Generalizing Link Prediction for Information Extraction

By

Sonit Singh

A thesis submitted to Macquarie University

for the degree of Master of Research

Department of Computing

June 2017

MACQUARIE
University
SYDNEY·AUSTRALIA

Typeset in LaTeX 2_{ε}.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Sonit Singh

Acknowledgements

This thesis has been possible with the support of many people to whom I want to acknowledge for their ideas, contributions and moral support. Though, I have covered this thesis work during the span of 7 months, but the journey was remarkable and provides me opportunities to learn new theoretical concepts, new state-of-the-art techniques in the field, various tools and libraries.

First and foremost I wish to thank **Professor Mark Johnson**, my thesis advisor for his valuable guidance, support and help at every step during the thesis. I am highly indebtful to him for his valuable suggestions and discussions during meetings. He helped me to come up with the thesis topic and guided me throughout the duration of thesis. I am also thankful to him for reading various drafts of thesis and providing suggestions to improve structure of thesis and its content.

Joining of **Dr. Hegler Tissot** as a Postdoctoral Fellow at the right time, when I need valuable comments while writing this thesis was great. I am highly thankful to him for his valuable comments on various drafts of thesis and paper. Thanks **Hegler** for your efforts, suggestions and open door of your office at all times.

Apart from the thesis work, I am thankful to **Professor Mark Dras** and **Dr. Rolf Schwitter** for planning various talks in *Machine Learning Reading Group* and *Center for Language Technology Reading Group* respectively, which provided me insight of various allied research topics in the field of NLP and various machine learning and deep learning techniques.

I am very thankful to the **Department of Computing Administrative Team**, especially **Donna Hua**, **Sylvian Chow**, **Melina Chan** and **Fiona** for their help in carrying out various administrative tasks without hassles. I am very thankful to **Dat Quoc Nguyen** for providing me guidance at every step of this thesis. I am also thankful to **Paria**, **Hamed**, **Peter Anderson**, **Jonas Groswitz**, **Xavier** and **COMP HDR Group**, which makes this research journey more beautiful by getting involved in various networking events and informal talks.

I am very thankful to my loving family members, without whom I am not able to think about

getting education and to see this beautiful earth. I am thankful to **Rajvir** for her loving support and understanding.

Finally, I must acknowledge the support of **international Macquarie University Research Excellence Scholarship (iMQRES)** which helped to carry out this research.

Abstract

Information Extraction (IE) is the task of extracting from a text the entities and the relationships that hold between them, in a form that can be stored in a database called a *Knowledge Base* (KB) or *Knowledge Graph* (KG). *Link prediction*, also called as *Knowledge Base Completion*, is the task of predicting missing links in order to make KG more complete. While most of IE and link prediction models have focused on binary relationships, in the real world relationships are often *n-ary* ($n > 2$). Recently, IE algorithms have been proposed that can extract relationships of arbitrary arity, but as far as we know there is no corresponding work on link prediction involving relationships of arbitrary arity. In this thesis, we introduce the task of n-ary link prediction by proposing two different models to model n-ary relationships and two different training methods to train the proposed models. We also provide new dataset (based on Wikidata) for training and evaluating our proposed approaches. We also propose a modification in the standard evaluation criteria in order to overcome the bottleneck of huge computational complexity when working on large-scale KBs. Evaluation in terms of Mean Rank, Hits@10 and classification accuracy on tuple dataset show that our proposed approaches have the ability to generalize link prediction over tuples having arbitrary arity.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

$\mathcal{G}$ ..... Denotes a Graph

E ..... Edges of a Graph

V ..... Vertices of a Graph

k ..... k dimensional embedding vector

η ..... Learning rate

γ ..... margin

L_2 ..... L2 distance as dissimilarity measure

E ..... Entities in a knowledge base

R ..... Relations in a knowledge base

$\mathbb{R}^k$ ..... k dimensional vector space

h ..... Denotes head entity in a triple

r ..... Denotes relation in a triple

t ..... Denotes tail entity in a triple

m ..... Denotes number of entities sampled from entire set of entities

AI ..... Artificial Intelligence

IE ..... Information Extraction

IR ..... Information Retrieval

KB ..... Knowledge Base

KG ..... Knowledge Graph

KBC ..... Knowledge Base Completion

RDF ..... Resource Description Framework

SRL ..... Statistical Relational Learning

NLP ..... Natural Language Processing

$PMKB$ ..... Precision Medical Knowledge Base

EHR ..... Electronic Health Records

IoT ..... Internet of Things

NER ..... Named Entity Recognition

SGD ..... Stochastic Gradient Descent Optimizer

$L-BFGS$ .. Limited-memory Broyden-Fletcher-Goldfarb-Shanno optimizer

$ADADELTA$ ADADELTA Optimizer

$ADAGRAD$. Adaptive Gradient Optimizer

$ADAM$ ..... ADAM Optimizer

$YAGO$ ..... Yet Another Great Ontology

$HeadMR$ ... Mean Rank for head side

$TailMR$ .... Mean Rank for tail side

$HeadH10$ ... Hits@10 for head side

$TailH10$ .... Hits@10 for tail side

MR ..... Overall Mean Rank of the dataset

$H10$ ..... Overall Hits@10 score of the dataset

$MaxEnt$ .... Maximum Entropy Classifier

CRF ..... Conditional Random Fields

SVM ..... Support Vector Machine

NMM ..... Neighborhood Mixture Model

PRA ..... Path Ranking Algorithm

$Freebase$ ... Freebase, a knowledge base

$NELL$ ..... Never Ending Language Learning

Language is a process of free creation; *its laws and principles of generation are used is free and infinitely varied. Even the interpretation and use of words involves a process of free creation.* [4]

Noam Chomsky, American Linguist

1 Introduction

1.1 Overview of Knowledge Bases (KBs)

Natural Language Processing (NLP) is the branch of *Artificial Intelligence* (AI) that deals with analyzing, understanding and generating the languages that humans use naturally in order to interface with computers in both written and spoken contexts. *Information Extraction* (IE), one of the major task of NLP, is the process of extracting facts (about the world) from from text information. Therefore, IE is the task of extracting from a text the entities and the relationships that hold between them, in a form that can be stored in a database called a *Knowledge Base* (KB) or *Knowledge Graph* (KG). These KBs are special kind of relational databases especially designed for knowledge management, collection and retrieval [5]. KBs are special in the sense that they capture human knowledge and places it into a system where they are used to solve complex problems normally requiring a high level of human expertise. KBs store real-world information in the form of entities and the relationships between them in structured form. These structured KBs provides easy reasoning ability and can be used for inference. This motivation has lead to built various KBs/KGs such as *Freebase* [6], *Wikidata* [7], *YAGO* [8] , *DBpedia* [9],

NELL [10] and *Google Knowledge Vault* [11].

KBs are collection of facts about people, places or things, and their relationship between them. These facts are generally stored in the form of *triples* (head entity, relation, tail entity) *i.e.*, (h, r, t), which are particular case of *tuples* where tuple length can be any arbitrary length in the form of $(h, r1, t1, \ldots, rn, tn)$. Figure 1.1 shows an example of sample knowledge graph where entities are represented as nodes in the graph and relationship between two entities is represented as an edge. Representing worldly knowledge in the form of triples (specially in the form of simple graph), is not able to express complete information. For example given the fact X, there is loss of information while representing this fact in the form of triples, because creating a direct relationship from the actor/actress to the award won means that there is no way to work out for which movie they have won award for.

X: *"Lauren Bacall was nominated for Academy Award for Best Actress in Supporting Role in The Mirror has 2 Faces"*

Hence, representing facts in the form of triples have limitations in terms that they are not able to model the long relationship between entities. In order to overcome this limitation, we represent these knowledge graphs in the form of a *hypergraph* in which a relationship (called hyperedge) can connect any number of given nodes. While simple graph permits a relationship to have only one start node and one end node, the hypergraph allows any number of nodes at either end of a relationship. Hence, hypergraphs are useful for modeling tuples having *n-ary relationships*. Hypergraphs are more accurate, are information rich and are multi-dimensional, hence they are more generalized than simple graphs. Figure 1.2 shows that we can represent the fact X along with other facts in the form of a hypergraph which is able to express complete information in the fact and is able to take account of long relationship between two entities. This thesis is about extending techniques that are used for triple-based KBs to tuple based KBs.

1.2 Applications of KBs

There has been much attention in the research community to construct large-scale KBs due to its varied applications in terms of enriching search engines [12], training relation extractors and semantic parsers [13], [14],[15], and in question answering [16]. Information Retrieval (IR), which refers to various techniques of searching of documents, searching specific information within documents, searching for meta-data and searching within databases corresponding to user's queries. Search engines are one of the main contribution of IR technology. KBs present structured data to the search engines in order to make it easy for the search engines to better

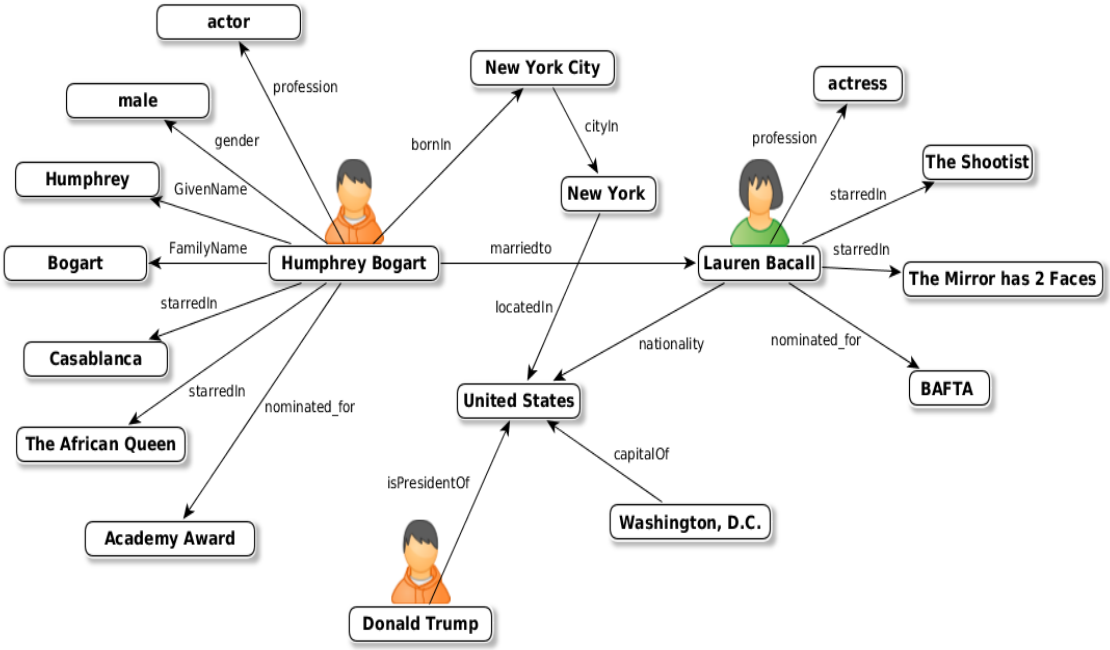

Figure 1.1: An example of sample Knowledge Graph.

understand the content and context of the web pages [17], [18]. Search engines use KBs (form of structured data) to generate rich snippets, which are small pieces of information that appear in search results. Search engines have limitations in the sense that they provide only documents but no specific answers to the query. Recently, NLP research community has moved towards integrating Information Extraction technology with Information Retrieval technology to make search engines more reliable, accurate, specific to user queries and having high semantic understanding. Though, user queries can be answered as Question Answering problem where we can provide direct answer to user queries. In order to accomplish this goal, there is a need to have highly structured KBs and better KB inference techniques.

With the increasing use of digital assistants (Apple Siri, Microsoft Cortana and Amazon Echo) and open ended show quiz (IBM Watson), research is in the direction to develop open ended question answering systems [19],[20], [21],[22]. Question Answering systems, which refers automatically answer questions posed by humans in natural language query. The goal of Knowledge Base question answering system is to automatically return answers from the KB given a natural language questions. There are two mainstream research directions for this task. One based on semantic parsing focus on constructing a semantic parser that could convert natural language questions into structured expressions like logical forms. On the other hand, information retrieval methods are more like to search answers from the KB based on the information conveyed in the questions. Ranking techniques are employed to score the answers

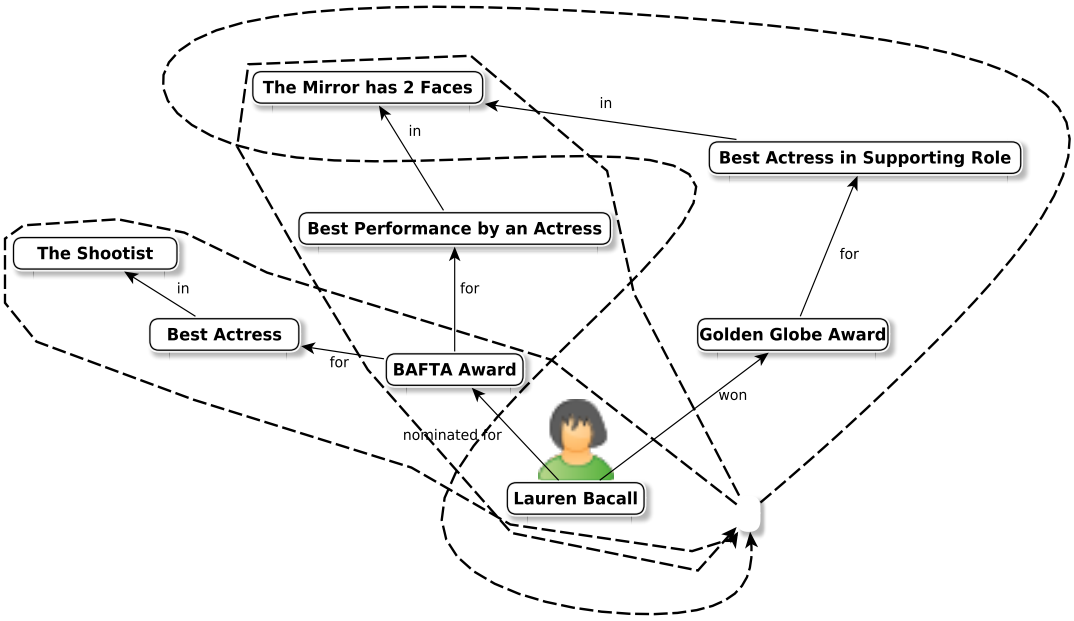

Figure 1.2: Representing facts in the form of a hypergraph.

and to select the correct one from top scored answers [23].

Relation extraction systems use KBs as a source of training data. The relation extraction task focuses on extracting individual mentions of relations from text. One key aspect of every relation extractor is how to annotate training and test data in order to leverage advantage of machine learning approaches. Recently, distant supervision or self-supervised learning [24],[25],[26],[27],[28] has become an important technique which exploit large KBs (such as Freebase, Wikidata or DBpedia) to automatically label entities in text and use the annotated text to extract features and train a classifier [29],[30],[31],[32]. Distant supervision, which is a semi-supervised learning algorithm expecially used in relation extraction that make use of weakly labelled training set (based on a heuristic labeling function) typically relying on a KB. Moreover, KBs helps in Named Entity Linking, also called as Named Entity Disambiguation or Named Entity Normalization, which is the task of identifying the entity that corresponds to a particular occurrence of a noun in a text document. References to entities in natural text is quite ambiguous, because particular entity can refer to many mentions and on the other hand, particular mention can refer to many entities. For example, "Paris" can refer to entity mention location (GPE) *i.e.* city (Capital of France), or it can refer to entity mention person (PER) *i.e.* Paris Hinton (an actress). Hence, entity linking is the task of resolving named entity mentions to entries in a structured KB. The similarity measure between the text mention and an entity candidate in KB helps to disambiguate named entities [33],[34],[35].

Knowledge bases also plays vital role in semantic parsing, which is mapping of text to a meaningful representation. In other words, semantic parsing is concerned with translating language utterances into executable logical forms [16]. Fundamental challenge in using semantic parsers is getting annotated data for training and extending it to new domains [36]. To overcome this limitation, KBs acts as training data in order to train semantic parsers [37], [38].

Recent interest is to use KBs for understanding complex diseases and for advancing precision medicine [39]. Researchers are developing Precision Medicine Knowledge Bases (PMKBs) [40] to pioneer a new model of personalized, individual, patient-powered research and treatment. With rise of Electronic Health Records (EHRs), Internet of Things (IoT), big data technology and building large scale PMKBs, it gives greater variety of characteristics to better understand which types of treatments work best for more specific subsets of population based on factors including health condition, environment and lifestyle [39]. Hence, PMKBs could mean personalized, individualized research and treatment that empowers patients [41], [42].

1.3 Link prediction in Knowledge Bases

Although KBs can be extremely large, containing billions of facts, million of entities and thousands of relations, but they are far from complete [43],[44], [45]. For *e.g.*, about 75% of people in Freebase do not have their nationalities and 70% of people in Wikidata do not have their known place of birth [11]. Also, it is impossible to put all entities, facts or relations into the KB while it is manually curated in the beginning. In order to overcome this missing information, researchers [46], [47], [48], [49], [43], [50], [51], [52], [53] have formulated ways by which we can do reasoning over the facts already existing in the KB to automatically derive new facts. The task of predicting missing entries in the KBs is called as *link prediction*, *knowledge base completion* or *knowledge graph inference* [54],[45],[46]. Link prediction can also be seen as *graph completion problem*, where we need to find missing links between various nodes (*i.e.* entities). Link prediction is one of the main problems in Statistical Relational Learning (SRL) [55].

Generally, KBs represent facts in the form of *triples* like *(h,r, t)*. Hence, link prediction can be formalized as filling in the missing or incomplete triples like *(h,r,?)* or *(?,r,t)* by predicting head or tail, by reasoning over existing triples in the KB [56]. Based on the example given in Figure 1.1, we would likely able to predict new facts (or links) such as *(Humphrey Bogart, nationality, United States)*, *(Humphrey Bogart, wins, Academy Award)*, given the facts that *(Humphrey Bogart, bornIn, New York City)*, *(Lauren Bacall, wins, BAFTA)*, *(Lauren Bacall,*

Subject	Predicate	Object
(Humprey Bogart,	nominated_for,	Academy Award)
(Humprey Bogart,	for,	Best Actor)
(Humprey Bogart,	in,	Casablanca)
(Lauren Bacall,	nominated_for,	Academy Award)
(Lauren Bacall,	for,	Best Actress)
(Lauren Bacall,	for,	2 Faces)

Table 1.1: Representing information in the form of triples.

nominate_for, Academy Award) and *(Humphrey Bogart, wins, BAFTA)*, because it is very likely that *Academy Award* and *BAFTA* are given to *movie stars* and *movie stars* have profession as *Actor/Actress*.

Link prediction is widely used in various NLP applications as discussed in section 1.2. Hence, correctness and completeness of KBs can improve these applications. Therefore, it is necessary to develop *link prediction* methods which find missing relationships in the KBs [57].

