph Embedding
- LP:** Link Prediction
- MRR:** Mean Reciprocal Rank
- MR:** Mean Rank
- NN:** Neural Network
- CNN:** Convolutional Neural Network
- RNN:** Recurrent Neural Network
- FB:** FreeBase
- WN:** WordNet
- URI:** Uniform Resource Identifier
- BCE:** Binary Cross Entropy
- NLL:** Negative Log-Likelihood
- BIC:** Bayesian Information Criterion
- PoE:** Product of Expert
- Q&A:** Question Answer

CHAPTER 1

INTRODUCTION

1.1 Background

1.1.1 Knowledge Graphs

With the spread of digitalization, the amount of data produced in daily life and business life has increased considerably. As it has become challenging to manage this data, many different concepts have been added to our lives. One of these concepts is structured data usage.

There are many advantages to expressing data as structured. Unstructured data requires more processing and is also more challenging to handle than structured. For instance, searching on unstructured data is quite costly compared to structured data.

As an example, one of the most popular knowledge graphs usage areas in our lives is Google searches. As shown in *Figure – 1.1* and *Figure – 1.2*, when we searched a word like country, object, person, Google shows a box on the right side of the search result screen. In that box, there are lots of useful information about the searched item. For example, *Figure – 1.1* is the result of the 'Vietnam' keyword search information box. When we look at this information box, we can easily understand that the word represents a country in Asia and we can see its flag, location on the world map, short description, capital city, dialing code, population, currency, and more. We can learn the same information in the text sources, but as can be seen, when structured data are used, it is easier to perceive, understand, and evaluate.

Knowledge graphs generally consist of a large amount of structured data. With the increasing importance of structured data, knowledge graphs are getting more popular in recent years. Commonly, knowledge graphs incorporate real-world information as a structured format (*e.g.*, (*Hanoi, capital_city_of, Vietnam*))[37]. In the literature, this



Figure 1.1: Screen Shot of Google Search for 'Vietnam' Keyword. Right Side Structured Box.

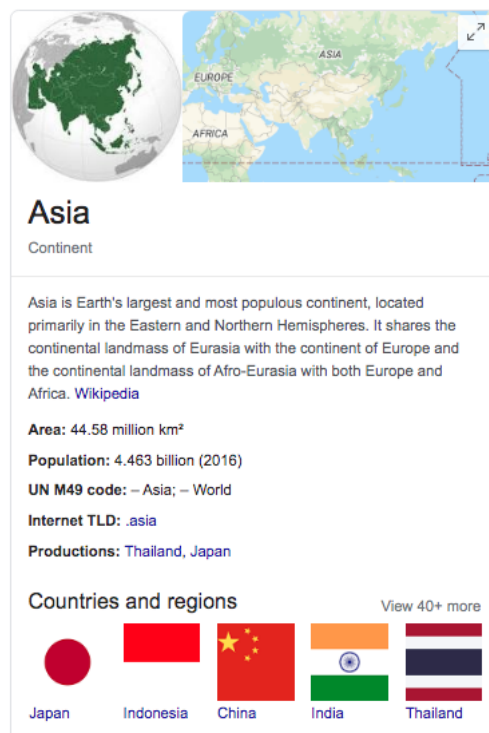


Figure 1.2: Screen Shot of Google Search for 'Asia' Keyword. Right Side Structured Box.

structured format is known as *triple*.

Knowledge graphs keep the information as triples. As we sampled above, these triples consist of 3 elements (*i.e.*, $(head, relation, tail)$). We will use h for the *head*, r for the *relation*, y for the *tail* from here on (*i.e.*, (h, r, t)).

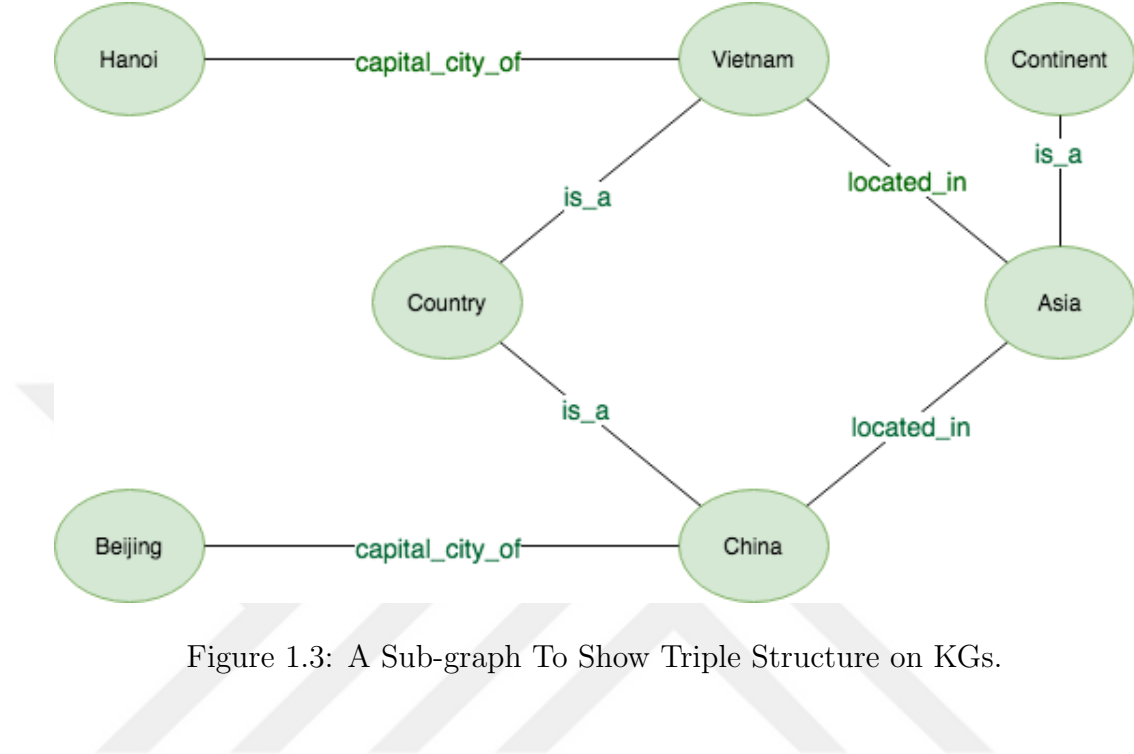


Figure 1.3: A Sub-graph To Show Triple Structure on KGs.

Table 1.1: Triple Instances From a Sub-graph To Show Triple Structure on KGs

	<i>Head</i>	<i>Relation</i>	<i>Tail</i>
1	Hanoi	capital_city_of	Vietnam
2	Vietnam	is_a	Country
3	Vietnam	located_in	Asia
4	Asia	is_a	Continent
5	Beijing	capital_city_of	China
6	China	is_a	Country
7	China	located_in	Asia

In *Figure – 1.3*, we created a sample part of the knowledge graph. In this sub-graph, several relations are illustrated. Certain information can be observed in this sub-graph, like Hanoi is the capital city of Vietnam, Asia is a continent, and China is located in Asia. We listed all triples from the sub-graph in *Figure – 1.3* and showed

in *Table – 1.1*. In the *Table – 1.1*, triples are showed in (h, r, t) format. As shown in the *Table – 1.1*, there are 7 triples in the sub-graph which is located in *Figure – 1.3*.

There are lots of knowledge graphs with different sizes, purposes, and features. Here are the some of the popular knowledge graphs: Freebase[3], DBpedia[19], YAGO[44], NELL[6], WordNet[25], Google Knowledge Graph[41], KnowledgeVault[9], Microsoft Satori[34], and Facebook Graph Search[43].

1.1.2 Applications of Knowledge Graph Usage

Because of increasing the importance of structured data, knowledge graphs are getting more popular in recent years. Thus, applications of the knowledge graphs are spread to different domains, and our interactions with knowledge graph applications in our professional and daily life increase. Some of the popular applications of knowledge graphs as follows:

- Knowledge graphs include a large amount of structured data. While knowledge graphs are large and structured, they are also quite incomplete. This means knowledge graphs are open to knowledge extraction. Link prediction means predicting relations between the existing head and tail elements (i.e., r in (h, r, t)).
- Entity classification: Entity classification is a type of giving category to an entity, e.g., Vietnam is a country. In short, entity classification is the prediction of entities' *'is_a'* relation.
- Question answering systems: Question answering systems got popular in recent years. These systems take questions as input, process as query, and predict relevant answers. Some of the most popular applications are Alexa, Siri, and Cortana.
- Item recommendation: Item recommendation systems which are based on knowledge graphs can recommend products for sale, movie, music, event, and rate.

Knowledge graph applications, which we mentioned above, are present in many aspects of our life. Some of the usage areas of this; web (especially search engine)[10], finance[21], medicine[23], social media[12], academy[52], and daily life[33].

1.1.3 Knowledge Graph Embedding

We made a mention of knowledge graphs and its applications, above. There is a vast number of different applications and domains for the applications of the aforementioned knowledge graphs, but as main and common points of these applications, we need to know how knowledge graph's triples are processed. As we mentioned before, knowledge graphs consist of (h, r, t) triples, but to apply something on these triples, like measuring distances between two triples, we need to process these triples. For the processing task, triples must be represented as vectors or combinations of vectors. In the literature, this vectorization process is known as embedding process. Moreover, the embedding process can be explained as projecting triples into a low-dimensional vector space.

Usually, embeddings are elements that are represented as vectors of numerical values. According to elements and types of the task between elements, embeddings are created automatically [37].

For example in this study, if the data is a knowledge graph, elements are entities and relations and embeddings are used to represent entities and relationships. In this way, the graph structure and the semantic are captured by the embeddings.