1.4 From Triples to Tuples

Completeness, *accuracy* and *data quality* are important parameters that defines the usability of KBs [45]. Various NLP applications are dependent upon KBs, so any deficiency in KBs leads to poor performance in the applications it leads to. Majority of existing KBs adopt *Resource Description Framework* (RDF) as their representation, which is the Web standard for expressing information about entities. Hence, KBs represent facts in the form of <*s,p,o*>, where *s*, *p* and *o* denotes subject, predicate and object respectively. Here subject and object are entities and predicate is relationship that holds between two entities. Note that <*s,p,o*> notation is same as *(h,r,t)* which are ways of denoting triples in the KB. For instance,

Example X: **Humprey Bogart was nominated for Academy Award for Best Actor in Casablanca**

Example Y: **Lauren Bacall was nominated for Academy Award for Best Actress in 2 Faces**

The information extracted from Examples X and Y can be represented in the form of triples as shown in Table 1.1.

Though modern KBs are based on triples, but real world facts are *tuples* and can have any arbitrary length, denoted by *(h,r1,t1,. . . , rn ,tn)*. Examples X and Y can be represented in the form of tuples as shown in Table 1.2

As seen in Table 1.1, the information that Bogart is nominated for Academy Award for his role in Casablanca is lost. But, when we represent the same information using tuples as

head	rel1	tail1	rel2	tail2	rel3	tail3
Bogart,	nominated_for	Academy Award	for	Best Actor	in	Casablanca
Bacall,	nominated_for	Academy Award	for	Best Actress	in	2 Faces

Table 1.2: Representing information in the form of tuples.

shown in Table 1.2, it holds information that Bogart is nominated for Academy Award for his role in Casablanca. Hence, tuples are more expressive than triples. Also, tuples hold complete information and takes longer context which can be beneficial for reasoning over KBs.

With the need to have modern question answering systems and web search, there is growing research in the direction of extracting *n-ary relations* where *(n>2)* [58]. Link prediction plays a similar role in relation extraction, what a language model plays role in speech recognition. Hence, link prediction and relation extraction are complementary to each other. Although in the real world relationships are often *n-ary* ($n > 2$) and *Information Extraction* algorithms that can identify relationships of arbitrary arity have been recently proposed [58], [59],[60],[61], most link prediction or KB completion models have focused on binary relationships. However, as far as we know there is no corresponding work on knowledge base completion involving relationships of arbitrary arity. Therefore, *n-ary link prediction* can be highly useful for improving accuracy of n-ary relation extraction. These reasons motivated us to look KBs in the form of tuples and to have reasoning over tuples in order to make KBs more complete.

1.5 Problem Description

State-of-the-art link prediction models such as TransE [46], SE [47], RESCAL [48], TRESCAL [49], NTN [43], TransH [50], TransR [51], TransG [52], STransE [53] work on *triples*, but real-world data is in the form of *n-ary relations* of arbitrary length denoted as *tuples*. Lot of research has already been done in developing triples based KBs and triple based link prediction models. Recently, research efforts are towards representing and extracting facts in the form of tuples. Representing any fact in the form of tuple takes into account of all entities and relations present in the original fact present in a text. Therefore, tuples hold semantics of the fact that is present in the text, in turn converting it to structured form which we can store in the form of a KB. Representing worldly facts in the form of tuples, instead of triples provides advantages in terms of more expressive in nature, takes context into account as well as hold semantics of the original fact in the form of natural language text.

In triple-based KBs, each fact denoted as (h, r, t), there is one head and one tail. On the other hand, fact in a tuple denoted as $(h, r1, t1, \ldots, rn, tn)$, there is one head and can have multiple

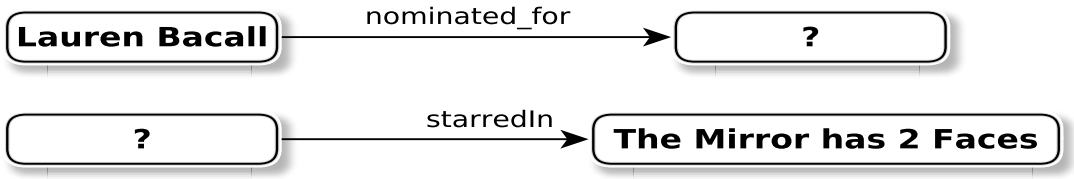

Figure 1.3: Triples based knowledge base completion. Upper example illustrates missing tail and lower one illustrates missing head in triples.

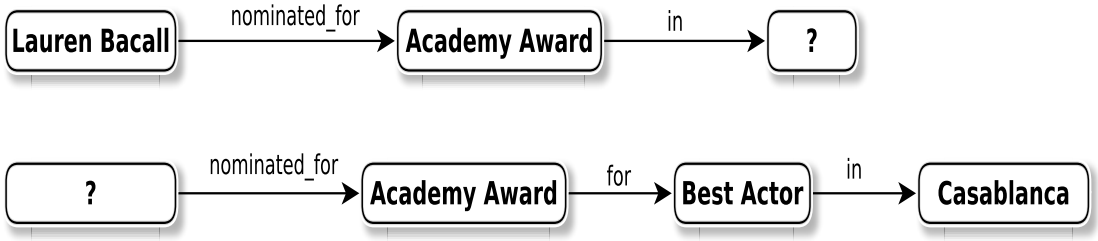

Figure 1.4: Tuples based knowledge base completion. Upper example illustrates missing tail and lower one illustrates missing head in tuples.

tails. For example, tuple of length 3 have one head and one tail, tuple of length 5 have one head and two tails, and so on. Triple based link prediction models predict missing head or tail, of given fact in the form of triple. For example, Figure 1.3 shows triples based knowledge base completion. But, we are interested in predicting missing head or tail, of given fact in the form of tuple. Figure 1.4 shows example of predicting missing tail or missing head in a given tuple. In this thesis, we aim at predicting missing values in n-ary relationships (also called tuples, or tuples having arbitrary arity). We focus on predicting a single missing entity either *head* or *tail* in an n-ary relationship tuple. The task of *n-ary link prediction* is useful for many purposes, including for improving the accuracy of an IE system where n-ary link prediction can identify implausible relations as likely extraction errors, and improving performance of open-ended question answering systems. So we aim at generalizing link prediction over tuples having arbitrary arity.

1.6 Objectives

The objectives of this thesis are identified as follows:

1. To propose the new task of knowledge graph completion for arbitrary arity relations (also called as tuples, or n-ary relationships).

   Lot of work has been done for embedding triple based KGs and link prediction over triple

based KGs, but no work has been towards link prediction over tuples. In this thesis, we proposed this new task as hypergraph completion problem because hypergraphs have ability to model n-ary relationships. The overall goal of this proposed task is to generalize link prediction so that it can take tuples of arbitrary arity into its account. Our general approach to hypergraph completion is to reduce a tuple to a set of triples, so we can then use any triple based graph completion algorithm.

2. To propose methods by which we can model n-ary relationships in a KG.

   In order to generalize link prediction and to model n-ary relationships, we proposed two models by which we are reducing tuples into triples, namely *The head-triple reduction model* and *The clique reduction model*. These models differs in the way we reduce tuples into triples. We also proposed two different training methods by which we train our two proposed models.

3. To provide tuple dataset in order to evaluate proposed approaches for modeling n-ary relationships.

   There are many standard datasets for triple prediction, but there is no dataset for tuple prediction. So we created our own dataset from latest Wikidata dump as on date 3^{rd} *January, 2017*. The dataset contains tuples in the form of *(h, r1, t1, . . . , rn,tn)*, where *h* denotes head entity, *r1* denotes first relation, *t1* denotes first tail entity, and so on. Our tuple dataset containing $102,231$ entities, and 353 relations is quite large compared to standard triple based datasets.

4. To propose modification in the existing triple based link prediction evaluation protocol for tuples and to scale it for large-scale KGs.

   There are standard evaluation metrics to evaluate triples based link prediction models. First, the evaluation metrics proposed to evaluate tuple based link prediction models are direct extensions of metrics used to evaluate triple based models. Second, link prediction task involves evaluating all possible triples that are formed by substituting all entities in the dataset, which becomes computationally expensive when the set of entities becomes very large, as it is in our dataset. So, we have also proposed modification in the evaluation criteria in order to scale it to large KGs.

1.7 Organization of Thesis

This thesis is structured as follows:

Chapter 2 provides brief literature review of Statistical Relational Learning, provides an overview of embedding triples based KBs, discussed state-of-the-art link prediction models for triple based KBs, their evaluation criteria and their comparative analysis.

Chapter 3 propose two different ways of modeling n-ary relations as a way of reducing tuples to triples namely, *The head-triple reduction model* and *The clique reduction model*. Two different methods of training the proposed models namely, *triples based training* and *tuples based training* are also proposed in this chapter.

Chapter 4 propose modification to existing link prediction evaluation protocol for evaluating large-scale KBs, provides statistics of tuple dataset generated from Wikidata. This chapter provides empirical evaluation of 4 proposed approaches (2 proposed models coupled with 2 proposed training methods) on tuple dataset and detailed analysis of the results.

Finally, Chapter 5 discusses the contributions of this thesis and important conclusions, highlights limitations of this work and provides directions for future research.

Research is to see what everybody else has seen, and to think what nobody else has thought

Albert Szent-Gyorgyi

2 Literature Review

This chapter describes the background knowledge that is necessary to further understand the concepts and proposed approaches introduced in this thesis. Section 2.1 gives overview of knowledge representation in the form of graphs and section 2.2 highlights link prediction in triple-based knowledge bases. Section 2.3 provides detailed analysis of various triple-based embedding models, including *TransE* model. Section 2.6 gives overview of standard evaluation metrics for triple-based models. Finally, chapter concludes with current research in *n-ary relation extraction* and the need to have *n-ary link prediction*. We discuss *TransE* model, a triple-based embedding model for link prediction in KBs in detail, because we want to generalize *TransE* model over our tuple dataset.

2.1 Representation and Reasoning of Knowledge with Graphs

With the rise in technology, humans are generating vast amount of data. The goal of Artificial Intelligence (AI) is to built intelligent machines that can make intelligent use of this vast amount of generated data. One of the most important and interesting area where every company is currently focusing on is how to do knowledge aggregation, knowledge representation and do

Subject	Predicate	Object
Humphrey Bogart	won	Academy Award
Humphrey Bogart	gender	male
Humphrey Bogart	nationality	American
Humphrey Bogart	nominated_for	BAFTA Award
Lauren Bacall	profession	actress
Lauren Bacall	gender	female
Lauren Bacall	nationality	American
Lauren Bacall	nominated_for	BAFTA Award

Table 2.1: Example RDF graph in triple representation.

reasoning over data collected from various sources[62]. Information Extraction (IE) technologies helps to efficiently and effectively analyze free text and to discover valuable and relevant knowledge from it in the form of structured information [63]. The input to IE system is a collection of documents (email, web pages, news groups, and so on) and output is a representation of the relevant information from the source document according to some specific criteria [64]. Hence, the goal of IE is to extract salient facts about pre-specified types of events, entities, or relationships, in order to build more meaningful, rich representations of their semantic content, which can be used to populate databases that provide more structured input [63].

Most of the real-world data we see around us is inherently relational, whether it is social networks, gene-protein interactions, hyper-linking pages for world wide web, clustering of documents based on particular topic, author-publication citations, large-scale networks as well as semantic web and related standards [65]. Graphs have ability to represent this inherently relational data in such a way that knowledge based reasoning becomes possible. Current Semantic Web technologies represent knowledge in various forms, most prevalent among which is semantic networks. These semantic networks structures can be found in Resource Description Framework (RDF) graph representation, whose nodes represents entities (or concepts) and arcs represents relations between those entities. Hence, facts in RDF have the form *(subject, predicate, object)*, which is also referred to as RDF triples. Most of the KGs store facts in the form of triples which can naturally be viewed as tables with more than 2 columns. Table 2.1 shows RDF graph in the form of triples and Figure 2.1 shows its corresponding graphical representation.

2.2 Link Prediction in Triple based KGs/KBs

Curated KBs typically have high precision, but suffer from a lack of coverage. In order to automatically increase coverage by predicting missing entries, researchers have developed methods

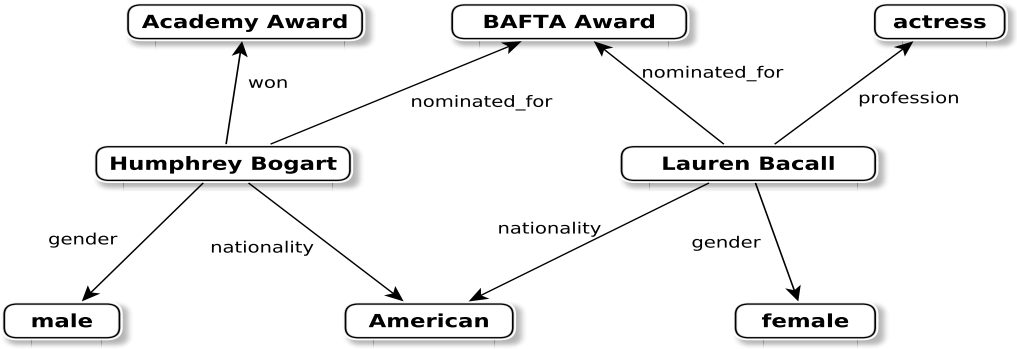

Figure 2.1: Example RDF graph from Table 2.1 in graphical representation.

[66], [67]. These methods are commonly categorized under the heading of Knowledge Base Completion (KBC) or link prediction in KBs, which is one of the main problems in Statistical Relational Learning [68]. Apart from this, there is lot of redundancy among relations that are existing in the KBs. This natural redundancy among relations often make it possible to fill in the missing entries of a KB [67]. For example, it is not possible to record awards won by all movie stars, but it can be inferred easily if we know the facts that movie stars are very likely to win Academy Award, BAFTA or Golden Globe award, compared to Pulitzer prize or Nobel prize. Also, we know the relation *PlaceOfBirth*, it is very likely to predict *Nationality* of someone, which might earlier be missing in the KB.

Link prediction in KBs is very similar to recommendation systems and finding missing link between users in social networks. In recommendation systems, goal is to predict the rating of the movies which are not already rated and recommending it to users to have better user experience. Similarly, in Knowledge Base Completion, the goal is to check whether a particular triple not in the KB is likely to be true or not [47], [69]. As an example, in Figure 2.2, the task of link prediction is to find the *profession* of *Humphrey Bogart* from the facts already exiting in the KB. Hence, link prediction in KBs is one attempt to mitigate the problem of knowledge sparsity and to make them more robust and complete so that they can be effectively used for various downstream applications, including question-answering, web/mobile search, social media analysis, recommendation systems, co-reference resolution, information retrieval and semantic parsing.

2.3 Approaches for link prediction in Knowledge Bases

Link prediction is quite well formulated problem in Statistical Relational Learning, also called as multi-relational learning, which refers to the set of methods that are used for statistical analysis

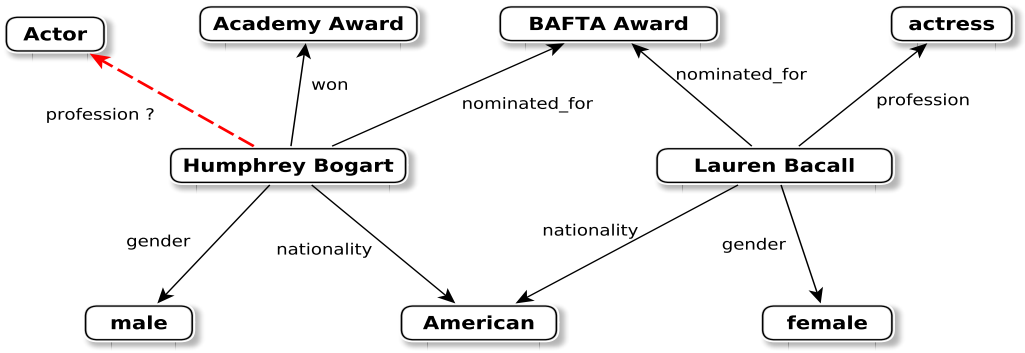

Figure 2.2: Link prediction in triple based KBs. Dotted line shows link to predict.

of relational or graph-structured data [45]. Early models for link prediction in KGs are based on *Probabilistic models*, also called as *Probabilistic Latent Factor models*, which conditions the probability distribution of the relations between two entities based on the latent factors of entities [3]. Probabilistic models have limitations in terms of scaling to large KGs because of the complexity of the probabilistic inference and learning [70], [3].

One of the most promising approach to link prediction in KGs is to embed entities and relations in a low-dimensional vector space. Representing KGs in low-dimensional vector space helps to learn *continuous*, *low-dimensional* representation (also called *embeddings*) for entities and relations between them. These embeddings captures the semantics of entities and relations in the knowledge graph. Representing embeddings in low-dimensional vector space not only leads to lower model complexities, but also reduced run time and low memory load.

2.3.1 Embedding based models

In recent years, lot of interest has risen in learning *low-dimensional embeddings* for representing language, entities and relations inspired by *word2vec* model [71]. The *word2vec* model learns *dense vector representation* of words that captures the semantic meaning of words and is useful for wide range of NLP applications. The most amazing property of word embeddings is that they capture the laws of analogy. Figure 2.3 represents four words in the continuous vector space and it shows that word2vec model is able to perform very well on this word analogy task [72], [73], for example

man is to *woman* as *king* is to –?–

walked is to *walking* as *swam* is to –?–

Sydney is to *Australia* as *Tokyo* is to -?–

Word embeddings is one of the major success in unsupervised learning which makes NLP

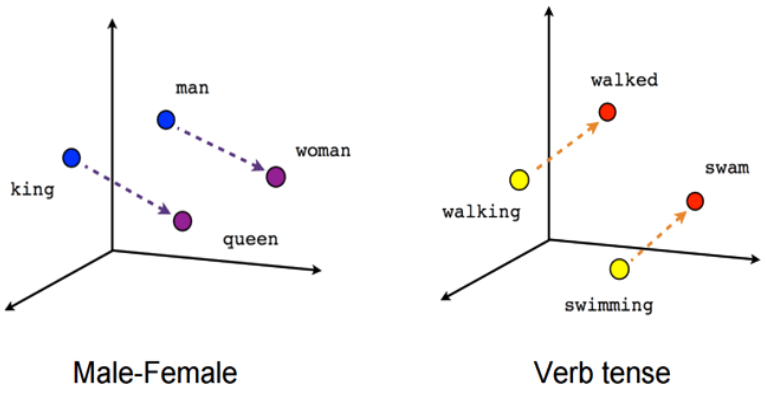

Figure 2.3: An example of word embeddings [1].

breakout. The word2vec model learns the embeddings from the given words in a way that, it gives

$$v_{king} - v_{man} + v_{woman} \approx v_{queen}$$

Generally, knowledge graph embed entities *h and/or t* and relations *r* as a *k*-dimensional vector. The embedding representation of each entity and relation in the KG encodes the global information, since the representation is obtained by minimizing a global loss function involving all entities and relations [2]. The plausibility of the triplet *(h, r, t)* is measured by a score function $f_r(h,t)$.

Let us denote a knowledge graph $\mathcal{G}$ having $\mathcal{E}$ set of entities and $\mathcal{R}$ set of relations. The knowledge graph $\mathcal{G}$ consists of true triplets (or facts) denoted as (h,r,t), where $(h,t) \in \mathcal{E}$ and $r \in \mathcal{R}$. The *embedding model* learns the embeddings of entities and relations in a knowledge graph, in a way that the score of a true triplet denoted by $f_r(h,t)$ is smaller than the score of false (or negative) triplet denoted by $f_r(h',t')$, where $(h,r,t) \in \mathcal{G}$ and $(h',r,t') \in \mathcal{G}'$. Various approaches have been explored for knowledge graph embedding which are described in the following sections.

Unstructured Model (UM): The Unstructured Model [74] considers graph as a mono-relational and sets all translations $r = 0$, hence the score function of UM model is $f_r(h,t) = \|h - t\|$. As UM model does not take relations r into account, it cannot distinguish different relations which limits its use for link prediction [50], [51], [52], [53], [75].

Structured Embeddings (SE) model: The SE model [47] is based on two relation specific matrices, $M_{r,1}$ for head entities and $M_{r,2}$ for tail entities, having score function as $f_r(h,t) = |M_{r,1}h - M_{r,2}t|$. Since, the SE model has two separate matrices to optimize, it cannot capture precise relation between entities and relations. The SE model extends the UM model by assuming that the head and tail entities are similar only in a relation-dependent subspace, hence have two

different matrices [50], [51], [52], [53], [75].

Neural Tensor Network (NTN) model: The score function of NTN model [43] is as $u_r^t g(h_t W_r t + W_{rh} h + W_{rt} t + b_r)$, where u_r is a relation-specific linear layer, g(.) denotes *tanh* operation, $W_r \in \mathbb{R}^{d \times d \times k}$, a 3-way tensor and $W_{rh}, W_{rt} \in \mathbb{R}^{k \times d}$, weight matrices. Though NTN model has high expressiveness, but has limitations in terms of having high complexity due to large number of network parameters which limits its use for large-scale KGs [50], [51], [52], [53], [75].

Latent Factor Model (LFM) model: LFM model [76] falls under the category of *bilinear* models which considers second-order correlations between entity embeddings using a quadratic form, and hence have scores function defined as $f_r(h,t) = h^T M_r t$. The bilinear model has its improved performance over SE and other models, but it is restricted to model linear interactions only. Also, bilinear model lacks its expressive power and requires lot of parameters [50], [51], [52], [53], [75].

TransE model: The TransE model [46] is one of the most promising embedding models as it is simple, efficient, while achieving state-of-the-art link prediction results. In TransE, both entities (h and t) as well as relation (r) are represented with k-dimensional vectors such that $(h,t) \in \mathbb{R}^k$ and $r \in \mathbb{R}^k$. These vectors are chosen such that for each triple (h,r,t), it maintains the relation:

$$h + r \approx t \tag{2.1}$$

The score function of TransE model is the norm of this translation as given:

$$f_r(h,t) = \|h + r - t\|_2^2 \tag{2.2}$$

The score function $f_r(h,t)$ is chosen such that the score for the positive triple $(h,r,t) \in \mathcal{G}$ is lower than the score for the corrupted triple $(h^t, r, t^t) \in \mathcal{G}^t$. In order to learn model parameters, margin-based objective function is minimized as follows:

$$\mathcal{L} = \sum_{\substack{(h,r,t) \in \mathcal{G} \\ (h^t,r,t^t) \in \mathcal{G}^t_{(h,r,t)}}} [\gamma + f_r(h,t) - f_r(h^t,t^t)]_+ \tag{2.3}$$

where $[x]_+ = max(0,x)$, γ is the margin hyper-parameter, $\mathcal{G}$ denotes training set of correct triples, $\mathcal{G}^t$ denotes set of incorrect triples that are formed by corrupting either h or t from the correct triples $(h,r,t) \in \mathcal{G}$. *"Bernoulli"* trick is applied to choose whether head or tail is to be corrupted from entire set of entities to form negative (or incorrect) triples [77]. Figure 2.4

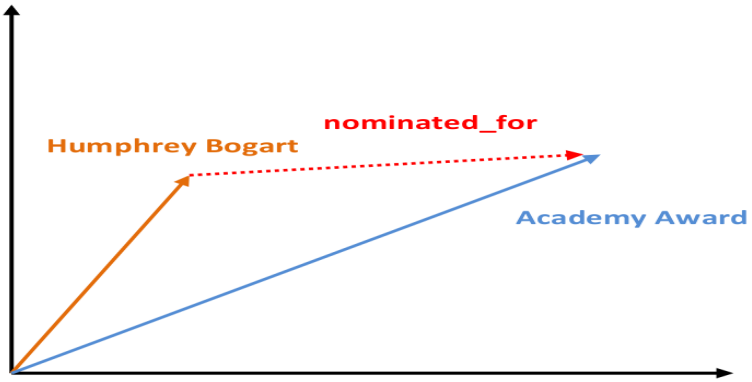

Figure 2.4: TransE embedding model [Adapted from [2]].

shows representation of triple in the embedding vector space for the triple *(Humphrey bogart, nominated_for, Academy Award)*. In this example, TransE model learns the embedding vectors of entities and relations in such as way that $v_{HumphreyBogart} + v_{nominatedfor} - v_{AcademyAward} \approx 0$. See algorithm 1 for detailed steps in TransE model.

Though TransE model is quite simple and highly effective, it has good performance for 1-to-1 relations, but has issues when modeling 1-to-M, M-to-1 and M-to-M relations [50], [51], [52], [53], [75]. From 1-to-1 relations, we mean that for a particular relation in triples-based KB, there is only one head and one tail attached to that particular relation. On the other hand, when for a particular relation it has only one head and multiple tails (1-to-M), multiple heads and only one tail (M-to-1) and multiple heads and multiple tails (M-to-M), the TransE models has limitations to such cases [50], [51], [52], [53], [75].

TransH model: To overcome the limitations of TransE model in terms of modeling 1-to-M, M-to-1 and M-to-M relations, [50] proposed TransH model which associates each relation with a relation specific hyperplane and uses a projection vector to project entity vectors onto that hyperplane. For a triple (h, r, t), the entity embeddings h and t are first projected to the hyperplane of w_r, denoted as $h_\perp$ and $t_\perp$. Hence, the score function is defined as $f_r(h, t) = |h_\perp + r - t_\perp|_2^2$. Restricting $|w_r|_2 = 1$, we get $h_\perp = h - w_r^T h w_r$ and $t_\perp = t - w_r^T t w_r$. By projecting entity embeddings into relation hyperplanes, it allows entities playing different roles in different relations [50], [51], [52], [53], [75].

TransR model: TransE and TransH model assume embeddings of entities and relations in the same space $\mathbb{R}^k$. An entity can have various semantic meanings when attached to different relations in a particular KB. To allow same entity playing different roles in different relations, TransR model [51] embeds entities and relations in separate space, *i.e. entity space* and *relation spaces* (relation-specific entity spaces), and performs translation in the corresponding relation space [50], [51], [52], [53], [75].

Algorithm 1 TransE Model Algorithm [46]

Input

Training set $S = \{(h, r, t)\}$, entity set E, relation set R, margin γ, and embedding dimension k

initialize $r \leftarrow$ uniform $\left(\frac{-6}{\sqrt{k}}, \frac{6}{\sqrt{k}}\right)$ for each $r \in L$
$r \leftarrow r/\|r\|$ for each $r \in L$
$e \leftarrow$ uniform $\left(\frac{-6}{\sqrt{k}}, \frac{6}{\sqrt{k}}\right)$ for each $e \in E$
loop
$e \leftarrow e/\|e\|$ for each $e \in E$
$S_{batch} \leftarrow$ sample (S, b)
$T_{batch} \leftarrow \phi$ //initialize a set of pair of triplets
for $(h, r, t) \in S_{batch}$ **do**
  $(h^t, r, t^t) \leftarrow$ sample $(S^t_{(h,r,t)})$ //sample a corrupted triplet
  $T_{batch} \leftarrow T_{batch} \cup \left\{\left((h, r, t), (h^t, r, t^t)\right)\right\}$
end for
Update embeddings w.r.t
$\sum_{\left((h,r,t),(h^t,r,t^t)\right) \in T_{batch}} \nabla[\gamma + d(h + r, t) - d(h^t + r, t^t)]_+$
end loop

2.4 Path based embedding models

In order to take rich context of information and to improve existing embedding models, relation path between entities has been taken into account. Path based models [78], [79], [80], [81], [82], [83], [84], [85], [86], [87] represent a relation path by a vector which is the composition of vectors of all relations in the path. Various compositions tried are sum operation, multiplication and other non-linear operations. PTransE [87] is path-based model which uses path information to find the missing facts in the KG. PTransE doubles the number of edges in the KG by creating reverse relationships for every existing relationship in the KG. [84] proposed Path Ranking Algorithm (PRA) based on the technique of random walk inference. [83] uses PRA algorithm which takes random walks from head to tail and predicts unseen relationship between head and tail. TransE-NMM [85] takes nearest neighbors of entities (both head and tail) into account to improve link prediction results. Table 2.2 provides the comparison of various prominent knowledge base embedding models in terms of their score function, required number of parameters to learn and optimization algorithm used.

Model	Score Function	Parameters	Op. Algo
Unstructured [74]	$\|\text{h-t}\|_2^2$	$O(n_e k)$	SGD
SE [47]	$\|W_{r,1}h\text{-}W_{r,2}t\|_{\ell/2}$	$O(n_e k)$	SGD
TransE [46]	$\|\text{h+r-t}\|_2^2$	$O(n_e k + n_r k)$	SGD
TransH [50]	$\|(\text{h-}w_r^T \text{h} w_r)\text{+ } d_r \text{ -(t-}w_r^T \text{t} w_r)\|_2^2$	$O(n_e k + 2n_r k)$	SGD
TransR [51]	$\|W_{r,1}h + \text{r - } W_{r,2}t\|_{\ell/2}$	$O(n_e k + n_r k + n_r k^2)$	SGD
Single Layer	$u_r^t g(W_{rh}h + W_{rt}t + b_r)$	$O(n_e k + n_r(sk + s))$	AdaGrad
NTN [43]	$u_r^t g(h_t W_r t + W_{rh}h + W_{rt}t + b_r)$	$O(n_e k + n_r(sk^2 + 2sk + 2s))$	L-BFGS
STransE [53]	$\|W_{r,1}h + \text{r -}W_{r,2}t\|_{\ell/2}$	$O(n_e k + n_r k)$	SGD
Bilinear Model [76]	$h^t W_r t$	$O(n_e k + n_r k + 10k^2)$	AdaGrad

Table 2.2: Comparison of various KB embedding models based on: (1). Score function $f(h, r, t)$, (2). Parameters required, (3). Optimization methods used. k is the dimension of embedding space, O represents number of parameters, s is the hidden nodes of neural network or slices of a tensor, n_e represents the number of unique entities and n_r represents the number of unique relations. h denotes head, r denotes relation and t denotes tail of triple.

2.5 Optimization algorithms and their effect on KB completion models

Optimization algorithms plays central role in training link prediction models and they may require large amount of time for converging to optimal solution. Optimization algorithms minimize the objective function to learn various model parameters in the form of learning entity embeddings, relation embeddings or matrices. Various optimization algorithms, including Stochastic Gradient Descent (SGD), AdaGrad [88], AdaDelta [89], L-BFGS [90], Adam [91] are used for optimizing objective function in various link prediction models proposed in the literature. SGD have limitations in terms of initialization and properly tuning of the learning rate η [92]. Hence, non-adaptive algorithms require longer time to learn. Moreover, if part of data is infrequent, of which parameters need to be updated (or tuned) less frequently during the learning process, then model can take longer time to train. Thus, optimization algorithms having fixed learning rate take days or week to terminate while training large-scale knowledge graphs [49], [3].

Optimization algorithm is used to minimize an objective function $J(\theta)$ parameterized by a model's parameter $\theta \epsilon \mathbb{R}^d$ by updating the parameters in the opposite direction to the gradient of the objective function $\nabla_\theta J(\theta)$ with respect to model parameters. The learning rate η decides the size of step we need to take to reach minimum value (better global minimum). Stochastic Gradient Descent(SGD) performs a parameter update for each training example $x^{(i)}$ and label $y^{(i)}$ *i.e.*

$$\theta = \theta - \eta.\nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.4}$$

Another variant of SGD which performs update for every *mini-batch* of n training examples is called *mini-batch gradient descent* and have following update equation:

$$\theta = \theta - \eta.\nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.5}$$

Gradient Descent algorithms suffer from limitation of carefully tuning the learning rate η and inability to adapt to dataset's characteristics [93]. AdaGrad [88] adapts the learning rate to the parameters because of which it is well-suited for dealing with sparse data [93]. The update rule of AdaGrad is given below:

$$G^k = G^{k-1} + \nabla J(\theta^{k-1})^2 \tag{2.6}$$

$$\theta^k = \theta^{k-1} - \frac{\alpha}{\sqrt{(G^{k-1})}}.\nabla J(\theta^{k-1}) \tag{2.7}$$

where . and sqrt are element wise operations. G is the historical gradient information. For each parameter, sum of squares is stored for all historical gradients. This sum is used to scale/adapt a learning rate. In contrast to SGD, AdaGrad learning rate is different for each of the parameters.

2.6 Evaluating triple-based link prediction models

The performance of various knowledge base embedding models focuses on two knowledge base completion tasks: (1) *Link prediction*, which aims at predicting the missing entity (either head or tail) in a given triple based on the existing facts in the KB. (2). *Triplet classification*, which aims at predicting whether the given triple is true or not.

2.6.1 Link prediction task

The task of *link prediction* is to predict the missing *head* or *tail* for a given triple in the form *(head, rel, tail)* as described in [46],[43],[50]. There are two metrics by which we measure the link prediction ability of any knowledge base embedding model, namely: (1). Mean Rank (MR), and (2). Hits@10 (H10).

Calculating Mean Rank (MR) and Hits@10 (H10)

For a given knowledge base denoted by $\mathcal{G}$, having m entities, n relations and N number of triples (or true facts) denoted as (h, r, t), where $(h, t) \in \mathcal{E}$ and $r \in \mathcal{R}$. We take triple (h, r, t) from dataset and replace either h or t, say h with all the m entities and calculate the score of each m triples.

head	relation	tail	label
Lauren Bacall	nominated_for	Academy Award	+1
Lauren Bacall	nominated_for	Booker Prize	-1

Table 2.3: Triplet classification example.

Then, sort these *m* triples in ascending order as per their score calculated by the *TransE* model and find the position of the correct triple in the entire *m* triples. This is called *rank* of the triple. We repeat the same process for all the triples present in the dataset and take the *mean* of *rank* of all triples, which give *Mean Rank* for *head side* called *HeadMR*. Same approach is followed by replacing tail t, which gives *Mean Rank* for *tail side* called *TailMR*. Finally, taking *mean* of *HeadMR* and *TailMR* gives overall *Mean Rank (MR)* of the entire dataset. If the *rank* of true triples lies within top 10 positions, then *Hits@10* is incremented by 1. For all the triples present in the dataset, calculating Hits@10 score for head side and dividing by the number of triples gives *HeadH10*. Similarly, we calculate Hits@10 score for tail side given by *TailH10*. Finally, taking *mean* of *HeadH10* and *TailH10* gives overall *Hits@10 score (H10)* for the entire dataset.

2.6.2 Triplet classification task

The task of *triple classification* is to confirm whether the given triple (h, r, t) is true or not [43], [75], [51], [50], [85]. The triplet classification is a *pair wise classification task* as proposed by [43], in which positive triple (labeled as +1) represents true fact having both entities and relation as true, but negative triple (labeled as -1) represents false fact having either of the entity as false. Each positive triple is provided with a negative triple which is formed by changing any of the entity (either head or tail) from positive one making it corrupt (or false). For each pair of triple (positive and negative), score is calculated and if the score of positive triple is *less* than the score of the negative triple, then it is predicted as *true*. As an example, given a positive and its negative triple in Table 2.3, the triple *(Lauren Bacall, nominated_for, Academy Award)* is correct, but the triple *(Lauren Bacall, nominated_for, Booker Prize)* is false because *Booker Prize* is always given for *literary work* and not to *movie stars*, and *Lauren Bacall* is a movie star. So, a good link prediction model will always give lower score to the correct triple and higher score to the false one. Hence, given test signed set, we count the number of triplet pairs that are predicted true, which gives the overall triplet classification accuracy. A good link prediction model gives *lower MR*, *higher H10* and *higher classification accuracy*. Lower the MR, better is the performance of the model in terms of Mean Rank evaluation metric. On the other hand, higher the H10 score, better is the performance of the model. Here H10 (in %) of 100 shows

Dataset	#E	#R	#Train	#Dev	#Test
WN18	40,943	18	141,442	5,000	5,000
FB15k	14,951	1,345	483,142	50,000	59,071

Table 2.4: Statistics of the experimental datasets used in previous works. #E is the number of entities, #R is the number of relations. #Train, #Dev and #Test are the numbers of triples in the train, dev and test sets, respectively.

Model	WORDNET (WN18)		FREEBASE(FB15K)	
-	MR	H10(%)	MR	H10 (%)
Unstructured [94]	315	35.3	1074	4.5
RESCAL [48]	1180	37.2	828	28.4
SE [47]	1011	68.5	273	28.8
SME Linear [95]	545	65.1	274	30.7
SME Bilinear [95]	526	54.7	284	31.3
LFM [76]	469	71.4	283	26.0
TransE [46]	263	75.4	243	34.9
TransE[AdaGrad] [3]	**169**	**80.5**	**189**	**44.0**

Table 2.5: Link Prediction Results: Test performance of several state-of-the-art link prediction models [Results reported from [3]].

best performance and 0 shows worst performance. Also, for classification task, higher the triplet classification accuracy, better is the performance of the model (where classification accuracy (in %) of 0 shows worst performance and 100 shows best performance).

2.7 Performance of triples-based link prediction models

In order to show that adaptive optimization algorithms are better at providing state-of-the-art link prediction results, [3] had done the empirical evaluation of various optimization algorithms on two standard datasets namely, *WORDNET* (WN18) and *FREEBASE* (FB15K)[46] based on two evaluation metrics as discussed in section 2.6 : (1). Mean Rank that measures the average position of the true test triplet in the ranking, and (2). Hits@10 score that measures the number of times the true test triplet is ranked among the most likely 10 triples (i.e. proportion of ranks not larger than 10). Table 2.4 shows the statistics of the standard datasets used. Experimental results shown in Table 2.5 based on two evaluation metrics shows that *TransE model* when trained with *adaptive learning rate* provides the best results.

The literature of Statistical Relational Learning [55], [96], [97], [98] is vast and lot of methods have been proposed for link prediction task in the past. The effectiveness of model for link prediction task depends upon: (1). require less parameters, (2). easy to train, (3). simple and efficient, (4). can be scalable to large knowledge graphs, (5). have better link prediction

and classification performance [98]. Taking consideration of all these points, *TransE* [46] is the most promising embedding model providing state-of-the-art link prediction results. Moreover, properly tuning model hyper-parameters and choosing better optimization algorithms (adaptive learning rates) can further enhance the performance of TransE model [99], [100], [101], [102]. Apart from this, simpler models like TransE tends to generalize better and are less prone to overfitting than complex models especially for real-world knowledge graphs which are very sparse [53]. Also, it has been experimentally proved that TransE model achieves state-of-the-art predictive performance on link prediction task, while being able to scale to large and web-scale knowledge graphs [103]. All these factors motivated us to opt TransE model for modeling n-ary relationship dataset. In this thesis work, we opted *TransE* model to generalize it over tuples having arbitrary arity.