There are lots of vectorization (i.e., embedding) techniques proposed for different reasons for knowledge graph link prediction tasks. Embedding techniques have different usage areas, geometries, and structures.

Rossi et al., split KH LP techniques into three different branches as tensor decomposition models, geometric models, deep learning models[37].

- Tensor decomposition models: In this approach, triples are represented as matrices. ComplEx[47], DistMult[54], and TuckER[2] models are the most popular instances of this concept.
- Geometric models: In this approach, the assessments of relations between triples over geometry. RotatE[45], and TransE[4] models are the most popular instances of this concept.
- Deep learning based models: In this approach, deep neural networks with different

structures for different concepts are used. ConvE[8], and ConvKB[26] models are the most popular instances of this concept.

1.1.4 Link Prediction on Knowledge Graphs

As mentioned before, link prediction merely predicts unknown r elements with known h and t elements in (h, r, t) triple format. For example, it is possible to (jeremy, father_of, alice) and (donald, brother_of, jeremy) triples to exist in the knowledge graph and (donald, uncle_of, alice) triple to not exist, although it is true. Moreover, it is the most common application area of knowledge graphs, because of its incompleteness.

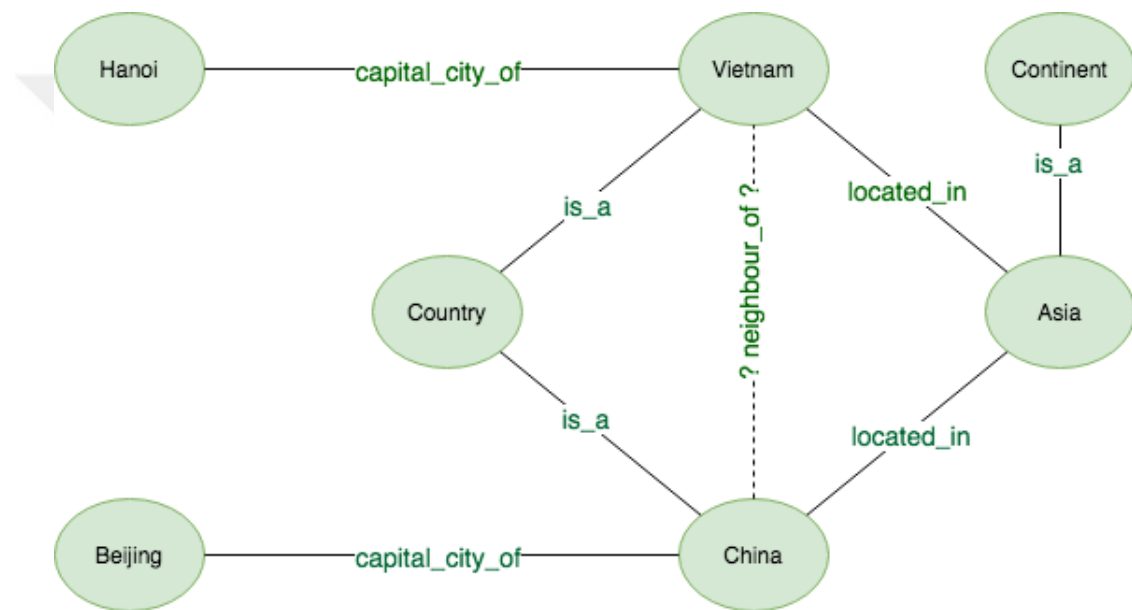


Figure 1.4: A Sub-graph To Show Link Prediction on KGs.

In *Figure – 1.3*, it is shown that Vietnam and China are countries in Asia. Is there any possibility about Vietnam and China’s neighbourhood? neighbouring countries? (Look at *Figure – 1.4*)? This is only a small example to illustrate the main process. Usually, a lot of different criteria and relations are taken into consideration while predicting new link unlike the example above. In the link prediction process, these types of possibilities are examined. Thus, new links are created, and information extraction is completed. This can be observed in the example results in *Figure – 1.5*.

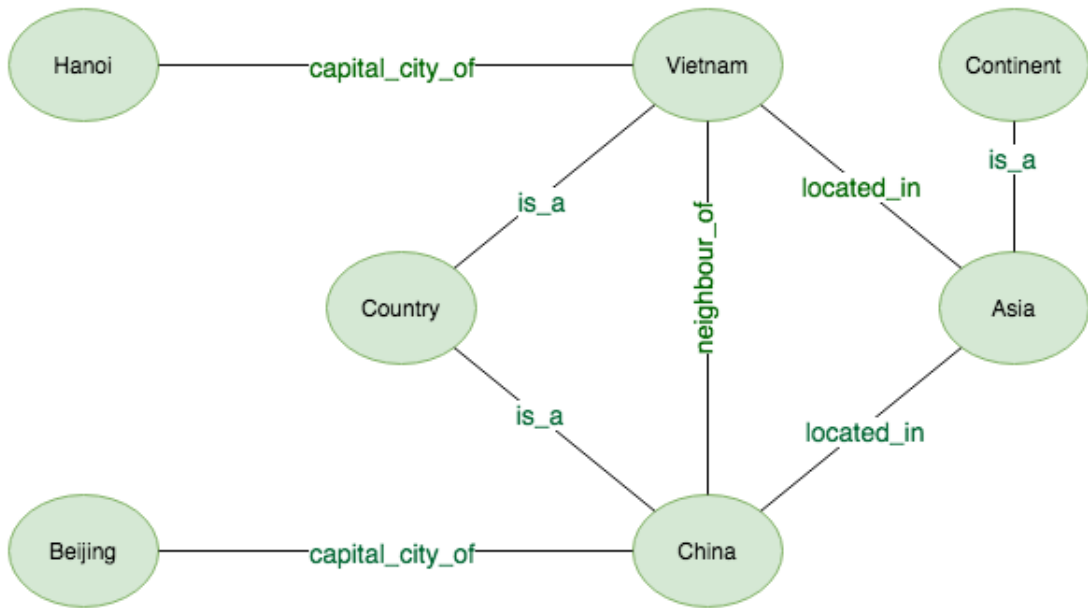


Figure 1.5: A Sub-graph To Show Link Prediction Results on KGs.

Link prediction is crucial for almost all applications because it can be used as an independent application for knowledge extraction and as a pre-process task for other applications to enhance their results to dense the training knowledge graph.

There are lots of knowledge graph embedding model for link prediction models like TransE[4], ComplEx[47], ConvKB[26], TransR[20], DistMult[54], Rescal[28], RotatE[45], TuckER[2], and ConvE[8].

1.2 Problem Statement

Knowledge graphs consist of 3 different elements; entities, relations, and literals.

Literals can be an image, video, numerical value, URI. For instance, (*vikings*,

$(number_of_seasons, 6)$ is a (e, r, l_n) type of triple, and $(vikings, imdb_link, https://www.imdb.com/title/tt2306299)$ is a (e, r, l_u) type of triple (see *Figure – 1.6*).

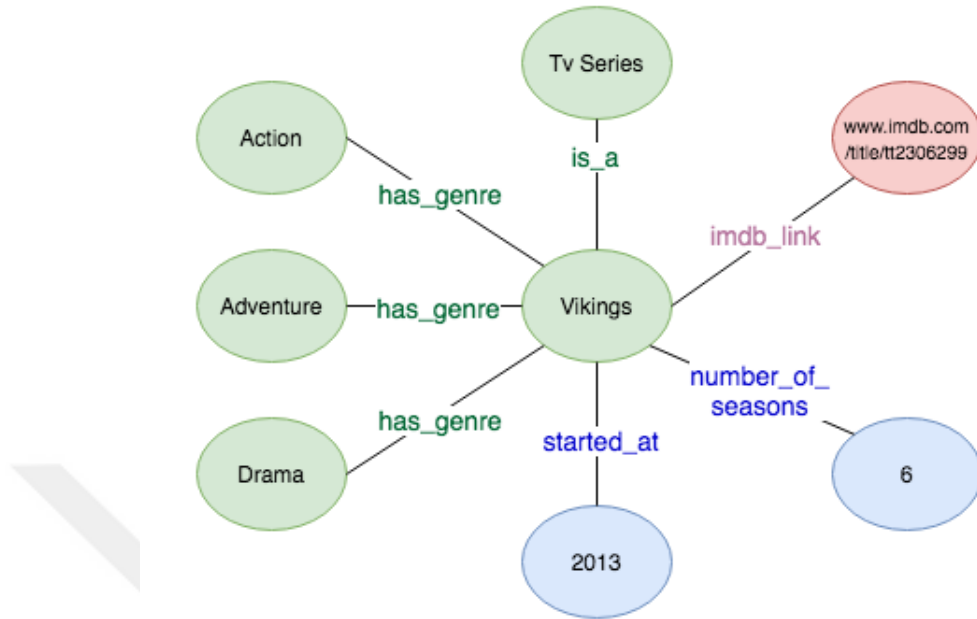


Figure 1.6: A Sub-graph To Show Instances For URL And Numerical (l_u and l_n) Literals in KGs.

The majority of studies that work on link prediction on knowledge graphs, use only entities and relations, i.e., (e, r, e) ; this decreases the data variety. Decreasing the data variety narrows the field of applications, and because of the closeness of the data distribution, it negatively affects the created embeddings’ quality. Respectively, with low-quality embeddings, results of the processes get worse. Using multi-modal data in the applications [11, 17, 32, 51], make some positive differences, but these applications are quite costly.

For instance, we have illustrated a sub-graph which includes countries and information about it in *Figure – 1.3*. However, in *Figure – 1.7*, there is extra information like dialling code and population of the countries. It can be noted, that the differences between the figures, using literals on knowledge graph applications enhance the substantiality of the applications.