The reason why we discussed *TransE* model in detailed manner is because we want to generalize TransE model over tuples having arbitrary arity. Moreover, we discussed *evaluation metrics* in detail because we want to extend the same evaluation metrics for tuples based link prediction. Though, there are various other more sophisticated triples-based link prediction models but majority of them are direct extensions of TransE model. So, it is reasonable to opt *TransE* model for generalizing it over tuples.

2.8 N-ary relation extraction

Past two decades have witnessed a significant advancement in extracting binary domain-dependent relations, but modern question-answering and summarization systems have triggered interest in capturing detailed information in a structured and semantically coherent fashion, thus motivating the need for complex n-ary relation extraction systems [58], [104]. [105] proposed n-ary relation extraction system that factorize complex n-ary relation into binary relations, representing them in a graph, and tried to reconstruct the complex relation by making tuples from selected maximal cliques in the graph. Then, each relation is classified using Maximum Entropy (MaxEnt) classifier. Recently, [106] make use of lexical semantics to train a model based on distant supervision for n-ary relation extraction. [58] proposed algorithm for extracting n-ary relations from biographical data which extracts entities using Conditional Random Fields (CRF) and n-ary relations using Support Vector Machine (SVM) from two manually annotated data-sets which contains biography summaries of Australian researchers.

Representing facts in the form of n-ary relations (or tuples) adds more expressiveness and also hold context of long relationships between entities, which in turn puts challenge of how we

can check the implausibility of these tuples. Hence, the task of n-ary link prediction will help to check the implausibility of facts represented in the form of tuples, improves the accuracy of n-ary relation extraction and will help to make KBs more complete, which in turn can help in varied downstream NLP applications.

2.9 Chapter Summary

This chapter gives overview of knowledge representation in the form of graph based databases, also known as KBs or KGs. It also provides what link prediction is and how link prediction can predict the missing facts in KBs based on the regularities in facts that are already exiting in the knowledge base. It also highlights literature of various triple-based embedding models for link prediction in KBs. Chapter concludes with importance of using adaptive learning rate optimization algorithms for improving performance of link prediction models and provide reasons for opting TransE model for modeling link prediction over n-ary relationships. In this thesis work we are generalizing *TransE* model over tuples, because of which we discussed in detail about *TransE* model and triple based link prediction evaluation metrics.

That there is no such thing as the scientific method, one might easily discover by asking several scientists to define it. One would find, I am sure, that no two of them would exactly agree. Indeed, no two scientists work and think in just the same ways.

Joel H. Hildebrand, American educator and pioneer chemist

3

Modeling N-ary Relationships

In this chapter, we describe Neo-Davidsonian representation, which is an obvious technique of decomposing tuples to triples. We also analyze its limitations when modeling n-ary relationships. Then, we propose two different models as a way of reducing tuples to triples namely: *The head-triple reduction* and *the clique reduction*. We also propose two different training methods namely: *triples based training* and *tuples based training*. In order to model our n-ary relationships (or tuples based) dataset using *TransE*, a triples-based embedding model, there is a need to convert tuples into triples so that we can provide training data in terms of triples to *TransE* model.

3.1 Reducing Tuples to Triples

In order to generalize link prediction for n-ary relationships and to leverage the advantage of predictive performance of *TransE* model, it is necessary to convert tuples into triples. Reducing tuples to triples provide triples based training data, which in turn can be used to train any standard triple-based link prediction model such as TransE. Though Neo-Davidsonian representation is quite obvious way of reducing tuples to triples, but it has limitations because of which it can't be used for modeling n-ary relationships. So, we proposed two different models as a way of

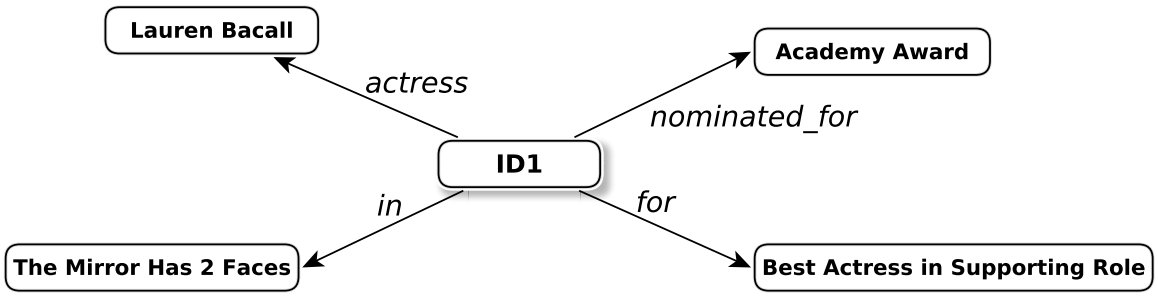

Figure 3.1: Neo-Davidsonian reduction of tuple P in terms of triples.

reducing tuples to triples. First, Neo-Davisonian reduction and its limitations are discussed and then we describe our proposed models and training methods.

3.1.1 Neo-Davidsonian reduction

Reification techniques such as Neo-Davidsonian representation [107] provides a way to represent n-ary relations [108]. In order to represent a tuple in the form of Neo-Davidsonian representation, an intermediate (or auxiliary) entity is created which acts as unique tuple ID and serves as the subject of the entire set of binary relations that are formed by decomposing tuples to triples. Let us suppose that tuple P is in our train set and tuple Q is in our test set as given below.

P: *(Lauren Bacall, nominated_for, Academy Award, for, Best Actress in Supporting Role, in, The Mirror has 2 Faces)*

Q: *(Lauren Bacall, nominated_for, BAFTA Award, for, Best Actress, in, The Shootist)*

The Neo-Davidsonian representation of tuple P in the form of triples is shown below, where ***ID1*** acts as a *tuple ID*.

*actress(**ID1**, Lauren Bacall)*
*nominated_for(**ID1**, Academy Award)*
*for(**ID1**, Best Actress in Supporting Role)*
*in(**ID1**, The Mirror has 2 Faces)*

The Neo-Davidsonian reduction of tuple P can be represented in the form of a graph as shown in Figure 3.1. Neo-Davidsonian reduction is an obvious way of decomposing tuples to triples as it completely encode all the information in the original tuple, but its limitation lies in getting latent representations (embeddings) of the tuple IDs that are unseen in training. Though model can learn the embeddings of entities, relations and tuple IDs that appear in the training

data, but it can't learn the embeddings of the tuple IDs representing tuples in the dev and test set that have never been seen by the model during training. Therefore, tuple IDs for tuples in dev and test set do not have embedding vectors. In Figure 3.1, ***ID1*** is the *tuple ID* for tuple P. Since, tuple P is in our train set, model can learn embedding vector for *ID1*, but the tuple Q is in test set and no doubt, we can convert it into triples assigning ***ID2*** to it as shown below, but model can't learn embeddings of ***ID2*** since it has never seen this id in the train set.

*actress(**ID2**, Lauren Bacall)*
*nominated_for(**ID2**, BAFTA Award)*
*for(**ID2**, Best Actress)*
*in(**ID2**, The Shootist)*

Learning embeddings of tuple IDs for the tuples present in *dev* and *test* set is not possible using Neo-Davidsonian reduction. Thus, Neo-Davidsonian reduction model can't learn embeddings of the tuple IDs that act as auxiliary nodes (or entities) for the entire dataset. Therefore, though Neo-Davidsonian reduction is able to reduce tuples of arbitrary arity into triples, but learning the embedding vectors of tuple IDs puts limitation in using Neo-Davidsonian reduction for modeling n-ary relationships (or tuples dataset). So, we propose two different models of decomposing tuples to triples overcoming the limitation of Neo-Davidsonian reduction, which are explained in next subsections.

3.1.2 The head-triple reduction model

The head-triple reduction model decomposes tuples to triples very similar to Neo-Davidsonian representation, overcoming its limitations of learning latent vector of tuple IDs. In head-triple reduction model, the *head* (h) acts as a common entity for all the triples formed by decomposing tuple. Given facts in the form of tuples, each tuple is reduced to triples using the head-triple reduction model as described in Algorithm 2. For example, given a tuple having representation: $(h, r_1, t_1, \ldots, r_n, t_n)$, it is decomposed into triples as follows: (h, r_1, t_1), (h, r_2, t_2), $\ldots$, (h, r_n, t_n).

As an example, given below are three tuples P, Q and R which are reduced to triples using head-triple reduction model as shown in Table 3.1. The horizontal lines in Table 3.1 are not the output of the model, they are to highlight the separation of all triples reduced from particular tuple.

P: ***(Lauren Bacall, nominated_for, Academy Award, for, Best Actress in Supporting Role, in, The Mirror has 2 Faces)***

Algorithm 2 The head-triple reduction model

Given tuple dataset having tuples T_i (where i= 1 to M) and let S denotes the set the triples that are reduced from tuples T. Then, the head-triple reduction model is defined by the following procedure:

Input

Tuples T_i where i=1 to M // Total M tuples in the dataset

Each tuple T_i is denoted by $(h, r1, t1, \ldots, rn, tn)$ where j=1 to n;

$(h, t1, \ldots, tn) \in \mathcal{E}$, $(r1, r2, \ldots, rn) \in \mathcal{R}$, $\mathcal{E}$ denotes entities and $\mathcal{R}$ denotes relations in the dataset;

Output

Set of triples S reduced from tuples T;

Procedure

1: **initialize** $S = \{\phi\}$ // Empty set of triples
2: **for** each tuple T_i where i=1 to M **do**
3: **for** j=1 to n **do**
4: $S = S\ U\ (h, r_j, t_j)$
5: **end for**
6: **end for**
7: **return**: S

head	rel	tail
(Lauren Bacall,	nominated_for,	Academy Award)
(Lauren Bacall,	for,	Best Actress in Supporting Role)
(Lauren Bacall,	in,	The Mirror has 2 Faces)
(Lauren Bacall,	nominated_for,	BAFTA Award)
(Lauren Bacall,	for,	Best Actress)
(Lauren Bacall,	in,	The Shootist)
(Humphrey Bogart,	nominated_for,	Academy Award)
(Humphrey Bogart,	for,	Best Actor in Leading Role)
(Humphrey Bogart,	in,	Casablanca)

Table 3.1: Reducing tuples P, Q and R into triples based on the head-triple reduction model. The horizontal lines highlight the separation of all triples reduced from particular tuple and are not the output of the model.

Q: ***(Lauren Bacall, nominated_for, BAFTA Award, for, Best Actress, in, The Shootist)***

R: ***(Humphrey Bogart, nominated_for, Academy Award, for, Best Actor in Leading Role, in, Casablanca)***

Graphical representation of reducing tuples P, Q and R using the head-triple reduction model is shown in Figure 3.2. As head is common for all the triples that are formed by reducing tuples to triples, this approach is named as the head-triple reduction. The head-triple reduction model arguably includes the key information from the tuples. One weakness is that the triples in the head-triple reduction associate all the relational information with the *head* entity. If the *head* entity is the only prominent entity in the tuple, then this is perhaps plausible, but in other relations this can result in important information being lost. For example, the head-triple reduction loses the information that *The Shootist* was nominated for a *BAFTA Award* while *Casablanca* was

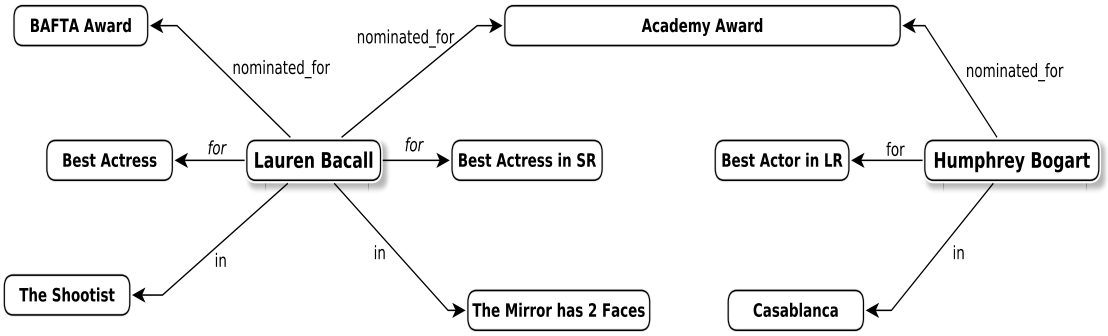

Figure 3.2: Graphical representation of Table 3.1 based on the head-triple reduction model.

nominated for an *Academy Award*. In order to overcome this ambiguity, we propose the clique reduction model which is described in the next subsection.

3.1.3 The clique reduction model

The clique reduction model adds some of the information that the head-triple reduction model is missing. It forms additional relations between entities from the existing relations in the tuple as a clique in a hypergraph. The formation of additional relations is similar to hyperedges in a hypergraph, therefore model is named as the clique reduction model. Given facts in the form of tuples, each tuple is reduced to triples using the clique reduction model as described in Algorithm 3. Given a tuple: $(h, r_1, t_1, \ldots, rn, tn)$, it is decomposed in the form of triples as follows: (h, r_1, t_1), (h, r_2, t_2), $(t_1, r_1^{-1}.r_2, t_2)$, (h, r_3, t_3), $\ldots$, (h, r_n, t_n). It is important to note that the clique reduction model, in addition to triples gives by the head-triple reduction model, gives *additional relations* in the form of $(r_1^{-1}.r_2), \ldots, (r_{n-1}^{-1}.r_n)$, and also *additional triples* in the form of $(t_1, r_1^{-1}.r_2, t_2), \ldots, (t_{n-1}, r_{n-1}^{-1}.r_n, t_n)$, . Here *new relation* $(r_1^{-1}.r_2)$ is formulated by *concatenation* of relations r_1 and r_2. The embedding vector of the new relation $(r_1^{-1}.r_2)$ is not derived from applying a composition function to the embedding on the inverse of relations r_1 and r_2. The reason is because it requires learning a function to map the embedding of a relation to its inverse too which will add additional computation complexity. To overcome this, the new relations are formed at the pre-processing stage and the model learns embedding of new relation in a standalone manner without any composition to its parent relations. The Formation of additional relations and additional triples depends upon the length of the tuple. For example, if tuple length is *five*, then there is *one additional relation* and *one additional triple*. If tuple length is *seven*, then there are *two additional relations* and *two additional triples*. Therefore, tuples having longer length add far more relations and triples as compared to shorter length.

Algorithm 3 The clique reduction model

Given tuple dataset having tuples T_i (where i= 1 to M) and let S denotes the set the triples that are reduced from tuples T. Then, the clique reduction model is defined by the following procedure:

Input

Tuples T_i where i=1 to M // Total M tuples in the dataset

Each tuple T_i is denoted by $(h, r1, t1, \ldots, rn, tn)$ where j=1 to n;

$(h, t1, \ldots, tn) \in \mathcal{E}$, $(r1, r2, \ldots, rn) \in \mathcal{R}$, $\mathcal{E}$ denotes entities and $\mathcal{R}$ denotes relations in the dataset;

Output

Set of triples S reduced from tuples T;

Procedure

initialize $S = \{\phi\}$ // Empty set of triples
for each tuple T_i where i=1 to M **do**
  for j=1 to n **do**
    $S = S\ U\ (h, r_j, t_j)$
    if $j < n$ **then**
      $S = S\ U\ (t_j, r_j^{-1}.r_{j+1}, t_{j+1})$
    end if
  end for
end for
return: S

Reducing tuples P, Q and R based on the clique reduction model contains all the triples from the head-triple reduction model, plus also the triples given in Table 3.2. The graphical representation of the tuples decomposed into triples using this model is shown in Figure 3.3. The entity name Supporting Role is shortened to SR and Leading Role is shortened to LR in figures to fit figures onto the page. In Figure 3.3, dotted lines represent new relations formed by concatenating existing relations. We can see that the extra triples in the clique reduction model encode the information that *The Shootist* was nominated for a *BAFTA Award*, while *The Mirror has 2 Faces* and *Casablanca* were nominated for *Academy Award*. In this case, the clique reduction model captures almost all the information in the source tuples.

head	rel	tail
(Academy Award,	nominated_for.for,	Best Actress in Supporting Role)
(Best Actress in Supporting Role,	for.in,	The Mirror has 2 Faces)
(Academy Award,	nominated_for.in,	The Mirror has 2 Faces)
(BAFTA Award,	nominated_for.for,	Best Actress)
(Best Actress,	for.in,	The Shootist)
(BAFTA Award,	nominated_for.in,	The Shootist)
(Academy Award,	nominated_for.for,	Best Actor in Leading Role)
(Best Actor in Leading Role,	for.in,	Casablanca)
(Academy Award,	nominated_for.in,	Casablanca)

Table 3.2: Reducing tuples P, Q and R into triples based on the clique reduction model. The horizontal lines highlight the separation of all triples reduced from particular tuple and are not the output of the model.

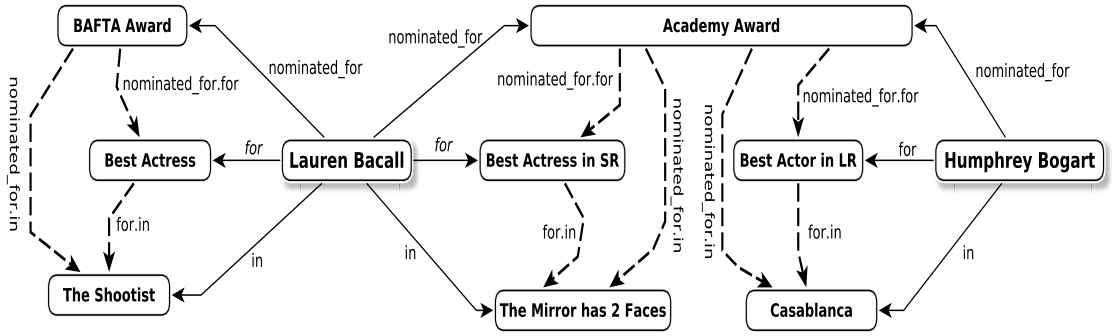

Figure 3.3: Graphical representation of Table 3.2 based on the clique reduction model. Dotted lines represent new relations formed by concatenating existing relations.

However, even the clique reduction model can sometimes fail to completely encode all the information in the source tuples. Suppose (counterfactually) that *Bacall* had also starred in *Casablanca*, and that *Bacall* had been nominated for an *Academy Award* and *Bogart* had been nominated for both an *Academy Award* and *BAFTA Award* for their performances. This would be encoded by the additional tuples S and T as given below:

S: *(Lauren Bacall, nominated for, Academy Award, for, Best Actress in Supporting Role, in, Casablanca)*

T: *(Humphrey Bogart , nominated for, BAFTA Award, for, Best Actor, in, Casablanca)*

It is easy to see that in this case the triples in the clique reduction model do not encode sufficient information to determine that Bogart was nominated for a BAFTA Award for Casablanca, while Bacall was not (Recall that Bacall was nominated for a BAFTA Award for a different movie).

3.2 Training the models

In order to train two different proposed models, we have proposed two different training methods. These training methods differs in the way we feed data as input during training. While triple based training method takes only triples as an input during training, tuples based training method takes complete tuple as an input during training. Both methods work by comparing the score of an element from the training data (which we call a positive example) with the score of a corrupted version of that element, in which an entity had been replaced with a randomly sampled entity (we call this a negative example). In triple based training the elements are triples, while in tuple based training the elements are tuples. The score of a tuple is the sum of the scores of the triples in its reduction.