In our study, we proposed a novel approach that uses not only entities and relations but also numerical triples, i.e., (e, r, l_n) . This approach does not need an external embedding process for numerical literals, and it clusters numerical literals considering

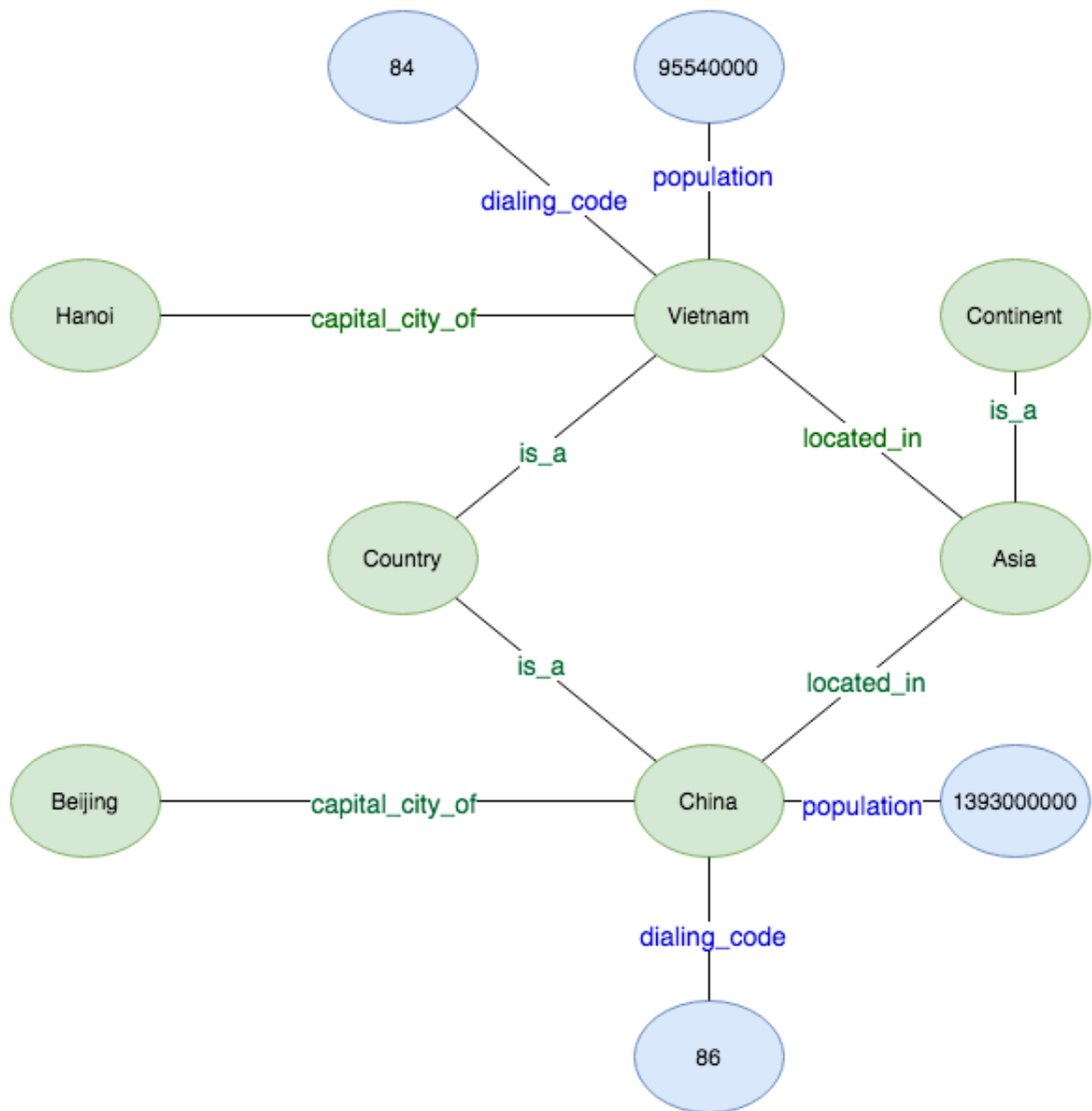


Figure 1.7: A Sub-graph To Show Numerical Literals in KGs.

their relation types, which is used in triples and processes them as entities. We worked on a part of the freebase knowledge graph, which includes entities, relations, and numerical literals, i.e. (e, r, e) , (e, r, l_n) .

1.3 Thesis Contribution

In this study, we have several contributions and enhancements. We declared our main contributions below.

- Our approach is quite simple when it is compared to former similar studies.
- Supports semantic perception owing to uses relation types based clustering method.
- Compatible with almost all knowledge graph link prediction models and applicable to knowledge graphs.
- Prevents distance based similarity approach for numerical literal values, unlike some models which consider numeric values as text.
- Captures a part of unit based differences and similarities.

1.4 Thesis Structure

This thesis contains five chapters. The chapters cover all researches that we have conducted to propose a knowledge graph link prediction approach which uses clustered numerical literal values, and all the results obtained are examined as described below.

In Chapter-II, you can find a detailed literature review for knowledge graph link prediction and knowledge graph link prediction using literal values.

In Chapter-III, the proposed approach is presented first, and also the points for the problem solving are explained as well as which are the problematic points in the state-of-the-art studies. We have also mentioned our clustering algorithm which we have customized to our triple clustering. Moreover, the compared models with their distinct features and compared results are further described.

Chapter-IV includes all the aspects of experiments and evaluation. The dataset, experiments details, parameters, evaluation metrics, and results are thoroughly described.

In the end, Chapter-V gives a summary of the study and mentions several future work ideas.



CHAPTER 2

LITERATURE

2.1 Knowledge Graph Link Prediction

To make predictions on the existence of relations, there are two types of approaches, internal and external methods [30]. Internal methods do not need any additional information, except what knowledge graph has. External methods need additional information such as other knowledge graphs. Some of the studies which used the internal approach, use a rule-based approach (rule mining) [16, 15]. As a newer approach, some studies, train a tensor neural network to make predictions [18, 42], and also mapping to lower dimensional space [16] is one of the internal methods. The external methods can use internal ones or different ones like distance-based methods [13, 36], but there is a difference, a corpus or a knowledge graph is used as an additional information source [30].

In the study[47] which proposed is the ComplEx model in 2016, they tried to maintain consistency in increasing and decreasing time and memory costs for every scale knowledge graph and also consider not only symmetric relations but also antisymmetric relations. ComplEx model is a bi-linear matrix decomposition method. The model uses dot product, thus, time and space complexity are linear for the scoring function. This employs scalability, unlike former methods that have used this approach between normal embeddings that occurred real numbers[54, 28, 27, 4], this method uses the approach between the complex embeddings (i.e., occurred complex numbers). The ComplEx method is an improved version of the DistMult[54] method via complex embeddings mentioned before. Trouillon et al. compared ComplEx, DistMult[54], and other similar methods scoring function, time and space complexity, and also link prediction results at WN18 and FB15K knowledge graphs, and it shows that the Com-

plEx method made positive differences with this approach. They showed the model performed well both on real-world datasets and artificial datasets. Furthermore, as a result, creating embedding vectors as complex numbers keeps more useful information and provides profiting of features.

Yang et al. proposed one of the basic but foremost approaches in knowledge graph embedding models named DistMult[54]. The DistMult model is a bi-linear matrix decomposition method. In the DistMult model, entities and relations are represented as vectors, and while making a prediction about them, the model uses element-wise multiplication. Nevertheless, in representation, no matter the entity located in the head or tail, the vector of the entity will be the same for both. In this way, the model performs poorly without 1-to-1 relations. The performance of the proposed model is shown and compared it with TransE[4] on FreeBase.

The TransE model is a translation based geometric knowledge graph embedding model. The model is quite simple and swallow, so it is easy to train and low cost for the link prediction process. Because of the model frugality, the model can easily apply for large scale datasets as shown in the original study and performs well on all relation types (e.g., 1-to-1, 1-to-N). The performance of the proposed model is shown and compared with Rescal[28], linear, and bi-linear models on FreeBase and WordNet. The effectiveness of the model is quite profitable when we consider scalability, costs, and performance.

The Holographic embeddings model (i.e., HolE)[27] is a geometrical and non-bilinear[37] model. The model uses a circular correlation between entity embeddings to achieve processes. Nickel et al. advocated in the study that the model can capture rich interactions, but the model is quite simple when compared with up-to-date models. This model performs well on large and small scale datasets, as shown in the study. The performance of the HolE model is described and compared with TransR[20], TransE[4], and Rescal[28] on three different datasets (i.e., FreeBase, WordNet, Countries). Free-base and WordNet datasets are large and well-known datasets, while Countries dataset is small real-world dataset. Because of the model's simplicity, the cost is quite low. Thus, the model has advantages of scalability and efficiency.

The ConvE model is one of the simplest multi-layered convolutional neural network-based knowledge graph embedding models[8]. The model has a convolution layer, a projection layer for embeddings, and another layer to multiply embedding vectors and get scores. Through using a convolutional model, capturing inverse relations is more obvious but more costly compared to other weak models. Nevertheless, this model is a low-cost model as opposed to complex models. The success of the proposed model has been proven while comparing their results with DistMult and ComplEx on Freebase, WordNet, and YAGO datasets.

The ConvKB model[26] is a convolutional neural network-based knowledge graph embedding model. Triples are represented as matrices that have a vector for each element of the triple. The matrix feeds the convolutional layer, which has filters to compose feature maps. After the filters, feature maps are gathered into a vector to symbolize the triple. As a scoring function, the dot product approach is used. The model performs well to capture relations and is tested on large datasets in the original study[26], but the link prediction process is quite high-cost because of the model complexity. In the study, the ConvKB model achieves better link prediction performance than previous state-of-the-art embedding models on two different datasets WordNet and FreeBase.