3.2.1 Triple based training

Triples-based link prediction models such as TransE takes training data in the form of triples. Both, the head-triple reduction model as well as the clique reduction model provides triples which we can directly provide as input training data to train triples-based link prediction model. In triple based training, the model is trained on the triples in the way triples based prediction models are trained. For a given tuple: $(h, r_1, t_1, \ldots, r_n, t_n)$, also given in equation 3.1, the TransE model is trained on the triples formed from the tuples either formed by the head-triple reduction model or the clique reduction model. In triple based training, the training set to *TransE* model is in the form of triples and the score function of each triple formed is given by equation 3.2.

$$(h, r1, t1, r2, t2, \ldots, rn, tn) \text{ where } i = 1 \; to \; n \tag{3.1}$$

The complete tuple in equation 3.1 is decomposed into triples as given below and for each of these triples, the objective function is given in equation 3.2

$$(h, r1, t1), (h, r2, t2), (h, r3, t3), \ldots\ldots, (h, r_n, t_n)$$

$$f_r(h, t) = \|v_h + v_r - v_t\|_2^2 \tag{3.2}$$

where $f_r(h, t)$ denotes score of the triple, v_h, v_r and v_t denotes embedding vector of head, relation and tail respectively. For e.g. given tuple: *(Bogart, wins, Academy Award, for, Best Actor)*, the head-triple reduction model gives 2 triples from this tuple namely: *(Bogart, wins, Academy Award)* and *(Bogart, for, Best Actor)*, then the TransE model is trained having score function for these triples as follows: $\|v_{Bogart} + v_{wins} - v_{AcademyAward}\|_2^2$ and $\|v_{Bogart} + v_{for} - v_{BestActor}\|_2^2$ where v represents the embedding vector for particular entity or relation. On the other hand, when the clique reduction model is trained based on triples based training, it also takes additional triples that are formed by concatenating existing relations in the tuple. Taking same above example, there is formation of one additional relation, i.e. $wins^{-1}.for$ which is formed by concatenating relations *wins* and *for* present in the tuple. The score function of extra triples formed by the clique reduction model is given in equation 3.3.

$$f_{r_{new}}(t_i, t_{i+1}) = \|v_{t_i} + v_{r_{new}} - v_{t_{i+1}}\|_2^2 \tag{3.3}$$

where i=1 to n, r_{new} denotes additional relation formed by concatenating relations r_i and r_{i+1}, $f_{r_{new}}(t_i, t_{i+1})$ denotes score of the extra triple formed by clique reduction model, v_{t_i} denotes embedding vector of i^{th} tail, $v_{r_{new}}$ denotes embedding vector of new relation formed by concatenating

relation r_i and r_{i+1}, and $v_{t_{i+1}}$ denotes embedding vector of $(i+1)^{th}$. The clique reduction model provides one extra triple formed from the tuple as *(Academy Award, wins.for, Best Actor)*, hence the score function of this extra triple is as follows: $\|v_{AcademyAward} + v_{wins^{-1}.for} - v_{BestActor}\|_2^2$.

3.2.2 Tuple based training

In triples based training, it is quite straightforward to feed triples directly to TransE model which we have got from our two proposed models. But, there is loss of information because triples based training do not take longer context between entities and relations in given tuples. In order to take longer dependency between entities and relations, tuple based training is proposed which takes whole tuple as an input for training. Given a tuple denoted by equation 3.4, for the head-triple reduction model, the TransE model is trained having score function given in equation 3.5.

$$(h, r1, t1, r2, t2, \ldots, rn, tn) \text{ where } i = 1 \; to \; n \tag{3.4}$$

$$f(h, r1, t1, \ldots, rn, tn) = \sum_{i=1}^{n} \|v_h + v_{r_i} - v_{t_i}\|_2^2 \tag{3.5}$$

where i=1 to n, $f(h, r1, t1, \ldots, rn, tn)$ denotes score of the tuple, v_h, v_{r_i} and v_{t_i} denotes embedding vector of head, i^{th} relation and i^{th} tail respectively. For e.g. given tuple: *(Ben Affleck, wins, Academy Award, for, Best Actor)*, the TransE model is trained on the tuple having score function: $\|v_{Bogart} + v_{wins} - v_{AcademyAward}\|^2 + \|v_{Bogart} + v_{for} - v_{BestActor}\|^2$. On the other hand, for the clique reduction model, the TransE model is trained having score function given by equation 3.6.

$$f(h, r1, t1, \ldots, rn, tn) \quad = \quad \sum_{i=1}^{n} \|v_h \; + \; v_{r_i} \; - \; v_{t_i}\|_2^2 \; + \; \sum_{i=1}^{n-1} \|v_{t_i} \; + \; v_{r_{new}} \; - \; v_{t_{i+1}}\|_2^2 \tag{3.6}$$

where $f(h, r1, t1, \ldots, rn, tn)$ denotes score of the tuple, $v_{r_{new}} = v_{r_i^{-1} r_{i+1}}$ is the embedding vector of new relation formed by concatenating concatenating relations r_i and r_{i+1}. Also, v_h, v_{r_i} and v_{t_i} denotes embedding vector of head, i^{th} relation and i^{th} tail respectively. For e.g. given tuple: *(Bogart, wins, Academy Award, for, Best Actor)*, the TransE model is trained on the tuple having score function: $\|v_{Bogart} + v_{wins} - v_{AcademyAward}\|^2 + \|v_{Bogart} + v_{for} - v_{BestActor}\|^2 + \|v_{AcademyAward} + v_{wins^{-1}.for} - v_{BestActor}\|^2$. Here $r_{new} = r_i^{-1}.r_{i+1}$ represents $wins^{-1}.for$, which is an *additional relation* formed by concatenating relations *wins* and *for*. Tuple based training takes account of all entities and relations in a tuple which is beneficial for link prediction task such as *(Lauren Bacall, wins, ?)*. Apart from inter-tuple context, tuple based training might be

beneficial to take intra-tuple dependencies. For example, tuple of length 5 *(Humphrey Bogart, wins, Academy Award, for, Best Actor)* might be beneficial for link prediction task on tuple having length 3 *(Humphrey Bogart, profession, ?)* to predict *actor*, because it is highly likely that *movie stars* win *Academy Award*.

Coupling two different proposed models with two different proposed training methods gives overall 4 approaches. We call these 4 approaches namely: *Approach 1* (the head-triple reduction model coupled with triples-based training), *Approach 2* (the clique reduction model coupled with triples-based training), *Approach 3* (the head-triple reduction model coupled with tuples-based training) and *Approach 4* (the clique reduction model coupled with tuples-based training). In this thesis, we aim to evaluate performance of these proposed 4 approaches on our n-ary relationship (or tuples) dataset.

3.3 Chapter Summary

This chapter provides overview of Neo-Davidsonian representation, a technique of reducing tuples into triples. Neo-Davidsonian representation have limitations in terms of learning embedding vectors for tuple IDs which are not seen while training, which limits its use for modeling n-ary relationships. To overcome this limitation, Chapter proposed two different models as two different ways of reducing tuples into triples, namely *the head-triple reduction* and *the clique reduction*. We also proposed two different training methods as two different ways of training the proposed models.

The scientific man does not aim at an immediate result. He does not expect that his advanced ideas will be readily taken up. His work is like that of a planter -for the future. His duty is to lay the foundation of those who are to come and point the way.

Nicola Tesla, Inventor, physicist and engineer

4

Experimental Evaluation

4.1 Proposed modification in standard evaluation criteria

Training link prediction models on large-scale Knowledge Bases (KBs) have bottlenecks in terms of computation time [109]. One have to wait long to check how well a prediction model is performing during training. Link prediction models are trained on a train set and the best model is selected based on performance in terms of Mean Rank (MR) based on the Dev set [46]. Final evaluation on test set is done based on the best model selected. As the size of KB grows, calculating MR and Hits@10 on large set of entities is computationally expensive and often training takes huge amount of time (may be days, or weeks) depending upon the total number of entities in the KB. To overcome this challenge of computationally demanding task, we propose modification to existing evaluation criteria for evaluating performance of link prediction models on large-scale KBs which is simple, highly efficient and faster than original evaluation criteria, ultimately leading to rapid prototyping and visualization of model performance. Our modified evaluation criteria can be applied to almost all link prediction models and experimenter have the privilege to choose required set of entities out of total entities upon which one wants to check

the performance of the prediction model.

In our modified evaluation criteria, MR is calculated on a selected set of entities called *"evaluation subset"*, which are sampled from the entire set of entities present in the KB based on their probability count. The steps are listed below:

- Collect all entities by processing train, dev and test set. Let N denotes total number of entities (i.e. unique set of entities) that appear in train, dev and test set.
- Count the number of times a particular entity appears in the entire dataset giving count of each entity. Let n_i denotes the count of i^{th} entity among N entities, where $i = 1, 2, 3, ..., N$.
- Dividing entity count by the sum of all entity counts gives the probability of occurrence of each entity in the entire data set. i.e. $p_i = \frac{n_i}{\sum_{i=1}^{N} n_i}$ where p_i denotes the probability of i^{th} entity.
- Depending upon the number of entities we want to take into account in order to check model performance in terms of MR and H10 score, m entities are sampled from the entire distribution of entities without replacement to form *evaluation subset*.

Evaluating link prediction model on *evaluation subset* of entities rather than complete set of entities helps to overcome the computational bottleneck. In this thesis, we have taken *evaluation subset* of 1000 entities to check model performance during training and also to test the proposed approaches. The *evaluation subset* of 1000 entities follows the same distribution as the original set of entities. Moreover, while evaluating models on link prediction task, it is ensured that the head (or tail) entity must be there in the evaluation subset. If not, then head (or tail) entity is appended making overall 1001 entities, in turn making the resulted performance a good approximation of the performance measures when computed on the original set of entities.

4.2 Evaluating n-ary relationship models

In section 2.6 we describe evaluation protocol for triple-based models. We follow the same protocol for evaluating our proposed models for modeling n-ary relationships. Since we are evaluating tuples denoted by $(h, r1, t1, \ldots, rn, tn)$ having arbitrary length, we evaluate our proposed approaches on two standard tasks: (1). *Tuple classification*, and (2). *Link prediction over tuples*.

h	r1	t1	r2	t2	label
Lauren Bacall	nominated_for	Academy Award	in	The Mirror has 2 Faces	+1
Lauren Bacall	nominated_for	Academy Award	in	Titanic	-1

Table 4.1: Tuple classification example.

Tuple Classification

The task of *tuple classification* is to validate whether the given tuple is correct or not. Similar to triplet classification task, tuple classification task comes with *tuple pairs*. For a given positive tuple say, $(h, r1, t1, r2, t2, \ldots, rn, tn)$, first we sample i uniformly from 0 to n, then if $i = 0$, we replace the first entity in the positive tuple (i.e. head of the tuple) from an overall distribution of entities in the knowledge graph and so on. Hence *positive* tuple is the correct tuple given in the dataset, whereas *negative* tuple is obtained by replacing one of the entities with another entity chosen randomly. While making negative tuples, it is ensured that the formed negative tuple does not appear as a positive tuple in the entire dataset.

For a given pair of tuple (one positive and other negative), score of each tuple is calculated as the sum of the scores of the triples in its reduction, and if the score of positive tuple is less than the score of negative tuple, then it is predicted as true. As an example, Table 4.1 shows a pair of tuples where the positive tuple (labeled +1) have all entities and relations correct because Lauren Bacall has starred in The Mirror has 2 Faces and she was nominated for Academy Award for this movie, but the negative tuple (labeled -1) have one entity (tail in this example) wrong because Lauren Bacall has not starred in movie Titanic, making it a false tuple. The score of tuple is calculated as the sum of score of all the triples it reduces to as per the model (either the head-triple reduction or the clique reduction). Given a tuple denoted as given in equation 4.1, the score of tuple is calculated based on the head-triple reduction model given by equation 4.2.

$$(h, r1, t1, r2, t2, \ldots, rn, tn) \text{ where } i = 1 \; to \; n \tag{4.1}$$

$$\text{Score of tuple} = \sum_{i=1}^{n} f_{r_i}(h, t_i) \tag{4.2}$$

where $f_{r_i}(h, t_i)$ denotes the score of the triples formed from the tuple. In case of calculating score of the tuple based on the clique reduction model we have to take account of additional triples formed from the tuple. Hence, the score of tuple is calculated based on the clique reduction

h	r1	t1	r2	t2
Lauren Bacall	nominated_for	Academy Award	in	The Mirror has 2 Faces

Table 4.2: An example of tuple for link prediction.

model given by equation 4.3.

$$\text{Score of tuple} = \sum_{i=1}^{n} f_{r_i}(h, t_i) + \sum_{i=1}^{n-1} f_{r_{new}}(t_i, t_{i+1}) \tag{4.3}$$

where $r_{new} = r_i^{-1}.r_{i+1}$ denotes additional relations formed by concatenating existing relations in the tuple. $f_{r_i}(h, t_i)$ denotes the score of the simple triples and $f_{r_{new}}(t_i, t_{i+1})$ denotes the score of the additional triples formed by the clique reduction model. As an example, the score of tuple given in Table 4.2 is calculated based on the head-triple reduction model as given in equation 4.4. The name of entities have been shortened to fit text on page where LB denotes Lauren Bacall and AA denotes Academy Award for tuple given in Table 4.2.

$$\text{Score of tuple} = \text{Score of triple (LB, nominated\_for, AA)} + \text{Score of triple (LB, in, 2 Faces)} \tag{4.4}$$

Based on equation 4.3, the score of tuple given in Table 4.2 is calculated based on the clique reduction model as given in equation 4.5.

$$\begin{aligned}\text{Score of tuple} = {} & \text{Score of triple (LB, nominated\_for, AA)} + \text{Score of triple (LB, in, 2 Faces)} \\ & + \text{Score of triple (AA, nominated\_for.in, 2 Faces)}\end{aligned} \tag{4.5}$$

Link prediction

The task of link prediction is to predict any missing entity in a given tuple. In this thesis work, we focus on predicting a single missing entity (head or tail) in an n-ary relationship tuple. By tail, we mean last entity appearing in the tuple. For a tuple having notation $(h, r1, t1, \ldots, rn, tn)$, here *head* is denoted by h and *tail* is the last tail appearing in the tuple denoted by tn. For the sake of simplicity, we call tn as t (*i.e.* tail) in a tuple. The intuition behind taking only the head and the tail (the last entity in the tuple) into evaluation is to make the task of n-ary link prediction the direct extension of triples-based link prediction. Moreover, in our tuples dataset, tuples have different lengths (having triples also). So, intermediate entities are not subject to

evaluation because it is not possible to do evaluation on the triples present in our dev and test set (as they have only one head and one tail).

For each tuple in the dataset, we replace either *h* or *t* by *m* entities i.e. *evaluation subset* which are sampled as described in section 4.1. We calculate MR for head side, denoted by *HeadMR* and MR for tail side denoted by *TailMR* as described in section 2.6 replacing *head* and *tail* respectively in tuples. Taking *mean* of *HeadMR* and *TailMR* gives overall *MR* of the dataset. Also, if the position of correct tuple lies in the top 10 positions, then *Hits@10* score is incremented by 1. Calculating *HeadH10*, *TailH10* as described in Chapter 2 section 2.6 and taking their *mean*, gives overall *H10* score of dataset. Taking example of tuple from Table 4.2, the tuple's *true head* is *Lauren Bacall* and *true tail* is *The Mirror has 2 Faces*. While calculating rank and Hits@10 for head side, we replace head with 1000 entities that belong to evaluation subset. If true head does not appear in 1000 entities, then true head is appended making 1001 entities in evaluation subset. Then, score of all tuples is calculated and tuples are arranged in ascending order as per their scores. Finally we find the position of true tuple giving *rank* of the tuple and if position lies within top 10 positions, then Hits@10 is incremented by 1. We repeat the process for all tuples in the dataset for head side and taking their mean, gives *HeadMR* and *HeadH10*. Similarly, in order to calculate rank and Hits@10 for tail side, we replace tail *i.e. The Mirror has 2 Faces* with 1000 entities in evaluation subset. It is ensured that true tail lies in the evaluation subset, if not, then it is appended making 1001 entities in evaluation subset. Then, score of all tuples is calculated and tuples are arranged in ascending order as per their scores. Finally we find the position of true tuple giving *rank* of the tuple and if position lies within top 10 positions, then Hits@10 is incremented by 1. We repeat the process for all tuples in the dataset for tail side and taking their mean, gives *TailMR* and *TailH10*. Finally, taking mean of *HeadMR* and *TailMR* gives overall *MR* of the dataset; also taking mean of *HeadH10* and *TailH10* gives overall *H10* of the dataset.

A good link prediction model should have *lower MR*, *higher H10* and *higher classification accuracy*. Given two link prediction models, say *model A* and *model B*, then model A is better than model B if *MR(A)<MR(B)*, *H10(A)>H10(B)* and *Classification Accuracy (A) > Classification Accuracy (B)*.

4.3 Dataset

There are many standard datasets for link prediction on triples, but there is no dataset for link prediction on tuples. So, we created our own tuple dataset from latest Wikidata dump [7].

# E	#R	#Train	#Dev	#Test
102,231	353	684,488	9,998	9,997

Table 4.3: Statistics of the tuple dataset. # E denotes number of entities, # R denotes number of relations, # Train, # Dev and # Test denotes number of tuples in train, dev and test set respectively.

The Wikidata tuple dataset we have worked on in this thesis is generated from Wikidata dump of $3^{rd} January, 2017$. Entities and relations are pruned (*i.e.* do not appear in the dataset) if they appear less than particular count in the Wikidata dump, and also if they appear less than particular count in a long relation in the Wikidata dump. The dataset used in this thesis is named as $Wikidata - m1000 - l20$, highlighting that the entities and relations are pruned (i.e., they do not appear in a data file) if they appear less than 1000 times in the Wikidata dump and if they appear less than 20 times in a long relation in the Wikidata dump. The dataset contains tuples in the form of $(h, r1, t1, \ldots, rn, tn)$, where h denotes head entity, $r1$ denotes first relation, $t1$ denotes first tail entity, and so on. A file mapping entity and relation ids to English descriptions is also given so that we can check the actual English name corresponding to each entity and relation. Appendix A.1 provides the detailed statistics of our tuple dataset. In this thesis, we constrained our work over tuples having length not greater than 13. The reason for this constraint is due to lack of tuples in our dev and test set having length greater than 13 (see Appendix A.1 for details). Though there are $684,665$ train, $10,000$ dev, and $10,000$ test tuples in our original dataset, we removed tuples having length greater than 13, leaving $684,488$ tuples in train, $9,998$ tuples in dev and $9,997$ tuples in test set. Table 4.3 shows the statistics of tuple dataset we have worked upon.