ProjE[40], one of the most basic methods, the authors not only worked on relation prediction but also entity prediction. In the ProjE method, embeddings are created via a 2-layered neural network (combination and projection), and these embeddings are presented as vectors. This method accepts graph completion as a ranking problem because when rankings are sorted, the closest values come up and no need to calculate another thing(do not use thing /makes the sentence informal) . To calculate rankings, pointwise, listwise, and weighted listwise methods are used. They are created to score and loss functions for each. In former methods like TransE [4], TransH [50], TransR [20], the same loss functions are used , so the study shows improving loss function that makes differences in results. They used DBpedia and SemMedDB knowledge graphs for showing their method's performance and compare it with others.

Sun et al. [45], who have also conducted studies on link prediction, consider dif-

ferent relationship features similarly the approach at some former studies like TransE [4], TransX [5], DistMult [54], ComplEx [47]. The features are symmetry, antisymmetry, inversion, composition. This model is a geometrical model. None of the previous studies, which used a similar approach, does not consider all of these features. For instance, the DistMult method only considers the symmetry feature. However, the RotatE method, which is developed in this study, considers all of the mentioned features. To consider these features, basic mathematical lemmas are adapted to the embedding process, and ranking function is created according to it. To improve the results of the method, a novel negative sampling technique is used. The technique is an improved version of a former study [24]. Results are shown on FB15k and WN18 knowledge graphs and compared with methods that use similar approaches to the ones mentioned before [4, 47].

Shi and Weninger [39], improve an open-world knowledge graph completion method known as ConMask, which is description-based. This method consists of two steps. The first one is content masking to reduce noisy text and also extract convenient parts, the second one is adding convenient parts to the knowledge graph via a trained fully convolutional neural network. They split to create the ConMask model into three steps, relation dependent selection of convenient words, creating a target entity from the text, choosing an entity using all features, respectively. To make relation-dependent content masking (relation dependent selection of convenient words), first, it assigns closeness weights to words and then calculates similarity score between entities using entity descriptions. In target fusion operation, 3 layered fully convolutional neural network is used, as a result, it returns k-dim embedding. In the end, a list-wise ranking loss function is used [40] to find the closest entities from the knowledge graph. To show the performance of the method, they use FreeBase and DBPedia knowledge graphs and get some higher results from TransE [4] and TransR [20] methods.

The Tensor Factorization for Knowledge Graph Completion study [2] looks at knowledge graph link prediction techniques divided into two main areas, linear techniques [28, 54, 47, 14] and non-linear techniques [1, 8]. The main difference between non-linear and linear methods is scoring functions; in non-linear model approaches,

scoring functions is a deep neural network, and also in linear models, it is a type of tensor factorization. Based on this approach, they improve a linear model named TuckER. It is a new knowledge graph link prediction technique. This method is inspired by Tucker’s study [48], the Tucker decomposition method distributes a tensor to matrices and another core tensor, and it is generally used for basic tasks like machine learning and data mining. But in this study, the method is combined with knowledge graph embedding for link prediction. The result of this innovative approach, unlike former approaches, is that all knowledge does not have to keep in only embeddings, some of the knowledge is stored in core-tensor. They compare their method with state of the art methods via scoring functions, space complexity, and results on some standard knowledge graphs (i.e., WN18, FB15k). The TuckER method shows its performance on multi-task learning on relations and gives more successful results than state of the art methods.

The ELPKG method [22] combines path-based and vector-based embedding methods, using a probabilistic method to remove knowledge abruptness, and predicts relations. The ELPKG method does not need any additional data from outside; it learns from the knowledge graph, which it is trying to complete. They show their result by comparing state-of-the-art methods (i.e., Rescal, TransE, HolE, PRA) on YAGO and NELL knowledge graphs.

Zhang et al. developed a neural network based few-shot model [55], named FSRL, for knowledge graph completion. The model is a unified form of former neural network based methods. They inspired a study [53] that used one-shot learning method for a similar process and used few-shot learning approach, which does not need a huge amount of training data for learning processes [35]. FSLR method’s steps are as following, firstly neighbour encoder takes few-shot reference set as input after that results go to aggregation network for aggregation. At last, from aggregated results, the matching network creates a matching score. As a result, it chooses the top-ranked result. They showed the performance of the proposed model and compare it with several different models[28, 4, 47, 54] on WikiData[49] and NELL[6] datasets.

2.2 Knowledge Graph Link Prediction Using Literal Values

In the MKBE study [32], unlike most prior studies, literals are not ignored, different types of data are considered like image, text description, numeric. They did not develop a new knowledge graph embedding technique, but rather created an innovative approach via using existing methods (i.e., DistMult [54], ConvE [8]) with enriched data with literals and processing different data types with neural network-based techniques (i.e., RNN, CNN) to include embedding process. To create embedding, there are no change ranking functions of DistMult and ConvE. Pezeshkpour et al. worked on multi-modal knowledge graph completion and applied it to two specific types, the first one is link prediction on multi-modal knowledge graph (relation prediction), and the second one is generating multi-modal missing data (entity prediction). For the evaluation of both two applications, they used MovieLens and YAGO knowledge graphs to show and compare step by step, pure model, numeric added model, textual description added model and image added model.

Garcia-Duran and Niepert discovered that including different entity types to embedding process is used to to make knowledge graph more fertile [11], in contrast to prior studies that used only latent entities. Moreover, to achieve and show that they use three different entity types - latent, numeric, and combined relations. The product of experts model is used for this process. To show the results, they used FB15k and WN18 knowledge graphs and compared results with the former to develop methods step by step.

One of the different approaches in knowledge graph embedding for link prediction is the study conducted by Kristiadi et al. [17]. The study developed a simple method for including literals into embedding methods, named as LiteralE, to make more successful link predictions from embeddings. The first step of the process is to divide link prediction methods into two [29], graph feature methods, and latent graph feature methods, and then worked on improving the second type of methods. Latent graph feature methods [8, 47, 9, 11, 46] generally learn entity and relation embeddings and measure closeness or likeliness via score functions or other probabilistic methods. It is considered as a ranking problem. However, some of these methods [11, 46] either

do not reckon on literals while creating embeddings or add literals externally. The study, unlike others, considers literals while creating embeddings, and the latent feature method is applied to basic existent methods like DistMult [54], ComplEx [47], and ConvE [8]. Furthermore, to show and compare results, they used FreeBase and YAGO knowledge graphs. They compared latent graph feature methods do not reckon with literals, latent graph feature methods reckon with literals, and combined with LiteralE methods(unclear sentence). As a result, combining LiteralE method with other embedding methods (considering literals while embedding) that are mentioned before, makes positive differences, especially for the link prediction tasks.

The TransEA model [51] is one of the studies which incorporate literal elements for the knowledge graph embedding process. The model arose from the TransE [4] model. As we mentioned before TransE model is a translation based embedding model. Wu et al. combined the attribute-based embedding model (for numerical literals) with TransE. In the study, they showed the results for the model, on YAGO [44] and FreeBase [3] comparing with TransE for the link prediction process. However, this approach can only be applied to the TransE model, so it is quite hard to compare it with other models.

CHAPTER 3

LINK PREDICTION WITH CLUSTERED LITERALS

3.1 Our Approach

Usually, knowledge graphs are composed of different elements: entities, relations, and literals. However, almost all knowledge graph embedding methods and applications presume it as consisting of only entities and relations, as mentioned before like in *Figure – 1.1*. In other words, they use only (e, r, e) triples, but in knowledge graphs, there are also (e, r, l) triples like in *Figure–1.7*. Different types of literals can be present in knowledge graphs. Overpassing the literal data prevents the use of a considerable part of the knowledge graph. Eventually, considering literal data means enriching training data, and this leads to more effective results in knowledge graph embedding applications (e.g., Q&A, link prediction).

We will look at problems on KG LP models which uses numerical literals because the approach which is proposed in this study is about using numerical literals on KG LP models.

In the section below, references are made to weak points of the state-of-the-art studies including a proposal on how to solve this issues.

- All of the similar studies mentioned above, which use literals, first create embeddings of (e, r, e) triples by using traditional knowledge graph embedding techniques, after that, create embeddings of literals (*i.e. (e, r, l) triples*) by using different techniques like neural networks. In the end, all embeddings are gathered. You can see this approach with general processes in *Figure – 3.1*. The mentioned approach, which is used in previous studies, is a high-cost approach for time and space complexity. Because of that, in our approach, we gathered two embedding processes into one process. In this way, we contribute to the efficiency

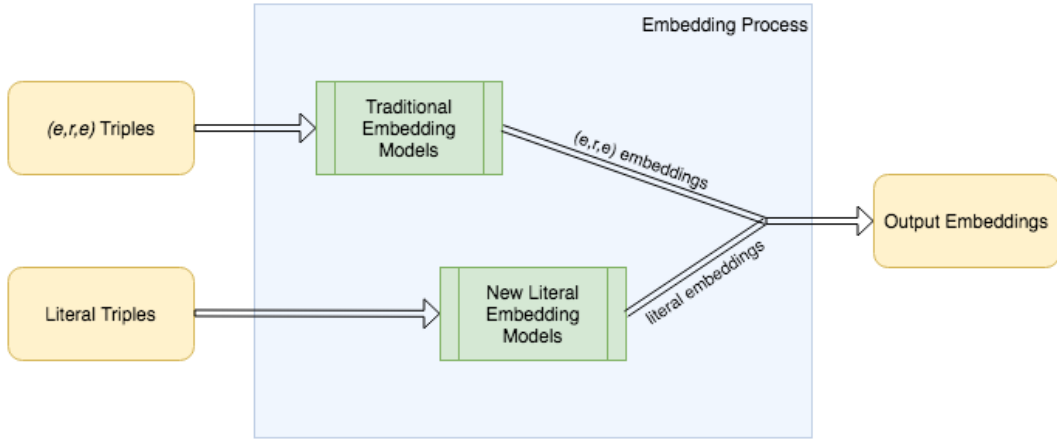


Figure 3.1: The approach of the state-of-the-art studies on knowledge graph link prediction using literal values.

of the approach by decreasing complexity.

- Despite the support of the state-of-the-art studies for numerical literals, all of the embedding methods discussed fail to interpret the semantics behind data types of literals and units. For an instance, as we can see in the *Figure – 3.2*, ‘1999€ ’ and ‘the year 1999’ could be considered the same because type semantics are discarded. And also, we summarize the same situations between *Table – 3.1*’s triples 5, 6, 7, and triples 8, 9. Nevertheless, in our approach, during the clustering process, clusters are created according to the relation types. Consequently, there is no perception problem in different units.
- None of the state-of-the-art models apply normalization for literal values; hence the semantic similarity between two literal values can not be caught. For example, as we can see in *Table – 3.1*, there are several triples about two different aspects . Triple 1 and 2 mean the length of the Insect775 is 200 mm and triple 3 and 4 mean diameter of Circle231 is 2 cm. 200 mm and 2 cm is the same length, but models perceived them as different values, so, the similarity between 200 mm and 2 cm is not captured. In our approach, because of the clustering process is according to distinct relation types, most part of this problem is solved. It is possible that there are exceptions to the case.
- The TransEA[51] study can only be used on TransE[4] model. In our approach, we

used triples that included numerical literals and categorized the numerical literals before the embedding process (see Figure–3.3), and every created category used the current knowledge graph model’s embedding method. Thus, we obtained that it can be applied for all datasets and knowledge graph embedding applications.

- Instead of processing numerical literals, if we directly add numerical literals, like some models which consider numeric values as text, the model perceives like 199 is closer than 400 to 399, semantically causing wrong results.

Table 3.1: Triple Instances To Show Perception Problems For KG LP with Literals

	<i>Head</i>	<i>Relation</i>	<i>Tail</i>
1	Insect775	length	200
2	Insect775	unit	mm
3	Circle231	diameter	2
4	Circle231	unit	cm
5	Computer	price	1999
6	Computer	currency	Euro
7	Computer	is_a	Product
8	XYZ	established_at	1999
9	XYZ	is_a	Brand

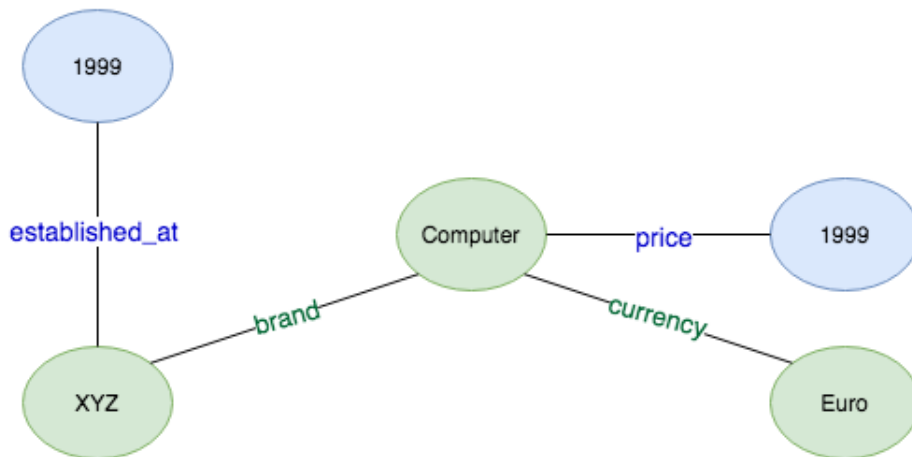


Figure 3.2: A Sub-graph To Show Perception Problems For KG LP with Literals.

We mentioned novel points of our approach to see general flow and main points of

the approach in this study which you can see at *Figure – 3.3*. Now we will look at the main processes in our approach.

One of the main processes is the clustering process. The clustering process is categorizing literal values. In the categorization process, similarly with KBLRN[11], which used PoE, this study used X-Means[31]. X-Means is an extended version of K-Means, cultivating clusters by repeatedly striving subclass and keeping the best splits, until a criterion. So, unlike K-means, the number of the clusters is determined automatically according to the distribution of the data in X-means algorithm.

In this study, we adapted X-means depending on the relation types, and we used BIC (Bayesian information criterion) as criteria. The number of the clusters determined up to the distribution of the data by the X-means algorithm is not a distinct value. Thus, we prevented the numerical values in different units being perceived as the same and obtained a more low-cost process.

One of the other main processes is the knowledge graph embedding process. Under favour of our approach’s compatibility with all knowledge graph embedding models, as we mentioned before, we applied six traditional models to our approach. There is a short explanation about why we used these traditional knowledge graph embedding models in *Chapter – 3.2*.

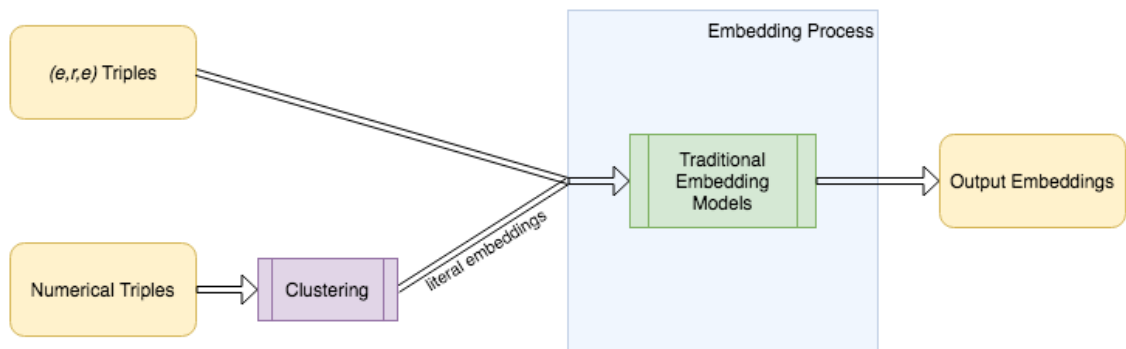


Figure 3.3: Our approach: Knowledge Graph Link Prediction Using Clustered Literal Values.

Finally, to sum up briefly the approach details, our approach:

- Supports semantic perception.
- Compatible with almost all knowledge graph embedding models.

- Compatible with almost all knowledge graphs.
- Prevents distance based similarity approach for numerical literal values.
- Uses relation types based clustering method.
- Captures unit based differences and similarities.

3.2 Models Compared

In our study, we used six different models to apply our approach. We wanted to show our approach can be applied to different types and different complexity models. So, we chose models with several architecture types. Some of the models are developed on the other one, which uses the same architecture; we chose some of them because we wanted to show and prove our approach’s stability and consistency.

In *Table – 3.2*, we gave this model’s link prediction results and showed that different models perform differently. We use MRR (Main Reciprocal Rank) and HIT@10 (Hits at first 10 predictions) metrics to show performances. Detailed explanations of the metrics can found in *Section – 4.3*. Also, we used the dataset at *Section – 4.2* and the same experimental setup with *Section – 4.1*

Table 3.2: Compared Models’ LP Results Comparison Table

MODEL	MRR	HIT@10
ComplEx	0.30	0.47
DistMult	0.28	0.45
TransE	0.23	0.34
HolE	0.25	0.39
ConvE	0.22	0.31
ConvKB	0.20	0.42

The common and the distinct points, scoring functions, and important sides of the models are explained below:

3.2.1 TransE

TransE model is a translation based simple model. It is based on distance measuring. In the scoring function, the similarity between embeddings is calculated, using distances of the embedding vectors. (*See Equation – 3.1*)

$$f_{transe} = -||h + r - t|| \quad (3.1)$$

3.2.2 DistMult

The DistMult model is a bi-linear matrix decomposition method. It uses the tri-linear dot product for the scoring function and element-wise multiplication and in representation, therefore regardless of the entity located in the head or tail, the vector of the entity will be the same for both. (*See Equation – 3.2*)

$$f_{distmult} = \langle h, r, t \rangle \quad (3.2)$$

3.2.3 ComplEx

ComplEx model is a bi-linear matrix decomposition method. The model was developed by extending DistMult model with the Hermitian dot product [38]. (*See Equation – 3.3*)

$$f_{complex} = Re(\langle h, r, \bar{t} \rangle) \quad (3.3)$$

3.2.4 HolE

HolE is a geometrical and non-bilinear [37] model. Moreover, the model uses a circular correlation between entity embeddings as a scoring function. (*See Equation – 3.4*)

⊗ implies the circular correlation function.

$$f_{hole} = \langle h, r \otimes t \rangle \quad (3.4)$$

3.2.5 ConvE

The ConvE model is one of the simplest multi-layered convolutional neural network based knowledge graph embedding models. The model has a convolution layer, a projection layer for embeddings, and another layer to multiply embedding vectors and get scores. The scoring function of the model is given in the *Equation – 3.5*.

g implies a non-linear activation function, $*$ implies the linear convolution operator, vec denotes a 2D reshaping of vector and also $.$ donates dot product. Ω implies a set of filters for the convolutional model.

$$f_{conve} = g(vec(g(concat(h, r) * \Omega))W).t \quad (3.5)$$

3.2.6 ConvKB

The ConvKB model is a convolutional neural network based knowledge graph embedding model. Triples are represented as matrices that have a vector for each element of the triple. The matrix feeds the convolutional layer, which has filters to compose future maps. After that filters, future maps are gathered into a vector to symbolize the triple. And after, the dot product approach is used. The scoring function of the model is given in the *Equation – 3.6*.

g donates a non-linear activation function, $*$ implies the linear convolution operator and $.$ donates dot product. Ω implies a set of filters for the convolutional model and $concat$ is used to imply concatenation operator.

$$f_{convkb} = concat(g([h, r, t] * \Omega)).w \quad (3.6)$$

In the end, every model, which mentioned above, has a different performance. In *Table – 3.2*, we showed the models performances on the first part of the dataset (look at *Section – 4.2*) and it is shown that ComplEx model is the best performing model on our data and environment. And DistMult, HolE, TransE, ConvE, ConvKB models perform respectively.