4.4 Experiments

4.4.1 Experimental setup

In order to evaluate our proposed approaches based on *TransE* model, we choose learning rate η for *AdaGrad* among {0.001, 0.01, 0.1} , embedding dimension k among {20, 50, 100, 200, 500}, margin γ among {1, 2, 5, 10}, dissimilarity measure d as $\{L_2\}$ and number of sampled entities $m = 1000$ (*i.e.* evaluation subset) for calculating MR and H10 scores that are randomly sampled as described in section 4.1. For fair evaluation, we used same set of *1000 sampled negative entities* (or *evaluation subset*) in all of the four proposed approaches. For each of the proposed approach, best model is selected based on lowest Mean Rank (MR) on Development

Approach	HeadMR	TailMR	MR	HeadH10	TailH10	H10	Acc
Approach 1	347.63	27.10	**187.37**	5.34	92.92	**49.13**	**77.16**
Approach 2	332.65	25.11	**178.88**	9.78	93.24	**51.51**	**78.47**
Approach 3	220.20	32.68	**126.44**	10.79	91.69	**51.24**	**78.03**
Approach 4	194.23	29.11	**111.67**	17.25	92.82	**55.04**	**80.24**

Table 4.4: Results of proposed 4 approaches. HeadMR denotes Mean Rank for head side, TailMR denotes Mean Rank for tail side, MR denotes overall Mean Rank. Similarly, HeadH10 denotes Hits@10 for head side (in %), TailH10 denotes Hits@10 for tail side (in %) and H10 denotes overall Hits@10 score (in %). Acc (in %) denotes tuple classification accuracy.

set. Table 4.4 gives the experimental results of all the proposed 4 approaches.

In Table 4.4, approach 1 refers to *"the head-triple reduction model coupled with triples based training"*, approach 2 refers to *"the clique reduction model coupled with triples based training"*, approach 3 refers to *"the head-triple reduction model coupled with tuples based training"* and approach 4 refers to *"the clique reduction model coupled with tuples based training"*. From experimental result in Table 4.4, *approach 2* performs better than *approach 1*, giving *lower MR*, *higher H10* and *higher classification accuracy*. Similarly *approach 4* performs better than *approach 3* giving *lower MR*, *higher H10* and *higher classification accuracy*. Hence, the *clique reduction model* performs better than *the head-triple reduction model*. The clique reduction model forms extra relations from the relations already existing in the tuples, it provides more relations between entities. The clique reduction model is trained with lot many new relations apart from the existing ones given by the head-triple reduction model, which leads the clique reduction model to provide better predictive performance than the head-triple reduction model.

Moreover, if we compare the performance of *tuple based training* (i.e. approach 3 and 4) with *triples-based training* (i.e. approach 1 and 2), the *tuple based training models* provides *lower MR*, *higher H10* and *higher classification accuracy* compared to the *triples-based training models*. This signifies that tuple based training performs better compared to triple based training. In tuples based training, model is able to take account of links between all entities and relations in a particular tuple leading to better predictive performance. It is worth highlighting that when we replace entities with tail in the tuples, the performance of all four approaches is pretty good (*i.e.* low TailMR and high TailH10), but when we replace entities with head in the tuples, the performance of all four approaches is poor (*i.e.* very high HeadMR and quite low HeadH10). This further raised question *"Why predicting head is difficult compared to tail?"*. In order to answer this question, we did detailed analysis described in section 4.6. It is due to *TransE* model's inability to handle M-to-1, 1-to-M and M-to-M relationships. Comparing all the 4 approaches, approach 4 (The clique reduction model coupled with tuple based training) performs better than

other 3 proposed approaches, giving lowest MR, highest H10 and highest classification accuracy.

In terms of computational complexity, the dataset used in this thesis is quite large compared to various standard triple based datasets. Moreover, the clique reduction approach gives rise to large number of new relations (increasing quadratically in the worst case), making original evaluation criteria totally prohibited in term of computation time required to train models. Our proposed modification in evaluation criteria train models in short computation time (depending upon the number of entities to work upon), making it highly efficient leading to rapid prototyping and visualization of model performance. In this thesis work, we are taking 1000 entities into account for calculating MR and H10 which are sampled from total 102, 231 set of entities based on their probability distribution as described in 4.1. Hence, we are able to test our models 100 times faster using our proposed modification.

4.4.2 Complexity of the proposed models

In tuple dataset, there are 684, 488 tuples in the train set. The head-triple reduction model decompose these tuples into 1, 544, 560 triples having 102, 231 entities and 353 relations. It is found that the number of triples resulted from the clique reduction depends upon the number of triples resulted by the head-triple reduction and the total number of tuples as given by equation 4.6.

$$n_{cr} = 2 * n_{htr} - n_{tup} \tag{4.6}$$

where n_{cr} denotes the number of triples resulted from the clique reduction, n_{htr} denotes the number of triples resulted from the head-triple reduction and n_{tup} denotes the number of tuples in the dataset. Hence, the clique reduction model decompose these tuples into 2, 404, 632 triples having 102, 231 entities and 1, 736 relations. So, the clique reduction model give rise to additional 860, 072 triples and 1, 383 relations which increases model complexity. Therefore, the clique reduction model encodes additional information about tuples at the cost of higher computational complexity.

4.5 Qualitative Results

To illustrate model's capability in modeling n-ary relationships, this section presents sample examples where model is able to predict true tail (or head) in the top 10 positions and some examples where model fails to predict true tail (or head) in top 10 positions. Predicted tails or heads are top 10 entities predicted by model and are presented in order given by ranks.

Input (h, r1, t1, . . .)	Predicted Tails (Top 10 positions)
Conservative Party, instance of	**Political Party**, university, private not for profit educational institution, male given name, position, profession, municipality of Brazil, comune of Italy, country, art museum
Protein FAM83D, biological process, mitotic nuclear division, determination method	TAS, ISO, ISS, IBA, IPI, IMP, IDA, IGI, **IEA**, NAS
The smiling Lieutenant, nominated for, Academy Award for Best Picture, subject of	29th Tony Awards, 10th Tony Awards, **5th Academy Awards**, 11th Academy Awards, 6th Academy Awards, 7th Academy Awards, 15th Tony Awards, 54th Tony Awards, 33rd Tony Awards, 3rd Academy Awards
Norma Shearer, nominated for, Academy Award for Best Actress, for work, The Divorcee, subject of	13th Academy Awards,**3rd Academy Awards**, 26th Academy Awards, 16th Academy Awards, 15th Academy Awards, 18th Academy Awards, 32nd Academy Awards, 37th Academy Awards, 17th Academy Awards, 29th Academy Awards
Joseph Konstan, educated at, University of California Berkeley, academic degree, Master of Science, academic major	jurisprudence, economics, **computer science**, physics, mathematics, philosophy, political science, theology, chemistry, law
Dmitri Bystrov, place of birth, Moscow, located in the administrative territorial entity, Russian Soviet Federative Socialist Republic, country	United States of America, **Soviet Union**, Russian Empire, United Kingdom, Socialist Federal Republic of Yugoslavia, Abbasid caliphate, Italy, Sweden, Japan, Switzerland

Table 4.5: Example predictions of tails on test set based on the head-triple reduction model where model is able to predict true tail in top 10 positions. Bold indicates the test tuple's true tail predicted by the model.

4.5.1 Sample successful example predictions

Table 4.5 shows sample example predictions where model is successful in predicting true tail of given tuple in top 10 positions based on *the head-triple reduction* model. Input in terms of $(h, r1, t1, \ldots, rn)$ is given to the model and goal is to predict the true tail for the given input. `text` denotes relations in the tuple. Similarly, Table 4.6 shows sample example predictions where model is able to predict tuple's true head in top 10 positions. From these examples, it is clear that model is not only able to predict tuple's true tail (or head) in top 10 positions, but also is able to cluster entities which are quite relevant (or near) to tuple's true tail (or head), which makes common-sense. In case of the clique reduction model, model is given with additional relations formed by concatenating two relations present in the tuple, in addition to relations already present in the tuple. Taking first example from Table 4.7, actual tuple was *(Mile Jedinak, `place of birth`, Sydney, `country`, Australia)*. So, model is provided with additional one relation

Predicted heads (Top 10 positions)	**Input(. . . , rn, tn)**
Milan, Brigham Young University, Florence, Tilburg, Leiden, Groningen, **Vietnam**, Singapore, Toulouse, National Library	`instance of`, country
Screen Actors Guild Award for Outstanding Performances by a cast in a Motion Picture, Screen Actors Guild Award for Outstanding Performance by an Ensemble in a Drama Series, Academy Award for Best Supporting Actress, Golden Raspberry Award for Worst Picture, Screen Actors Guild Award for Outstanding Performance by an Ensemble in a Comedy Series, Directors Guild of America Award for Outstanding Directing Feature Film, Broadcast Film Critics Association Award for Best Cast, Hugo Award for Best Dramatic Presentation, Los Angeles Film Critics Association Award for Best Director, **BAFTA Award for Best Direction**	`winner`, Ang Lee, `for work`, Crouching Tiger Hidden Dragon, `winner.for work`
Douglas Shearer, **Norma Shearer**, Cedric Gibbons, Greg P Russell, Lyle R. Wheeler, Walter M Scott, Charles LeMaire, Edith Head, Edwin B Willis, Victor Young, Thomas Newman	`nominated for`, Academy Award for Best Actress, `for work`, The Divorcee, `subject of`, 3rd Academy Awards

Table 4.6: Example predictions of heads on test set based on the head-triple reduction model where model is able to predict true head in top 10 positions. Bold indicates the test tuple's true head predicted by the model.

`place of birth.country` which is formed by concatenating relation `place of birth` and relation `country`. Here, as the length of tuple is 5, so there is only 1 additional relation. If length of tuple is 7, then there are 2 additional relations, and so on. Hence, model has to predict true tail of the given tuple, i.e. *Australia* in this particular example. Table 4.7 shows various example for the clique reduction model, where model is able is predict given tuple's true tail in top 10 positions.

For predicting head based on the clique reduction model, model is provided with all the information as described above except head and model has to predict given tuple's true head. As an example from Table 4.8, the task for model is *(?, employer, College de France, position held, professor)*, i.e. predicting true head which is *Adam Mickiewicz* in this case. Here addition relation `employer.position held` is also provided to the model.

4.5.2 Sample failure example predictions

There are cases where model is unable to predict tuple's true tail (or head) in top 10 positions. Table 4.9 shows examples where model is unable to predict tuple's true tail in top 10 positions based on the head-triple reduction. In the first example of Table 4.9, model has to predict

Input (h, r1, t1, . . .)	Predicted Tails (Top 10 positions)
Mile Jedinak, `place of birth`, Sydney, `country`,`place of birth.country`	Russian Empire, United States of America, **Australia**, France, Soviet Union, Spain, United Kingdom, Germany, Sweden, Socialist Federal Republic of Yugoslavia
Avengers:Age of Ultron, `cast member`, Robert Downey Jr., `character role`, `cast member.character role`	B'Elanna Torres, Josh Lyman, Josiah Bartlet, Toby Ziegler, **Tony Stark/Iron Man**, Adam Schiff, C.J.Cregg, Charlie Young, Tom Paris, Margaret Hooper
Nucleoporin Nup43, `biological process`, cell cycle, `determination method`, `biological process.determination method`	**IEA**, IMP, IDA, IBA, TAS, ISS, IGI, ISO, NAS, IEP

Table 4.7: Example predictions of tails on test set based on the clique reduction model where model is able to predict true tail in top 10 positions. Bold indicates the test tuple's true tail predicted by the model.

Predicted heads (Top 10 positions)	Input(. . . , rn, tn)
Erhard Schmidt, Sagiri Kitao, Ernest Renan, Robert Ballard, **Adam Mickiewicz**, Herbert Kurke, Mashiro Okunu, Serotinin binding, Nicholas Rowe, Homeobox protein Hox-D3	`employer`, College de France, `position held`, professor, `employer.position held`
Phenacyl chloride exposure, propanoic acid exposure, abrin exposure, **Petroleum ether exposure**, Prosti, adiponitrile exposure, chloropicrin exposure, theionyl chloride exposure, Dimethyl carbamoyl chloride exposure, VX exposure	`first aid measure`, prompt washing with soap, `route of administration`, skin absorption, `first aid measure.route of administration`

Table 4.8: Example predictions of heads on test set based on the clique reduction model where model is able to predict true head in top 10 positions. Bold indicates the test tuple's true head predicted by the model.

true tail for the given input *(China Mieville,* `award received`*, Locus Award for Best Fantasy Novel,* `for work`*, ?)*, though model is unable to predict true tail but looking in details about the predicted tails reveal that all are movie names which makes sense for the given tuple. Similar trend can be seen in predicted heads in Table 4.10 where model is able to predict closely related head for the given tuple but unable to predict it in the top 10 positions.

Table 4.11 and Table 4.12 shows few examples where model is unable to predict tuple's true tail and head respectively based on the clique reduction model. Closely analyzing all these example reveals that though model is unable to predict true tail (or head) in top 10 positions but all the 10 top predicted entities (either head or tail) are closely related to the entity to be predicted. This shows that both models (the head-triple reduction model and the clique reduction model) are able to learn the semantics of the tuples quite well.

Input (h, r1, t1, . . .)	**Predicted Tails (Top 10 positions)**
China Mieville, award received, Locus Award for Best Fantasy Novel, for work	The Last King of Scotland, The Lord of the Rings:The Return of the King, Lost in Translation, Sideways, Antwone Fisher, Traffic, Argo, Villain, Karl Heinz Clasen, The Prestige
Julia Gillard, position held, Deputy Prime Minister of Australia, replaces, Mark Vaile, replaced by	Benjamin Disraeli, Bernhard von, Charles de Gaulle, Michael Foot, Andrei Smirnov, John Foster Dulles, Benedetto Cairoli, Avigdor Lieberman, Helmut Kohl, Kim Beazley

Table 4.9: Example predictions of tails on test set based on the head-triple reduction model where model fails to predict true tail in top 10 positions.

Predicted heads (Top 10 positions)	**Input(. . . , rn, tn)**
Mandatory Minimums, Rise, Ninety Miles Away, Posse Comitatus, Royal Affairs in Versailles, 20 Hours in America, Bad Moon Rising, Walter Hugo Gross, Should I Stay or Should I Go?, Requiem	cast member, John Spencer, character role, Leo McGarry
Joseph Heller, Abigail Breslin, Bruce Boxleitner, Nancy Allen, Ante Covic, Aleksandr Dugin, Anthony Franciosa, Nina Menshikova, Ivan Turgenev, Yuri Zavadsky	place of birth, Moscow, located in the administrative territorial entity, Russian Soviet Federative Socialist Republic, country, Soviet Union
ZNF573, EX0SC10, ADAMTS8, FAM83B, BMP2K, ATXN7, IMPAD1, CELSR2, TAF11, APCDD1	strand orientation, Reverse Strand, genomic assembly, Genome assembly GRCh38, genomic assembly, Genome assembly GRCh37

Table 4.10: Example predictions of heads on test set based on the head-triple reduction model where model fails to predict true head in top 10 positions.

On the basis of results, it is clear that both models (the head-reduction and the clique reduction) have predictive performance high for the tail side and have low for the head side. Moreover, there are various cases where the head-triple reduction model fails to predict true head (or tail) in top 10 positions, but the clique reduction model is able to predict them in top 10 positions. This reflects that the clique reduction, with additional complexity have higher predictive performance compared to the head-triple reduction. Finally, based on the examples of both successful and failure cases, it is clear that models are able to learn the properties of KG quite well. This is revealed from the fact that in all the examples, models are able to predict entities very similar to the tuple's actual tail (or head).

Input (h, r1, t1, . . .)	Predicted Tails (Top 10 positions)
Carlos Beltran, award received, Major League Baseball All-Star, league, National League, position player on team, award received.league, league.position player on team	power forward, first baseman, relief pitcher, center, shooting guard, forward, defender, small forward, midfielder, Carlo Lizzani
Hugo Award for Best Dramatic Presentation, winner, Mario Puzo, for work, winner.for work	The Lord of the Rings:The Return of the King, The Help, Gosford Park, The Sopranos, The King's Speech, Brokeback Mountain, Traffic, No Country for Old Men, Talk to Her, Aliens
Die vielen Abenteuer von Winnie Puuh, voice actor, Inge Wolffberg, character role, Rabbit, applies to part, voice actor.character role, character role.applies to part	background, takeoff, landing, left, foreground, right, Rabbit, down, adult, uridine kinase URK1 YNR012W

Table 4.11: Example predictions of tails on test set based on the clique reduction model where model fails to predict true tail in top 10 positions.

4.6 Performance of link prediction models with cardinality constraints

The performance of link prediction models are subject to different *cardinality constraints* [110], [46], [50], [51]. *Cardinality* on relations specifies the number of relations the particular entity is associated with. There are two cardinalities for each particular relation, namely *head cardinality* and *tail cardinality*. As an example, taking triple *(Humphrey Bogart, nominated_for, Academy Award)*, *head cardinality* of relation *nominated_for* is the number of entities that appear as head in this particular relation, which are the name of *movie stars*. On the other hand, *tail cardinality* of relation *nominated_for* is the number of entities that appear as tail in the relation *nominated_for*, which are the name of *Awards* that movies stars have been nominated for. Therefore, *cardinality constraints* can be *1-to-1* (one-to-one), *1-to-M* (one-to-many), *M-to-1* (many-to-one) and *M-to-M* (many-to-many).

TransE model has poor performance in handling *1-to-M* and *M-to-1* cardinality of relations [46], [50], [51]. To illustrate this fact, let us take an example for any triple based prediction model, *(Humphrey Bogart, gender, male)*. In this example, the relation *gender* can have very *high head cardinality*, because there can be *M* number of people in the KB who have gender as male. On the other hand, the relation *gender* have very *low tail cardinality* (exactly 2), because there are only two possible cases of gender (either male or female). Given the task of link prediction *(Lauren Bacall, gender, ?)*, any link prediction model can very well predict the true tail, which is female because the tail cardinality for relation gender is only 2 which is very small.

Predicted heads (Top 10 positions)	**Input(. . . , rn, tn)**
Santo Loquasto, John Lee Beatty, Anne Bancroft, Jo Mielziner, Diego Abad de santillan, Donald Pleasence, Ethan Coen, carbenoxolone, Jason Robards, National Football League	nominated for, Tony Award for Best Featured Actor in a Play, subject of, 58th Tony Awards, nominated for.subject of
Transforming growth factor beta-1, Neurogenic locus notch homolog protein 1, Sonic hedgehog protein, RAC-alpha serine, Presenilin-1, Bone morphogenetic protein 7, Apoptosis regulator Bcl 2, Catenin beta-1, P2X purinoceptor 7, cell division control protein 42 homolog	biological process, cell cycle, determination method, IEA, biological process.determination method
Georgy Malenkov, Valery Legasov, Eduard Bagritsky, Siko Dolidz, Mike Bryan, Roy Dolby, Aleksandr Domogarov, Boris Dolin, Ben Stiller, Paul Greengard	place of death, New York City, located in the admin territory, New York, country, United States of America, place of death.located in the admin territory, located in the admin territory.country

Table 4.12: Example predictions of heads on test set based on the clique reduction model where model fails to predict true head in top 10 positions.

On the other hand, given the task of link prediction *(?, gender, female)*, it is quite difficult to predict the very true head of the triple because the head cardinality of relation gender is very high.