CHAPTER 4

EVALUATION

4.1 Experimental Setup

We tried to stabilize all parameters and factors to ensure the reliability of the comparisons. The parameters used in this study and their explanations as follows:

(Some of the parameters are obliged to change up to used model)

- *batches_count*: 100
- *epochs*: 400
- *k*: 200
- *eta*: 20
- *optimizer*: 'adam'
- *optimizer_params*: 'lr':1e-4
- *loss*: 'multiclass_nll'
- *regularizer*: 'LP'
- *regularizer_params*: 'p':3, 'lambda':1e-5

As an exception, for ConvE model, we used 'bce' loss function instead of 'multiclass_nll'.

batches_count is the number of batches, splitting the number of training set for each loop. *epochs* imply the iterations of the training loop, and *k* implies the dimension of the embedding space. *eta* is the number of negative triples which created artificially for each positive triple and *optimizer* implies the optimizer for minimizing the loss function. *loss* is the type of loss function for training and *regularizer* is the regularization strategy to use with the loss function.

For the implementation part, we used Python’s AmpliGraph[7] library. It is an open source Python library that implements several knowledge graph link prediction models.

4.2 Dataset

In this study, we used a part of FreeBase[3] knowledge graph. The dataset consists of two different parts which are mentioned as FB and N in Table-4.1.

The first part of the dataset which we mentioned as FB in the Table-4.1, includes 324340 triples which is in (e, r, e) format with 1003 relations and 13882 entities. We used this part to show the existent knowledge graph link prediction models performance and to compare our approach.

The second part of the dataset which we mentioned as N in the Table-4.1 which is in (e, r, l_n) format. We concatenated both FB and N and used which includes 327534 triples with 1011 relations, 13882 entities and 3194 numerical literals to show models performance with numeric literals and our approach performance.

Table 4.1: Datasets Details

KG	Triple	$ \mathcal{R} $	$ \mathcal{E} $	Numeric
FB	324340	1003	13882	-
FB + N	327534	1011	13882	3194

4.3 Evaluation Metrics

Almost all of the state-of-the-art studies use ranking based evaluation metrics. The most common ones are *MRR* (*i.e.*, *mean reciprocal rank*), *MR* (*i.e.*, *mean rank*), and *HIT@N*. We use the most commonly used evaluation metrics, which we mentioned before, to evaluate our experiments.

As shown in the *Section – 4.3.1*, for each prediction, a ranking list is created. All of the evaluation metrics that we used are calculated using these list indices.

We explained the calculations of the metrics below. Q implies the set of test triples.

4.3.1 Ranking

In the testing process, for each triple, a list of predicted triples are created on the trained model. The rank of the triple corresponds to the index of the correct triple in the list.

$rank_{(s,p,o)}$ function returns the ranking of the (s,p,o) triple and a ranking.

4.3.2 MRR

Mean reciprocal rank is the harmonic mean of ranking values.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_{(s,p,o)_i}} \quad (4.1)$$

The higher value is represents a better result for MRR metric.

4.3.3 MR

Mean rank value is the average rank value of all triples (i.e., *arithmetic mean*).

$$MR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} rank_{(s,p,o)_i} \quad (4.2)$$

The lower value is represents a better result for MR metric.

4.3.4 HIT@N

The ratio of triples that have top-N ranking to all other test triples.

$$Hits@N = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \begin{cases} 1 & \text{if}(rank_{(s,p,o)_i} \leq N) \\ 0 & \text{otherwise} \end{cases} \quad (4.3)$$

In literature, commonly, 1, 3, 5, and 10 is used for the N value. The higher value is represents a better result for the HIT@N metric.

4.4 Experimental Results

To show our approach performance, pros and cons, we conducted the experiments on six different typed conventional knowledge graph embedding models, namely ComplEx[47], DistMult[54], TransE[4], HolE[27], ConvE[8], and ConvKB[26]. You can find features of these methods in Chapter-II. For each model, we conducted three experiments,

- In *Experiment – 1*, the train and test sets consist of only (e, r, e) triples. It shows the model’s performance.
- In *Experiment – 2*, the train and test sets consist of both (e, r, e) and (e, r, l_n) triples. The main aim of this experiment is to show the performance of the model (e, r, l_n) triples are directly added to (e, r, e) triples.
- In *Experiment – 3*, the train and test sets consist of both (e, r, e) and (e, r, l_n) triples such as in *Experiment – 2*. The main aim of this experiment is to show the performance of the model with our approach, (e, r, l_n) triples are not directly added to (e, r, e) triples. Before the training process, l_n elements of the triples are clustered via considering the type of the r elements.

Our main aim in these experiments is to show change on link prediction results between when we used only (e, r, e) triples, directly added (e, r, l_n) triples, and clustered added (e, r, l_n) triples. We summarized all of the experimental results in Table 4.2,

In Table 4.2, we used *Exp-1*, *Exp-2*, and *Exp-3* respectively for *Experiment – 1*, *Experiment – 2*, and *Experiment – 1*. We showed the results on models for explained experiments above. To compare the results of experiments, we showed five different evaluation metrics in the table.

We need to mention some details about the experiments; we split the dataset into train and test sets randomly. %67 for the train set and %33 for the test set. However, the splitting process is not entirely random; the test set must include every entity and relation type in the train set. We tried to stabilize all parameters and other factors like hardware to take more consistent results and ensure the truth on comparisons; we kept them as steady as possible. The parameters which were used in this study

and their explanations as follows; the number of epochs (iterations) for training is 400, batch count for every training epoch is 100, embedding space dimension is 200, the number of created negative samples for each positive sample is 20, to minimize loss function 'adam' optimizer is used, for loss function 'multiclass_nll' is used. However, as an exception, the ConvE model is not compatible with the 'multiclass_nll' loss function, so 'bce' loss function is used for the ConvE model.

In experiments results can differ up to dataset, dataset split ratio, hardware, environment, parameters, and also other reasons. In our experiments, as we mentioned above, we tried the all of them stable to make reasonable comparisons on results.

Generally, the comparisons on knowledge graph link prediction, are made with MRR and hits@10 metrics, because these metrics are more distinctive and consistent compared to others. We preferred to evaluate the results of this study with these two metrics from now on.

For all models, there is a considerable difference between *Experiment - 1* and *Experiment - 2*, and also between *Experiment - 2* and *Experiment - 3*. The difference between *Experiment - 1* and *Experiment - 2* is more substantial than *Experiment - 2* and *Experiment - 3*; this is because, when we add different types of data, the distribution of the data changed but, in *Experiment - 3*, after added clustering numeric literals we prevented wrong semantic perceives so it improved *Experiment - 2*'s results.

The amount of the added data is quite small, but its effect of enhancement on results is enormous. As we mentioned above, adding different types of data changes the distribution of embeddings, this changed not only the numerical triples embeddings but also the regular triples embeddings.

While the differences in some models' results are relatively big, other are significantly small, for instance, the difference between the TransE model's *Experiment - 1* and *Experiment - 3* is around 0.30, but the difference between HolE model's *Experiment - 1* and *Experiment - 3* is around 0.05 on MRR metric with the same datasets. The reason for this is the embedding model's procedures and geometries; when triples embedding, triples affect each other up to model geometry.

Table 4.2: Result Table

MODEL		MRR	MR	HITS@10	HITS@3	HIT@1
ComplEx	Exp-1	0.30	356.81	0.47	0.33	0.22
	Exp-2	0.42	157.46	0.61	0.47	0.31
	Exp-3	0.53	120.49	0.73	0.59	0.42
DistMult	Exp-1	0.28	345.61	0.45	0.31	0.19
	Exp-2	0.39	154.06	0.57	0.44	0.29
	Exp-3	0.44	123.25	0.65	0.50	0.33
TransE	Exp-1	0.23	1068.94	0.34	0.25	0.16
	Exp-2	0.50	253.47	0.65	0.49	0.38
	Exp-3	0.53	120.27	0.73	0.60	0.43
HoIE	Exp-1	0.25	347.14	0.39	0.27	0.18
	Exp-2	0.28	216.08	0.44	0.31	0.21
	Exp-3	0.30	199.06	0.47	0.33	0.22
ConvE	Exp-1	0.22	3448.54	0.31	0.25	0.18
	Exp-2	0.24	1458.22	0.36	0.27	0.18
	Exp-3	0.25	927.55	0.38	0.27	0.18
ConvKB	Exp-1	0.20	3474.44	0.42	0.20	0.14
	Exp-2	0.29	1294.28	0.49	0.32	0.21
	Exp-3	0.33	153.82	0.53	0.37	0.23

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

We proposed a new approach to use numerical literals in link prediction models on knowledge graphs. Our approach can be applied to all knowledge graph embedding models and does not have high cost like state-of-the-art studies. As shown in the experiments, changing data distribution affects the embedding process and its effects are clearly shown in the results. Increasing the data variety of knowledge graphs causes both improved results because of data distribution and increased process's domain area on the knowledge graph. However, it also results in higher time and memory complexities. For further studies, while improving and adapting this approach, we are planning to add other literal data types (e.g., image, URI) with lower costs as opposed to existing methods.

REFERENCES

- [1] Ivana Balažević, Carl Allen, and Timothy M Hospedales. “Hypernetwork knowledge graph embeddings”. In: *International Conference on Artificial Neural Networks*. Springer. 2019, pp. 553–565.
- [2] Ivana Balažević, Carl Allen, and Timothy M Hospedales. “Tucker: Tensor factorization for knowledge graph completion”. In: *arXiv preprint arXiv:1901.09590* (2019).
- [3] Kurt Bollacker et al. “Freebase: a collaboratively created graph database for structuring human knowledge”. In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008, pp. 1247–1250.
- [4] Antoine Bordes et al. “Translating embeddings for modeling multi-relational data”. In: *Advances in neural information processing systems*. 2013, pp. 2787–2795.
- [5] Antoine Bordes et al. “Translating embeddings for modeling multi-relational data”. In: *Advances in neural information processing systems*. 2013, pp. 2787–2795.
- [6] Andrew Carlson et al. “Toward an architecture for never-ending language learning”. In: *Twenty-Fourth AAAI conference on artificial intelligence*. 2010.
- [7] Luca Costabello et al. *AmpliGraph: a Library for Representation Learning on Knowledge Graphs*. Mar. 2019. DOI: 10.5281/zenodo.2595043. URL: <https://doi.org/10.5281/zenodo.2595043>.
- [8] Tim Dettmers et al. “Convolutional 2d knowledge graph embeddings”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

- [9] Xin Dong et al. “Knowledge vault: A web-scale approach to probabilistic knowledge fusion”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 601–610.
- [10] Jeffrey Scott Eder. *Knowledge graph based search system*. US Patent App. 13/404,109. June 2012.
- [11] Alberto Garcia-Duran and Mathias Niepert. “Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features”. In: *arXiv preprint arXiv:1709.04676* (2017).
- [12] Larry Heck and Hongzhao Huang. “Deep learning of knowledge graph embeddings for semantic parsing of twitter dialogs”. In: *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE. 2014, pp. 597–601.
- [13] Johannes Hoffart et al. “Robust disambiguation of named entities in text”. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. 2011, pp. 782–792.
- [14] Seyed Mehran Kazemi and David Poole. “Simple embedding for link prediction in knowledge graphs”. In: *Advances in neural information processing systems*. 2018, pp. 4284–4295.
- [15] Jiseong Kim et al. “The Association Rule Mining System for Acquiring Knowledge of DBpedia from Wikipedia Categories.” In: *NLP-DBPEDIA@ ISWC*. 2015, pp. 68–80.
- [16] Kristian Kolthoff and Arnab Dutta. “Semantic relation composition in large scale knowledge bases”. In: *CEUR Workshop Proceedings*. Vol. 1467. RWTH. 2015, pp. 34–47.
- [17] Agustinus Kristiadi et al. “Incorporating literals into knowledge graph embeddings”. In: *arXiv preprint arXiv:1802.00934* (2018).
- [18] Denis Krompaß, Stephan Baier, and Volker Tresp. “Type-constrained representation learning in knowledge graphs”. In: *International semantic web conference*. Springer. 2015, pp. 640–655.

- [19] Jens Lehmann et al. “DBpedia—a large-scale, multilingual knowledge base extracted from Wikipedia”. In: *Semantic web 6.2* (2015), pp. 167–195.
- [20] Yankai Lin et al. “Learning entity and relation embeddings for knowledge graph completion”. In: *Twenty-ninth AAAI conference on artificial intelligence*. 2015.
- [21] Yang Liu et al. “Stock price movement prediction from financial news with deep learning and knowledge graph embedding”. In: *Pacific Rim Knowledge Acquisition Workshop*. Springer. 2018, pp. 102–113.
- [22] Jiangtao Ma et al. “ELPKG: A High-Accuracy Link Prediction Approach for Knowledge Graph Completion”. In: *Symmetry* 11.9 (2019), p. 1096.
- [23] Fang Miao et al. “Construction of semantic-based traditional Chinese medicine prescription knowledge graph”. In: *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE. 2018, pp. 1194–1198.
- [24] Tomas Mikolov et al. “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.
- [25] George A Miller. “WordNet: a lexical database for English”. In: *Communications of the ACM* 38.11 (1995), pp. 39–41.
- [26] Dai Quoc Nguyen et al. “A novel embedding model for knowledge base completion based on convolutional neural network”. In: *arXiv preprint arXiv:1712.02121* (2017).
- [27] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. “Holographic embeddings of knowledge graphs.” In: *AAAI*. Vol. 2. 1. 2016, pp. 3–2.
- [28] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. “A three-way model for collective learning on multi-relational data.” In: *Icml*. Vol. 11. 2011, pp. 809–816.
- [29] Maximilian Nickel et al. “A review of relational machine learning for knowledge graphs”. In: *Proceedings of the IEEE* 104.1 (2015), pp. 11–33.

- [30] Heiko Paulheim. “Knowledge graph refinement: A survey of approaches and evaluation methods”. In: *Semantic web* 8.3 (2017), pp. 489–508.
- [31] Dan Pelleg, Andrew W Moore, et al. “X-means: Extending k-means with efficient estimation of the number of clusters.” In: *Icml*. Vol. 1. 2000, pp. 727–734.
- [32] Pouya Pezeshkpour, Liyan Chen, and Sameer Singh. “Embedding multimodal relational data for knowledge base completion”. In: *arXiv preprint arXiv:1809.01341* (2018).
- [33] Danh Le-Phuoc et al. “The Graph of Things: A step towards the Live Knowledge Graph of connected things”. In: *Journal of Web Semantics* 37 (2016), pp. 25–35.
- [34] Richard Qian. “Understand your world with bing”. In: *Bing search blog, Mar* (2013).
- [35] Sachin Ravi and Hugo Larochelle. “Optimization as a model for few-shot learning”. In: (2016).
- [36] Giuseppe Rizzo and Raphaël Troncy. “Nerd: evaluating named entity recognition tools in the web of data”. In: *Workshop on Web Scale Knowledge Extraction (WEKEX’11)*. Vol. 21. 2011.
- [37] Andrea Rossi et al. “Knowledge Graph Embedding for Link Prediction: A Comparative Analysis”. In: *arXiv preprint arXiv:2002.00819* (2020).
- [38] Helmut H Schaefer. “Locally Convex Topological Vector Spaces”. In: *Topological Vector Spaces*. Springer, 1971, pp. 36–72.
- [39] Baoxu Shi and Tim Weninger. “Open-world knowledge graph completion”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- [40] Baoxu Shi and Tim Weninger. “ProjE: Embedding projection for knowledge graph completion”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.
- [41] Amit Singhal. “Introducing the knowledge graph: things, not strings”. In: *Official google blog* 5 (2012).

- [42] Richard Socher et al. “Reasoning with neural tensor networks for knowledge base completion”. In: *Advances in neural information processing systems*. 2013, pp. 926–934.
- [43] Tom Stocky and Lars Rasmussen. “Introducing Graph Search Beta”. In: *Retrieved February 12 (2013)*, p. 2013.
- [44] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. “Yago: a core of semantic knowledge”. In: *Proceedings of the 16th international conference on World Wide Web*. 2007, pp. 697–706.
- [45] Zhiqing Sun et al. “Rotate: Knowledge graph embedding by relational rotation in complex space”. In: *arXiv preprint arXiv:1902.10197* (2019).
- [46] Yi Tay et al. “Multi-task neural network for non-discrete attribute prediction in knowledge graphs”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 1029–1038.
- [47] Théo Trouillon et al. “Knowledge graph completion via complex tensor factorization”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 4735–4772.
- [48] Albert W Tucker and Philip D Straffin Jr. “The mathematics of Tucker: A sampler”. In: *The Two-Year College Mathematics Journal* 14.3 (1983), pp. 228–232.
- [49] Denny Vrandečić and Markus Krötzsch. “Wikidata: a free collaborative knowledgebase”. In: *Communications of the ACM* 57.10 (2014), pp. 78–85.
- [50] Zhen Wang et al. “Knowledge graph embedding by translating on hyperplanes.” In: *Aaai*. Vol. 14. 2014. 2014, pp. 1112–1119.
- [51] Yanrong Wu and Zhichun Wang. “Knowledge graph embedding with numeric attributes of entities”. In: *Proceedings of The Third Workshop on Representation Learning for NLP*. 2018, pp. 132–136.
- [52] Chenyan Xiong, Russell Power, and Jamie Callan. “Explicit semantic ranking for academic search via knowledge graph embedding”. In: *Proceedings of the 26th international conference on world wide web*. 2017, pp. 1271–1279.

- [53] Wenhan Xiong et al. “One-shot relational learning for knowledge graphs”. In: *arXiv preprint arXiv:1808.09040* (2018).
- [54] Bishan Yang et al. “Embedding entities and relations for learning and inference in knowledge bases”. In: *arXiv preprint arXiv:1412.6575* (2014).
- [55] Chuxu Zhang et al. “Few-Shot Knowledge Graph Completion”. In: *arXiv preprint arXiv:1911.11298* (2019).