To have an idea of how cardinality constraints affects our results, we did analysis of tuples in our train set to get *head cardinality* and *tail cardinality* for each relation. Table 4.13 shows 10 most common relations in our training data with their *actual name*, *head cardinality* and *tail cardinality*. For all the relations in Table 4.13, the *head cardinality* value is quite high compared to *tail cardinality*, which indicates that predicting head is difficult as compared to predicting tail. In order to investigate further, we did analysis of our dev set. The intuition behind analyzing dev set is that our test set is generated very similar to dev set. The analysis on dev set will directly reflect analysis of test set, without actually seeing test set. We processed tuples in our dev set to get *first relations* and *last relations*. First relation refers to the relation that appears after head and last relation refers to the relation that appear before the last tail. For example, given tuple W,

W: *(Battenin,* biological_process*, membrane organization,* determination_method*, IMP)*

In tuple W, first relation is *biological_process* as it is coming after the head, and last relation is *determination_method* as it is just appearing before tail. The reason why we focused on

rel	Actual Name	Head cardinality	Tail cardinality
P459	determination method	22839	25
P680	molecular function	21785	1790
P681	cell component	19962	745
P682	biological process	21846	4328
P1057	chromosome	6630	24
P2548	strand orientation	9951	2
P17	country	8637	224
P19	place of birth	3673	995
P805	subject of	5884	234
P1411	nominated for	6208	406

Table 4.13: Head cardinality and tail cardinality of most common relations in train set.

first and last relation in a tuple is because, when calculating MR and H10 for head side, we are replacing *m* entities with the head, which is directly linked to the first relation. Hence, *head cardinality* of first relation defines how model will perform in predicting head in top 10 positions. Similarly, when calculating MR and H10 for tail side, we are replacing *m* entities with that tail which is directly linked to the last relation. Hence, *tail cardinality* of last relation defines how model will perform in predicting tail in top 10 positions. In our *dev set*, having 9998 tuples, there are 94 relations that appear as first relation and there are 74 relations that appear as last relation. The three most common relations that appear as *first relation* in our dev set are *P680* (`molecular function`), *P681* (`cell component`) and *P682* (`biological process`) (these three relations occur in 8056 tuples out of total 9998 tuples in our dev set). Looking at these three relations in Table 4.13, we find that the *head cardinality* is very high compared to *tail cardinality*, making the task of predicting true head of the tuple quite difficult. On the other hand, the most common relation that appear as *last relation* in our dev set is *P459* (`determination method`) (this relation occurs in 8088 tuples out of total 9998 tuples in our dev set). Looking at relation *P459* in Table 4.13, we find that the *tail cardinality* is very low compared to *head cardinality*, making the task of predicting true tail of the tuple quite easy.

Link prediction models do not behave equally for head and tail side due to cardinality constraints. Due to high head cardinality and low tail cardinality, the predictive performance of our proposed approaches is poor for head side and good for tail side. This is directly reflected in the results we presented in Table 4.4.

4.7 Significance Test

Statistical evaluation of experimental results has widely been seen as a method of validating new machine learning models [111]. In any field of study, one is often testing whether a particular technique gives improved results in terms of any evaluation metrics on some test set, when compared to other techniques proposed for the same problem. Often this raises question in researcher's mind whether the improved results are due to particular technique actually being good or it is just due to chance. The random variation in evaluation results can be due to data bias, different intuition of human annotators, different experimental setups, and so on. Significance tests are designed to account for these variations that is due to random effects, assuming that all parameters that may have a systematic influence on evaluation results are kept constant. Hence, statistical significance is an estimate of the degree to which a particular technique (or approach) lies within a confidence interval. A commonly used level of reliability of result is 95%, also known as *p-value* or *p-level*. A p-value less than 0.05 is generally considered statistically significant.

It is the common practice in NLP to evaluate the performance of some given model on a small sample of held-out labeled data [112]. Hence, statistical significance testing provides the confidence that the observed difference in performance of two models is not likely to be due to mere chance [113], [112], [114]. [111] presents various arguments in favor of using *non-parametric* methods such as *Wilcoxon signed rank test* to estimate significance. Also, nonparametric methods are suitable for nominal, ordinal, interval and ratio scaled data. Since our evaluation metrics are nominal, nonparametric test such as Wilcoxon sign rank is most suitable for our problem. Non-parametric tests are distribution free tests which means that they do not assume a certain distribution of the input data. Wilcoxon singed rank test [115] is a non-parametric method alternative to *paired t-test* which ranks the differences in performances of two models.

We conducted *paired*, *two-sided* Wilcoxon sign rank test in R[1] [116], [117] for our proposed approaches. Approach 1 (The head-triple reduction model coupled with triples based training) is the obvious baseline for modeling n-ary relationship prediction. Approach 2 (The clique reduction model coupled with triples based training), Approach3 (The head-triple reduction model coupled with tuples based training) and Approach 4 (The clique reduction model coupled with tuples based training) are compared with the baseline (i.e. Approach 1).

The Wilcoxon sign rank test shows that approach 2 is significantly better than baseline having

[1]R: A Language and Environment for Statistical Computing and Graphics

$W = 28406000$ and $p < 2.2e^{-16}$, where p-value <0.01 shows statistical significant results.

The Wilcoxon sign rank test shows that approach 3 is significantly better than baseline having $W = 33451000$ and $p < 2.2e^{-16}$, where p-value <0.01 shows statistical significant results.

The Wilcoxon sign rank test shows that approach 4 is significantly better than baseline having $W = 35424000$ and $p < 2.2e^{-16}$, where p-value <0.01 shows statistical significant results.

For all the three evaluation metrics namely MR, H10 and classification accuracy, approach 2, approach 3 and approach 4 are significantly better than the baseline (p-value <0.01). The sample size of our dataset is huge because of which the calculated p-value given by R is quite low.

4.8 Chapter Summary

Experimental results on our tuple dataset showed that our proposed approaches have ability to generalize link prediction task in order to make KBs more robust and complete. Out of four proposed approaches, approach 4 (the clique reduction coupled with tuples-based training) performed significantly better than other 2 approaches and baseline in term of MR, H10 and classification accuracy. The proposed approaches gave high predictive performance for tail side and low for head side. Based on the analysis on dev set, performance is low for head side due to very high head cardinality of relations and inability of TransE model to handle M-to-1 and 1-to-M property of relations. Sample example predictions results highlighted the ability of our proposed approaches to model n-ary relationships and to hold the semantics of tuples, which is reflected from the examples that the predicted heads (or tails) highly matches to actual domain of entity (either head or tail) to be predicted.

There will come a time when you believe everything is finished. That will be beginning.

Louis L Amour

5 Conclusions and Future Work

In this thesis we presented novel approaches for link prediction on n-ary relationship data. We proposed two different models as two different ways of reducing tuples into triples, which in turn can be provided to any triple-based models to train. We also proposed two different training methods according to which the TransE model is trained. To evaluate our proposed approaches we presented tuple dataset generated from Wikidata. Experimental results showed that the clique reduction model when coupled with tuples-based training performed significantly better than other three proposed approaches.

5.1 Conclusion

The need to capture detailed and complete information in a structured and semantically coherent fashion has triggered research in extracting n-ary relations. Predicting n-ary relationships in KBs can be useful for relation extraction, question answering and summarization systems. We introduced the new task of n-ary relationship prediction, defined a new dataset and evaluation methodology for n-ary relationship prediction models, and introduced baseline models for n-ary relation prediction which are based on state-of-the-art TransE, triples-based link prediction

model. Experimental results showed that the clique reduction model performs better at link prediction task compared to the head-triple reduction model. Moreover, tuple based training provides better link prediction results compared to triple based training.

5.2 Future Work

The work in this thesis is more exploratory in nature and it opens many opportunities for extending the scope of this thesis. This section presents some of the research directions which would be explored in order to improve the task of n-ary link prediction.

- Using more sophisticated models

  It is reasonable and quite obvious to start task of modeling n-ary relationships with TransE model which is one of the most promising triple-based model due to its simplicity and state-of-the-art predictive performance. The work in this thesis have limitations in terms of poor performance in predicting tuple's correct head in top 10 positions. TransE model has issues in modeling M-to-1 and 1-to-M relationships, in turn gives poor predictive performance, because of which there is the need to use more sophisticated triple based models such as *TransH*, *TransR* and path based models such as *PTransE*.

- To try other optimization algorithms

  There exists a whole series of optimization algorithms. In this thesis we used *AdaGrad* optimization algorithm to optimize the loss function of TransE model. It would be interesting to explore other optimization algorithms including, *Momentum*, *RMSProp*, *Adam* and *Adadelta*. Also, more wider range of hyperparameters can be tried with these optimization algorithms, which might lead to improvement in results.

- Predicting two or more entities

  In this thesis we focus on predicting a single missing entity (either head or tail). Future work might involve simultaneously predicting two or more missing entities in a tuple, or predicting the relationships between a sequence of entities, which will be highly useful for improving the performance of n-ary relation extraction and also populating knowledge bases in a more robust and in an efficient manner. Also, it is of utmost importance in validating factoid answers where two or more entities (or relations) are erroneous produced as an output of the relation extractor.

- All entities and relations subject to evaluation

  As the objective of this thesis is to create a baseline, only the head and the tail (the last entity in the tuple) are subject to evaluation in the link prediction task. In future, all entities and relations present in the tuple (irrespective of their position) will be subject to evaluation.

- Leveraging composition of new relation from existing relations

  In this thesis, the new relation is formed by concatenating the strings of its two parent relations. It would be interesting to learn a mapping function in order to have composition of embedding of the parent relations to form embedding of new relation and its inverse too. Hence, the embedding vector of *new relation* ($r_1^{-1}.r_2$) can be derived by applying a composition function to the embeddings on the inverse of relations r_1 and r_2. This will lead to more sophisticated model leveraging the advantage of semantic composition of embedding vectors.

A

Appendix:Description of tuple dataset

A.1 Database Description

The tuple dataset that we worked on to evaluate our proposed approaches for modeling n-ary relationships is generated from Wikidata dump of $3^{rd} January$, 2017. Each fact is represented in the form of a tuple denoted as $(h, r1, t1, \ldots)$, where h denotes head, $r1$ denotes first relation, $t1$ denotes first tail, and so on. The tuples are extracted from Wikidata dump as per the thresholds on number of entities and relations. So, our tuple dataset contains facts formed by entities that appears at least 1000 times in any relation, or appear at least 20 times in a long relation (one with more than 2 entities). We split this collection into TRAIN, DEV and TEST sets. The entities are denoted with QXXX, where XXX after Q refers to entity ids and relations (or property) are denoted with PXXX, where XXX after P refers to relation ids. There is a file mapping entity and relation ids to English descriptions. As an example, Table A.1 shows actual tuple present in our train set and its corresponding actual representation in English. The statistics of the dataset is given in A.2. The detailed description about tuples of individual length in train, dev and test set is given in A.3.

Q16293174	P69	Q156598	P812	Q2329
Rutger van Santen	`educated at`	Leiden University	`academic major`	Chemistry

Table A.1: Sample tuple present in our dataset.

Entities	**Relations**	**Train**	**Dev**	**Test**	**DevSigned**	**TestSigned**
111,354	353	684,665	10,000	10,000	20,000	20,000

Table A.2: Statistics of Wikidata-m1000-l20 dataset.

Tuple Length	**Train**	**Dev**	**Test**	**Total**
3	2,239	45	42	2,326
5	537,731	7,800	7,817	553,348
7	115,871	1,733	1,735	119,339
9	24,507	364	333	25,204
11	3,622	54	63	3,739
13	518	2	7	527
15	72	0	0	72
17	26	1	0	27
19	13	1	0	14
21	15	0	2	17
23	8	0	0	8
25	7	0	0	7
27	4	0	1	5
29	6	0	0	6
31	1	0	0	1
33	8	0	0	8
35	1	0	0	1
37	5	0	0	5
41	1	0	0	1
43	6	0	0	6
49	1	0	0	1
53	2	0	0	2
107	1	0	0	1

Table A.3: Detailed statistics of Wikidata-m1000-l20 dataset.

References

[1] T. Mikolov, W.-t. Yih, and G. Zweig. *Linguistic regularities in continuous space word representations*. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 746–751 (Association for Computational Linguistics, 2013). URL http://aclweb.org/anthology/N13-1090.

[2] J. Feng, M. Huang, M. Wang, M. Zhou, Y. Hao, and X. Zhu. *Knowledge graph embedding by flexible translation*. In *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'16, pp. 557–560 (AAAI Press, 2016). URL http://dl.acm.org/citation.cfm?id=3032027.3032102.

[3] P. Minervini, C. D'amato, and N. Fanizzi. *Efficient energy-based embedding models for link prediction in knowledge graphs*. J. Intell. Inf. Syst. **47**(1), 91 (2016). URL http://dx.doi.org/10.1007/s10844-016-0414-7.

[4] N. Chomsky. *For Reasons of State* (Penguin Books, 2003).

[5] I. Rahwan and G. R. Simari. *Argumentation in Artificial Intelligence* (Springer Publishing Company, Incorporated, 2009), 1st ed.

[6] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. *Freebase: A collaboratively created graph database for structuring human knowledge*. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pp. 1247–1250 (ACM, New York, NY, USA, 2008). URL http://doi.acm.org/10.1145/1376616.1376746.

[7] D. Vrandečić and M. Krötzsch. *Wikidata: A free collaborative knowledgebase*. Commun. ACM **57**, 78 (2014).

[8] F. M. Suchanek, G. Kasneci, and G. Weikum. *YAGO: A core of semantic knowledge*. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pp. 697–706 (ACM, New York, NY, USA, 2007). URL http://doi.acm.org/10.1145/1242572.1242667.

[9] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. *DBpedia - a large-scale, multilingual knowledge base extracted from Wikipedia*. Semantic Web Journal (2014).

[10] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. *Toward an architecture for Never-Ending Language Learning*. In *AAAI* (2010).

[11] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. *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, KDD '14, pp. 601–610 (ACM, New York, NY, USA, 2014). URL http://doi.acm.org/10.1145/2623330.2623623.

[12] M. Gardner and T. Mitchell. *Efficient and expressive knowledge base completion using subgraph feature extraction*. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1488–1498 (Association for Computational Linguistics, 2015). URL http://aclweb.org/anthology/D15-1173.

[13] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and S. D. Weld. *Knowledge-based weak supervision for information extraction of overlapping relations*. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 541–550 (Association for Computational Linguistics, 2011). URL http://aclweb.org/anthology/P11-1055.

[14] J. Krishnamurthy and T. Mitchell. *Weakly supervised training of semantic parsers*. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 754–765 (Association for Computational Linguistics, 2012). URL http://aclweb.org/anthology/D12-1069.

[15] Y. Artzi and L. Zettlemoyer. *Weakly supervised learning of semantic parsers for mapping instructions to actions*. Transactions of the Association of Computational Linguistics **1**, 49 (2013). URL http://aclweb.org/anthology/Q13-1005.

[16] J. Berant, A. Chou, R. Frostig, and P. Liang. *Semantic parsing on Freebase from question-answer pairs*. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544 (Association for Computational Linguistics, 2013). URL http://aclweb.org/anthology/D13-1160.

[17] D. N. Milne, I. H. Witten, and D. M. Nichols. *A knowledge-based search engine powered by Wikipedia*. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pp. 445–454 (ACM, New York, NY, USA, 2007). URL http://doi.acm.org/10.1145/1321440.1321504.

[18] M. Zhang, B. Qin, T. Liu, and M. Zheng. *Triple based background knowledge ranking for document enrichment*. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 917–927 (Dublin City University and Association for Computational Linguistics, 2014). URL http://aclweb.org/anthology/C14-1087.

[19] J. Kim, G. Choi, J.-U. Kim, E.-k. Kim, and K.-S. CHOI. *The open framework for developing knowledge base and question answering system*. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pp. 161–165 (The COLING 2016 Organizing Committee, 2016). URL http://aclweb.org/anthology/C16-2034.

[20] Y. Kiyota, S. Kurohashi, and F. Kido. *"Dialog Navigator": A question answering system based on large text knowledge base*. In *COLING 2002: The 19th International Conference on Computational Linguistics* (2002). URL http://aclweb.org/anthology/C02-1084.

[21] W. Yih and H. Ma. *Question answering with knowledge base, web and beyond*. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pp. 8–10 (Association for Computational Linguistics, 2016). URL http://aclweb.org/anthology/N16-4003.

[22] S. Jain. *Question answering over knowledge base using factual memory networks*. In *Proceedings of the NAACL Student Research Workshop*, pp. 109–115 (Association for Computational Linguistics, 2016). URL http://aclweb.org/anthology/N16-2016.

[23] Y. Zhang, K. Liu, S. He, G. Ji, Z. Liu, H. Wu, and J. Zhao. *Question answering over knowledge base with neural attention combining global knowledge information*. CoRR **abs/1606.00979** (2016). URL http://arxiv.org/abs/1606.00979.

[24] R. Bunescu and R. Mooney. *Learning to extract relations from the web using minimal supervision*. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 576–583 (Association for Computational Linguistics, 2007). URL http://aclweb.org/anthology/P07-1073.

[25] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. *Distant supervision for relation extraction without labeled data*. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 1003–1011 (Association for Computational Linguistics, 2009). URL http://aclweb.org/anthology/P09-1113.

[26] B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek. *Distant supervision for relation extraction with an incomplete knowledge base*. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 777–782 (Association for Computational Linguistics, 2013). URL http://aclweb.org/anthology/N13-1095.

[27] M. Fan, D. Zhao, Q. Zhou, Z. Liu, F. T. Zheng, and Y. E. Chang. *Distant supervision for relation extraction with matrix completion*. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 839–849 (Association for Computational Linguistics, 2014). URL http://aclweb.org/anthology/P14-1079.

[28] G. Angeli, J. Tibshirani, J. Wu, and D. C. Manning. *Combining distant and partial supervision for relation extraction*. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1556–1567 (Association for Computational Linguistics, 2014). URL http://aclweb.org/anthology/D14-1164.

[29] I. Augenstein, A. Vlachos, and D. Maynard. *Extracting relations between non-standard entities using distant supervision and imitation learning*. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 747–757 (Association for Computational Linguistics, 2015). URL http://aclweb.org/anthology/D15-1086.

[30] T. T. V. Nguyen and A. Moschitti. *End-to-end relation extraction using distant supervision from external semantic repositories*. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 277–282 (Association for Computational Linguistics, 2011). URL http://aclweb.org/anthology/P11-2048.

[31] B. Rosenfeld and R. Feldman. *Using corpus statistics on entities to improve semi-supervised relation extraction from the web*. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 600–607 (Association for Computational Linguistics, 2007). URL http://aclweb.org/anthology/P07-1076.

[32] I. Augenstein, D. Maynard, and F. Ciravegna. *Distantly supervised web relation extraction for knowledge base population*. Semantic Web **7**(4), 335 (2016). URL http://dx.doi.org/10.3233/SW-150180.

[33] B. Hachey, W. Radford, and J. R. Curran. *Graph-based named entity linking with wikipedia*. In *Proceedings of the 12th International Conference on Web Information System Engineering*, WISE'11, pp. 213–226 (Springer-Verlag, Berlin, Heidelberg, 2011). URL http://dl.acm.org/citation.cfm?id=2050963.2050980.

[34] A. Chisholm, W. Radford, and B. Hachey. *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, chap. Discovering Entity Knowledge Bases on the Web, pp. 7–11 (Association for Computational Linguistics, 2016). URL http://aclweb.org/anthology/W16-1302.