APPENDIX A
NOMENCLATURE

Table A.1: Nomenclature Table.

h	head element of triple
r	relation element of triple
t	tail element of triple
e	entity
l	a literal
l_n	a numerical literal
l_u	an url literal
l_i	an image literal
l_v	a video literal
f_{transe}	transe scoring function
$f_{distmult}$	distmult scoring function
$f_{complex}$	complex scoring function
f_{hole}	hole scoring function
f_{conve}	conve scoring function
f_{convkb}	convkb scoring function
Re	real axis
\otimes	circular correlation

Ω	a set of filters for the convolutional model
g	non-linear activation function
$*$	linear convolution operator
vec	a 2D reshaping of vector
$concat$	conctenation operator
\mathcal{R}	set of relations
$ \mathcal{R} $	number of relations
\mathcal{E}	set of entities
$ \mathcal{E} $	number of entities
k	dimension of the embedding space
eta	number of negatives triples for each positive triple

APPENDIX B
DATA SET DESCRIPTION

Table B.1: (e, r, e) Triples From Dataset

<i>head</i>	relation	tail
/m/027z0pl	/film/producer/film	/m/080nwsb
/m/04htfd	/organization/organization/leadership./organization/leadership/role	/m/0dq_5
/m/020xn5	/film/film_job/films_with_this_crew_job./film/film_crew_gig/film	/m/04n52p6
/m/0kzy0	/people/person/profession	/m/0gbbt
/m/01pgzn_	/base/popstar/celebrity/shops_at./base/popstar/shopping_choice/company	/m/0r0m6
/m/02jqjm	/music/musical_group/member./music/group_membership/role	/m/018vs
/m/0170_p	/film/film/country	/m/09c7w0
/m/03176f	/film/film/distributors./film/film_film_distributor_relationship/distributor	/m/086k8
/m/0gffmn8	/film/film/starring./film/performance/actor	/m/015lhm
/m/05gml8	/base/popstar/celebrity/friendship./base/popstar/friendship/participant	/m/03m8lq
/m/07kb5	/user/alexander/philosophy/philosopher/interests	/m/02jcc
/m/09byk	/people/person/languages	/m/04306rv
/m/07_dn	/business/employer/employees./business/employment_tenure/title	/m/060c4
/m/043dix	/government/legislative_session/members./government/government_position_held/district_represented	/m/05fkf
/m/018vs	/music/instrument/instrumentalists	/m/0274ck
/m/012cj0	/people/person/nationality	/m/09c7w0

Table B.2: (e, r, l_n) Triples From Dataset

<i>head</i>	<i>relation</i>	<i>tail</i>
/m/072kp	/tv/tv_program/number_of_seasons	9.0
/m/0358x_	/tv/tv_program/number_of_seasons	49.0
/m/01yb1y	/tv/tv_program/number_of_seasons	14.0
/m/02rzdcp	/tv/tv_program/number_of_episodes	92.0
/m/016tvq	/tv/tv_program/number_of_episodes	202.0
/m/02py9yf	/tv/tv_program/number_of_episodes	91.0
/m/0199wf	/film/film/initial_release_date	1978
/m/045r_9	/film/film/initial_release_date	1985
/m/0b76kw1	/film/film/initial_release_date	2010
/m/02qjv1p	/tv/tv_program/air_date_of_final_episode	2010
/m/025ljp	/tv/tv_program/air_date_of_final_episode	2009
/m/04hs7d	/tv/tv_program/air_date_of_final_episode	2004
/m/0266s9	/tv/tv_program/air_date_of_final_episode	2010
/m/0d68qy	/tv/tv_program/air_date_of_final_episode	2013
/m/01q_y0	/tv/tv_program/air_date_of_final_episode	2006
/m/04x4gj	/tv/tv_program/air_date_of_first_episode	1985
/m/02qkq0	/tv/tv_program/air_date_of_first_episode	1986
/m/01hvv0	/tv/tv_program/air_date_of_first_episode	2001
/m/0ddd0gc	/tv/tv_program/episode_running_time	65.0
/m/017dbx	/tv/tv_program/episode_running_time	30.0
/m/0gxsh4	/tv/tv_program/episode_running_time	60.0

Table B.3: (e, r, l_n) Triples From Dataset After Clustering Process

<i>head</i>	<i>relation</i>	<i>tail</i>
/m/05h95s	/tv/tv_program/air_date_of_first_episode	cluster5
/m/06qwh	/tv/tv_program/air_date_of_first_episode	cluster6
/m/06r4f	/tv/tv_program/episode_running_time	cluster13
/m/027tbrc	/tv/tv_program/episode_running_time	cluster11
/m/02skyy	/tv/tv_program/episode_running_time	cluster11
/m/06f0k	/tv/tv_program/episode_running_time	cluster14
/m/0sxgv	/film/film/initial_release_date	cluster8
/m/0gy4k	/film/film/initial_release_date	cluster8
/m/01vksx	/film/film/initial_release_date	cluster8
/m/0gj50	/tv/tv_program/number_of_episodes	cluster15
/m/02648p	/tv/tv_program/number_of_episodes	cluster17
/m/01s81	/tv/tv_program/number_of_episodes	cluster15
/m/0q9jk	/tv/tv_program/number_of_seasons	cluster0
/m/0524b41	/tv/tv_program/number_of_seasons	cluster3
/m/06mr2s	/tv/tv_program/number_of_seasons	cluster1
/m/0524b41	/tv/tv_program/number_of_seasons	cluster1
/m/02py9yf	/tv/tv_program/air_date_of_final_episode	cluster21
/m/06qwh	/tv/tv_program/air_date_of_final_episode	cluster22
/m/01s81	/tv/tv_series_season/number_of_episodes	cluster15

APPENDIX C

DETAILED EXPERIMENTS ON TRANSE MODEL

We conducted several experiments to show results more detailed. In *Table – C.1*, we showed our results on the TransE model. The experiments and their explanations are as follows.

- In *Experiment – A*, results of only (e, r, e) triples are trained.
- In *Experiment – B*, results of (e, r, e) triples and (e, r, l_n) triples are trained together without clustering.
 - In *Experiment – B1*, results of only (e, r, e) triples in *Experiment – B*
 - In *Experiment – B2*, results of only (e, r, l_n) triples in *Experiment – B*
- In *Experiment – C*, results of (e, r, e) triples and (e, r, n) triples are trained together with clustering.
 - In *Experiment – C1*, results of only (e, r, e) triples in *Experiment – C*
 - In *Experiment – C2*, results of only (e, r, l_n) triples in *Experiment – C*
- In *Experiment – D*, results of only (e, r, l_n) triples are trained without clustering.
- In *Experiment – E*, results of only (e, r, l_n) triples are trained with clustering.

As we mentioned in *Section – 4.4*, results can differ up to dataset, dataset split ratio, hardware, environment, and parameters. The experiments in *Table – C.1*, uses the same dataset with *Table – 4.2*, but the split ratio of the dataset is %80 - %20 for train and test sets respectively. Furthermore, our environment and hardware are different from previous experiments. Nevertheless, we used the same parameters for the TransE model. In this way, we got different results for the same experiments on TransE.

As we did in *Section – 4.4*, again, we preferred to evaluate the results of these experiments with the MRR metric from now on because this metric is more distinctive and consistent for these experiment results compared to others.

The variations in *Experiment – A*, *Experiment – B*, and *Experiment – C* are similar to the previous experiments *Experiment – 1*, *Experiment – 2*, *Experiment – 3*. Nevertheless, the main aim of experiments in *Table – C.1* is to show the differences between results for (e, r, e) triples and results for (e, r, l_n) and numerical triples how effects the distribution and results in each experiment. Moreover, we conducted *Experiment – D* and *Experiment – E* to show the link prediction results of (e, r, l_n) triples without (e, r, e) triples.

In *Experiment – B*, MRR value is 0.58 for whole dataset, also in *B1*, MRR value is the same, but in *B2*, MRR value is 0.14. As you can see, the data which we used for *Experiment – A* and *Experiment – B1* are the same, but results are changed, although new added data’s results are lower respectively. This proved that, changing the distribution of the data affects the embeddings. Also, with the results of the *Experiment – D*, we showed the numerical triples also effected from non-numerical triples. When we trained numerical triples with non-numerical ones (*i.e.*, *Experiment – B2*) and numerical triples without non-numerical ones (*i.e.*, *Experiment – D*), we got different results, like consequently 0.14 and 0.06 for the MRR value.

In *Experiment – C*, MRR value is 0.60 for the whole dataset, also in *C1*, MRR value is the same, but in *C2*, MRR value is 0.49. As you can see, the data which we used for *Experiment – A*, *Experiment – B1*, and *Experiment – C1* are the same, but results are changed although new added and processed data’s results are lower respectively (*i.e.*, 0.49). This proved that again, changing the distribution of the data affects the embeddings, and also the difference between *Experiment – B2* and *Experiment – C2* has shown that our approach achieved its mission and gave a considerable increase in numerical triples results. Also, with the results of the *Experiment – E*, we showed the numerical triples are also effected by non-numerical triples. When we trained clustered numerical triples with non-numerical ones (*i.e.*, *Experiment – C2*) and clustered numerical triples without non-numerical ones (*i.e.*, *Experiment – E*), we got different

results, like consequently 0.49 and 0.37 for the MRR value.

Finally, the amount of the added data is quite small, but its effect of enhancement on results is enormous. As we mentioned above, adding different types of data changes the distribution of embeddings, this changed not only because of the numerical triples embeddings but also changed the regular triples embeddings. As a result, our approach enhanced these situations and decreased the difference between non-numerical and numerical triples results by increasing the non-numerical results.



Table C.1: Detailed Experiment Result Table For TransE Model

Experiment	Sub-results	<i>MRR</i>	<i>MR</i>	<i>HITS@10</i>	<i>HITS@3</i>	<i>HIT@1</i>
Experiment - A		0.24	992.02	0.36	0.26	0.18
Experiment - B		0.58	88.80	0.77	0.65	0.47
B1	(e, r, e) in Experiment - B	0.58	84.57	0.77	0.65	0.47
B2	(e, r, l_n) in Experiment - B	0.14	518.51	0.77	0.65	0.47
Experiment - C		0.60	86.43	0.79	0.67	0.49
C1	(e, r, e) in Experiment - C	0.60	86.88	0.79	0.67	0.49
C2	(e, r, l_n) in Experiment - C	0.49	40.76	0.79	0.67	0.49
Experiment - D		0.06	666.07	0.14	0.06	0.02
Experiment - E		0.37	83.55	0.50	0.44	0.27