[35] L. Du, A. Kumar, M. Johnson, and M. Ciaramita. *Using entity information from a knowledge base to improve relation extraction*. In *Proceedings of the Australasian Language Technology Association Workshop 2015*, pp. 31–38 (2015). URL http://aclweb.org/anthology/U15-1004.

[36] J. M. Zelle and R. J. Mooney. *Learning to parse database queries using inductive logic programming*. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pp. 1050–1055 (AAAI Press, 1996). URL http://dl.acm.org/citation.cfm?id=1864519.1864543.

[37] S. Yavuz, I. Gur, Y. Su, M. Srivatsa, and X. Yan. *Improving semantic parsing via answer type inference*. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 149–159 (Association for Computational Linguistics, 2016). URL http://aclweb.org/anthology/D16-1015.

[38] J. Herzig and J. Berant. *Neural semantic parsing over multiple knowledge-bases*. CoRR **abs/1702.01569** (2017). URL http://arxiv.org/abs/1702.01569.

[39] Y. Xue, E.-W. Lameijer, K. Ye, K. Zhang, S. Chang, X. Wang, J. Wu, G. Gao, F. Zhao, J. Li, C. Han, and S. Xu. *Precision medicine: What challenges are we facing?* Genomics, Proteomics and Bioinformatics **14**(5), 253 (2016). SI: Big Data and Precision Medicine, URL http://www.sciencedirect.com/science/article/pii/S1672022916301401.

[40] L. Huang, H. Fernandes, H. Zia, P. Tavassoli, H. Rennert, D. Pisapia, M. Imielinski, A. Sboner, M. A. Rubin, M. Kluk, and O. Elemento. *The cancer precision medicine knowledge base for structured clinical-grade mutations and interpretations*. Journal of the American Medical Informatics Association **1**(0(0)), 1 (2017).

[41] D. L. Mowery, B. R. South, L. Christensen, J. Leng, L.-M. Peltonen, S. Salentera, D. Suominen, Hanna Martinez, S. Velupillai, N. Elhadad, G. Savova, S. Pradhan, and W. W. Chapman. *Normalizing acronyms and abbreviations to aid patient understanding of clinical texts:shARe/CLEF eHealth challenge 2013, task 2*. Journal of Biomedical Semantics **7:43**, 1 (2016). URL http://doi.org/10.1186/s13326-016-0084-y.

[42] M. Adnan, J. Warren, and H. Suominen. *Patient empowerment via technologies for patient-friendly personalized language*, pp. 153–164 (De Gruyter, Berlin, Germany, 2015). URL https://doi.org/10.1515/9781614514343.

[43] R. Socher, D. Chen, C. D. Manning, and A. Ng. *Reasoning with neural tensor networks for knowledge base completion*. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds., *Advances in Neural Information Processing Systems 26*, pp. 926–934 (Curran Associates, Inc., 2013). URL http://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion.pdf.

[44] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin. *Knowledge base completion via search-based question answering*. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pp. 515–526 (ACM, New York, NY, USA, 2014). URL http://doi.acm.org/10.1145/2566486.2568032.

[45] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. *A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction*. CoRR **abs/1503.00759** (2015). URL http://arxiv.org/abs/1503.00759.

[46] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. *Translating embeddings for modeling multi-relational data*. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds., *Advances in Neural Information Processing Systems 26*, pp. 2787–2795 (Curran Associates, Inc., 2013). URL http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf.

[47] A. Bordes, J. Weston, R. Collobert, and Y. Bengio. *Learning structured embeddings of knowledge bases*. In *Conference on Artificial Intelligence* (2011).

[48] M. Nickel, V. Tresp, and H.-P. Kriegel. *A three-way model for collective learning on multi-relational data*. In L. Getoor and T. Scheffer, eds., *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pp. 809–816 (ACM, New York, NY, USA, 2011).

[49] K.-W. Chang, S. Wen-tau Yih, B. Yang, and C. Meek. *Typed tensor decomposition of knowledge bases for relation extraction*. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, 2014). URL https://www.microsoft.com/en-us/research/publication/typed-tensor-decomposition-of-knowledge-bases-for-relation-extraction/.

[50] W. Zhen, Z. Jianwen, F. Jianlin, and C. Zheng. *Knowledge graph embedding by translating on hyperplanes* (AAAI - Association for the Advancement of Artificial Intelligence, 2014). URL https://www.microsoft.com/en-us/research/publication/knowledge-graph-embedding-by-translating-on-hyperplanes/.

[51] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. *Learning entity and relation embeddings for knowledge graph completion*. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pp. 2181–2187 (AAAI Press, 2015). URL http://dl.acm.org/citation.cfm?id=2886521.2886624.

[52] H. Xiao, M. Huang, and X. Zhu. *Transg : A generative model for knowledge graph embedding*. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2316–2325 (Association for Computational Linguistics, Berlin, Germany, 2016). URL http://www.aclweb.org/anthology/P16-1219.

[53] D. Q. Nguyen, K. Sirts, L. Qu, and M. Johnson. *STransE: a novel embedding model of entities and relationships in knowledge bases*. In *Proceedings of the 2016 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 460–466 (Association for Computational Linguistics, San Diego, California, 2016). URL http://www.aclweb.org/anthology/N16-1054.

[54] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon. *Representing text for joint embedding of text and knowledge bases*. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1499–1509 (Association for Computational Linguistics, 2015). URL http://aclweb.org/anthology/D15-1174.

[55] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)* (The MIT Press, 2007).

[56] A. García-Durán, A. Bordes, N. Usunier, and Y. Grandvalet. *Combining two and three-way embedding models for link prediction in knowledge bases*. J. Artif. Int. Res. **55**(1), 715 (2016). URL http://dl.acm.org/citation.cfm?id=3013558.3013578.

[57] B. Shi and T. Weninger. *ProjE: Embedding projection for knowledge graph completion*. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pp. 1236–1242 (2017). URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14279.

[58] G. Khirbat, J. Qi, and R. Zhang. *N-ary Biographical Relation Extraction using Shortest Path Dependencies*, chap. In Proceedings of Australasian Language Technology Association Workshop (Association for Computational Linguistics, 2016). URL http://alta2016.alta.asn.au/U16/U16-1008.pdf.

[59] N. Grewe. *A generic reification strategy for n-ary relations in dl*. In H. Herre *et al.*, eds., *Proceedings of the 2nd workshop of the GI-Fachgruppe 'Ontologien in Biomedizin und Lebenswissenschaften' (OBML) : Mannheim, Germany, Sep 9–10*, pp. N1–5 (2010).

[60] D. Calvanese, G. De Giacomo, and M. Lenzerini. *Conjunctive query containment in description logics with n-ary relations*. In R. Brachman *et al.*, eds., *Proceedings of the 1997 Description Logic Workshop (DL97)*, pp. 5–9 (1997).

[61] H.-U. Krieger and C. Willms. *Proceedings of the 1st Workshop on Language and Ontologies*, chap. Extending OWL Ontologies by Cartesian Types to Represent N-ary Relations in Natural Language (Association for Computational Linguistics, 2015). URL http://aclweb.org/anthology/W15-0401.

[62] E. Portmann, P. Kaltenrieder, and W. Pedrycz. *Knowledge representation through graphs*. Procedia Computer Science **62**, 245 (2015). URL http://www.sciencedirect.com/science/article/pii/S1877050915025818.

[63] J. Piskorski and R. Yangarber. *Information Extraction: Past, Present and Future*, pp. 23–49 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2013). URL http://dx.doi.org/10.1007/978-3-642-28569-1_2.

[64] G. Palshikar, R. Srivastava, S. Shankar Deshpande, and S. Bhat. *Information extraction for effective knowledge management*. Research Gate **10.13140/RG.2.1.3629.5525** (2012). URL https://www.researchgate.net/publication/280941207.

[65] I. Borislav. *Hypergraphdb: A generalized graph database*. In *Proceedings of the 2010 International Conference on Web-age Information Management*, WAIM'10, pp. 25–36 (Springer-Verlag, Berlin, Heidelberg, 2010). URL http://dl.acm.org/citation.cfm?id=1927585.1927589.

[66] X. Li, A. Taheri, L. Tu, and K. Gimpel. *Commonsense knowledge base completion*. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers* (2016). URL http://aclweb.org/anthology/P/P16/P16-1137.pdf.

[67] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard. *Complex embeddings for simple link prediction*. In *International Conference on Machine Learning (ICML)*, vol. 48, pp. 2071–2080 (2016).

[68] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)* (The MIT Press, 2007).

[69] B. Taskar, M. fai Wong, P. Abbeel, and D. Koller. *Link prediction in relational data*. In S. Thrun, L. K. Saul, and P. B. Schölkopf, eds., *Advances in Neural Information Processing Systems 16*, pp. 659–666 (MIT Press, 2004). URL http://papers.nips.cc/paper/2465-link-prediction-in-relational-data.pdf.

[70] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning* (The MIT Press, 2009).

[71] T. Mikolov, S. Wen-tau Yih, and G. Zweig. *Linguistic regularities in continuous space word representations*. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)* (Association for Computational Linguistics, 2013). URL https://www.microsoft.com/en-us/research/publication/linguistic-regularities-in-continuous-space-word-representations/.

[72] J. Turian, L. Ratinov, and Y. Bengio. *Word representations: A simple and general method for semi-supervised learning*. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pp. 384–394 (Association for Computational Linguistics, Stroudsburg, PA, USA, 2010). URL http://dl.acm.org/citation.cfm?id=1858681.1858721.

[73] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. *Natural language processing (almost) from scratch*. J. Mach. Learn. Res. **12**, 2493 (2011). URL http://dl.acm.org/citation.cfm?id=1953048.2078186.

[74] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. *Joint learning of words and meaning representations for open-text semantic parsing*. In *In Proceedings of 15th International Conference on Artificial Intelligence and Statistics* (2012).

[75] D. Q. Nguyen. *An overview of embedding models of entities and relationships for knowledge base completion*. CoRR **abs/1609.04747** (2017). URL http://arXiv:1703.08098v2.

[76] R. Jenatton, N. L. Roux, A. Bordes, and G. Obozinski. *A latent factor model for highly multi-relational data*. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'12, pp. 3167–3175 (Curran Associates Inc., USA, 2012). URL http://dl.acm.org/citation.cfm?id=2999325.2999488.

[77] Q. Xie, X. Ma, Z. Dai, and E. Hovy. *An interpretable knowledge transfer model for knowledge base completion*. CoRR **arXiv:1704.05908v1** (2017). URL http://arxiv.org/1704.05908v1.

[78] Y. Luo, Q. Wang, B. Wang, and L. Guo. *Context-dependent knowledge graph embedding*. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1656–1661 (Association for Computational Linguistics, 2015). URL http://aclweb.org/anthology/D15-1191.

[79] A. Garcia-Duran, A. Bordes, and N. Usunier. *Composing relationships with translations*. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 286–290 (Association for Computational Linguistics, 2015). URL http://aclweb.org/anthology/D15-1034.

[80] M. Gardner and T. Mitchell. *Efficient and expressive knowledge base completion using subgraph feature extraction*. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1488–1498 (Association for Computational Linguistics, 2015). URL http://aclweb.org/anthology/D15-1173.

[81] M. Gardner, P. Talukdar, J. Krishnamurthy, and T. Mitchell. *Incorporating vector space similarity in random walk inference over knowledge bases*. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 397–406 (Association for Computational Linguistics, 2014). URL http://aclweb.org/anthology/D14-1044.

[82] K. Guu, J. Miller, and P. Liang. *Traversing knowledge graphs in vector space*. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 318–327 (Association for Computational Linguistics, 2015). URL http://aclweb.org/anthology/D15-1038.

[83] N. Lao, T. Mitchell, and W. W. Cohen. *Random walk inference and learning in a large scale knowledge base*. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 529–539 (Association for Computational Linguistics, 2011). URL http://aclweb.org/anthology/D11-1049.

[84] N. Lao and W. W. Cohen. *Relational retrieval using a combination of path-constrained random walks*. Mach. Learn. **81**(1), 53 (2010). URL http://dx.doi.org/10.1007/s10994-010-5205-8.

[85] D. Q. Nguyen, K. Sirts, L. Qu, and M. Johnson. *Neighborhood Mixture Model for Knowledge Base Completion*. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 40–50 (2016).

[86] K. Toutanova, V. Lin, W.-t. Yih, H. Poon, and C. Quirk. *Compositional learning of embeddings for relation paths in knowledge base and text*. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1434–1444 (Association for Computational Linguistics, 2016). URL http://aclweb.org/anthology/P16-1136.

[87] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu. *Modeling relation paths for representation learning of knowledge bases*. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 705–714 (Association for Computational Linguistics, 2015). URL http://aclweb.org/anthology/D15-1082.

[88] J. Duchi, E. Hazan, and Y. Singer. *Adaptive subgradient methods for online learning and stochastic optimization*. J. Mach. Learn. Res. **12**, 2121 (2011). URL http://dl.acm.org/citation.cfm?id=1953048.2021068.

[89] M. D. Zeiler. *ADADELTA: an adaptive learning rate method*. CoRR **abs/1212.5701** (2012). URL http://arxiv.org/abs/1212.5701.

[90] D. C. Liu and J. Nocedal. *On the limited memory BFGS method for large scale optimization*. Mathematical Programming **45**(1), 503 (1989). URL http://dx.doi.org/10.1007/BF01589116.

[91] D. P. Kingma and J. Ba. *Adam: A method for stochastic optimization*. CoRR **abs/1412.6980** (2015). URL http://arxiv.org/abs/1412.6980.

[92] T. Schaul, S. Zhang, and Y. LeCun. *No more pesky learning rates*. CoRR **abs/1206.1106** (2013). URL http://arXiv:1206.1106v2.

[93] S. Ruder. *An overview of gradient descent optimization algorithms*. CoRR **abs/1609.04747** (2016). URL http://arxiv.org/abs/1609.04747.

[94] A. Bordes and E. Gabrilovich. *Constructing and mining web-scale knowledge graphs: Kdd 2014 tutorial*. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 1967–1967 (ACM, New York, NY, USA, 2014). URL http://doi.acm.org/10.1145/2623330.2630803.

[95] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. *A semantic matching energy function for learning with multi-relational data*. Machine Learning **94**(2), 233 (2014). URL http://dx.doi.org/10.1007/s10994-013-5363-6.

[96] L. D. Raedt, K. Kersting, S. Natarajan, and D. Poole. *Statistical Relational Artificial Intelligence: Logic, Probability and Computation* (Morgan and Claypool Publishers, 2016).

[97] L. D. Raedt. *Logical and Relational Learning* (Cognitive Technologies, Springer Berlin Heidelberg, 2008).

[98] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. *A review of relational machine learning for knowledge graphs*. Proceedings of the IEEE **104**(1), 11 (2016).

[99] B. Chiu, G. Crichton, A. Korhonen, and S. Pyysalo. *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, chap. How to Train good Word Embeddings for Biomedical NLP, pp. 166–174 (Association for Computational Linguistics, 2016). URL http://aclweb.org/anthology/W16-2922.

[100] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. *A latent variable model approach to pmi-based word embeddings*. Transactions of the Association of Computational Linguistics **4**, 385 (2016). URL http://aclweb.org/anthology/Q16-1028.

[101] B. Chandra and R. K. Sharma. *Deep learning with adaptive learning rate using laplacian score*. Expert Syst. Appl. **63**(C), 1 (2016). URL https://doi.org/10.1016/j.eswa.2016.05.022.

[102] R. An, W. J. Li, H. G. Han, and J. F. Qiao. *An improved levenberg-marquardt algorithm with adaptive learning rate for rbf neural network*. In *2016 35th Chinese Control Conference (CCC)*, pp. 3630–3635 (2016).

[103] E. Gabrilovich and N. Usunier. *Constructing and mining web-scale knowledge graphs*. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pp. 1195–1197 (ACM, New York, NY, USA, 2016). URL http://doi.acm.org/10.1145/2911451.2914807.

[104] M. Banek, D. Jurić, and Z. Skočir. *Learning Semantic N-Ary Relations from Wikipedia*, pp. 470–477 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010). URL http://dx.doi.org/10.1007/978-3-642-15364-8_39.

[105] R. McDonald, F. Pereira, S. Kulick, S. Winters, Y. Jin, and P. White. *Simple algorithms for complex relation extraction with applications to biomedical ie*. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 491–498 (Association for Computational Linguistics, 2005). URL http://aclweb.org/anthology/P05-1061.

[106] H. Li, S. Krause, F. Xu, A. Moro, H. Uszkoreit, and R. Navigli. *Improvement of n-ary relation extraction by adding lexical semantics to distant-supervision rule learning*. In S. Loiseau, J. Filipe, B. Duval, and H. J. van den Herik, eds., *ICAART (2)*, pp. 317–324 (SciTePress, 2015). URL http://dblp.uni-trier.de/db/conf/icaart/icaart2015-2.html#LiKX0UN15.

[107] D. Davidson. *The logical form of action sentences*. In N. Rescher, ed., *The Logic of Decision and Action* (University of Pittsburgh Press, 1967).

[108] J. Lehmann and J. Voelker. *Perspectives On Ontology Learning*. Studies in the Semantic Web (AKA / IOS Press, 2014). URL http://jens-lehmann.org/files/2014/perspectives_on_ontology_learning.pdf.

[109] P. Minervini, C. d'Amato, N. Fanizzi, and F. Esposito. *Efficient learning of entity and predicate embeddings for link prediction in knowledge graphs.* In F. Bobillo, R. N. Carvalho, D. Ceolin, P. C. G. da Costa, C. d'Amato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz, T. P. Martin, M. Nickles, and M. Pool, eds., *URSW@ISWC*, vol. 1479 of *CEUR Workshop Proceedings*, pp. 26–37 (CEUR-WS.org, 2015). URL http://dblp.uni-trier.de/db/conf/semweb/ursw2015.html#MinervinidFE15.

[110] J. Zhang, J. Chen, J. Zhu, Y. Chang, and P. S. Yu. *Link prediction with cardinality constraint*. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pp. 121–130 (ACM, New York, NY, USA, 2017). URL http://doi.acm.org/10.1145/3018661.3018734.

[111] J. Demšar. *Statistical comparisons of classifiers over multiple data sets*. J. Mach. Learn. Res. **7**, 1 (2006). URL http://dl.acm.org/citation.cfm?id=1248547.1248548.

[112] A. Søgaard. *Estimating effect size across datasets*. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 607–611 (Association for Computational Linguistics, 2013). URL http://aclweb.org/anthology/N13-1068.

[113] T. Berg-Kirkpatrick, D. Burkett, and D. Klein. *An empirical investigation of statistical significance in nlp*. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pp. 995–1005 (Association for Computational Linguistics, Stroudsburg, PA, USA, 2012). URL http://dl.acm.org/citation.cfm?id=2390948.2391058.

[114] Y. Zhang and S. Vogel. *Significance tests of automatic machine translation evaluation metrics*. Machine Translation **24**(1), 51 (2010). URL http://dx.doi.org/10.1007/s10590-010-9073-6.

[115] F. Wilcoxon. *Individual Comparisons by Ranking Methods*, pp. 196–202 (Springer New York, New York, NY, 1992). URL http://dx.doi.org/10.1007/978-1-4612-4380-9_16.

[116] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R* (Springer Publishing Company, Incorporated, 2014).

[117] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning* (Springer, 2001).