LANGUAGE, STRUCTURE, TIME AWARE KNOWLEDGE BASE COMPLETION

PRACHI JAIN

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY DELHI

October 2022

LANGUAGE, STRUCTURE, TIME AWARE KNOWLEDGE BASE COMPLETION

by

PRACHI JAIN

Department of Computer Science and Engineering

Submitted

in fulfillment of the requirements of the degree of Doctor of Philosophy

to the

INDIAN INSTITUTE OF TECHNOLOGY DELHI

October 2022

This dissertation is dedicated to
my loving grandparents
and
my son Arinjay – may you never stop learning.

Certificate

This is to certify that the dissertation titled **Language, Structure, Time aware Knowledge Base Completion** being submitted by **Ms Prachi Jain** for the award of **Doctor of Philosophy** in Department of Computer Science and Engineering is a record of bonafide work carried out by her under my guidance and supervision at the **Department of Computer Science and Engineering, Indian Institute of Technology Delhi**. The work presented in this dissertation has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma unless otherwise stated explicitly. In particular, work done in Chapters 4, 5, 6 and 7 were done jointly with undergraduate students. In each case, the part done by the collaborators appeared in their respective bachelor's thesis.

Mausam
Professor
Department of Computer Science and Engg.
Indian Institute of Technology Delhi
New Delhi- 110016

Soumen Chakrabarti
Professor
Department of Computer Science and Engg.
Indian Institute of Technology Bombay
Powai, Mumbai, Maharashtra- 400076

Acknowledgements

*"***It's all about the journey, not the outcome.***"*

Carl Lewis

As I am about to end my past-due PhD journey. I want to look back and extend my gratitude to all the people whom I met in the course of my PhD, without whom the long journey would have looked very dull.

I would like to express my deepest gratitude to my advisors Mausam and Soumen for their guidance and support during my PhD. Mausam's drive to do high-quality research and a constant push to go beyond the limits always kept me on my toes. His honest critical feedback always empowered me to be the best I can be. On the other hand, Soumen's passion for the subject and ability to pay attention to the minutest of details always kept me motivated and pushed me to do my best. I have vastly benefited from this unique mix of advisors. In these last two years, there were times when work life was unavoidably affected by events outside work. I am so grateful to Mausam and Soumen for caring, understanding, and prioritizing their students' well-being. Thank you Mausam and Soumen, for patiently being there in this period of my life and my career and always showering me with the wisest of advice.

I am thankful to Dr Denial Bernhardt and Apoorv Saxena for hosting me for an internship at Facebook. I would also like to thank Dr Sebastian Riedel, Sara Khodeir, and Gerard Goossen for their constant feedback and support during the internship.

I also want to acknowledge the TCS PhD fellowship for supporting me financially during my PhD. I am also thankful to Microsoft Research, Xerox Research, Google Research, IJCAI, and ACL student travel grants for supporting my travel to numerous conferences at various points in my PhD. I would also like to acknowledge the administrative staff (especially Rekha mam and Hemant bhaiya) at the Computer Science Department in IIT Delhi for their continuous help throughout these years. I was very lucky to have collaborated with some of the brightest undergraduate students on campus — Shikhar Murty, Kabir Chhabra, Pankaj Kumar, Mukund Mundhra, Sushant Rathi. I learnt a lot from them.

I would like to thank the organizing committee of the CIFAR Deep Learning + Reinforcement Learning (DLRL) Summer School and the Lisbon School of Machine Learning Summer

School (LxMLS) for organizing amazing summer schools. I learned a lot and made good friends during the summer schools. I would also like to thank them for supporting my travel. I will always remember the interesting discussions and fun trips with Disha during my time at DLRL summer school.

I have made lifelong friends over the last few years – Ankit, Happy, Himanshu and Yashoteja (aka JHAMP). Thank you for all the laughter, food, and wisdom to keep me moving over these years. I will always look upto them for any professional or personal advise. There are many more wonderful friends I made in my PhD — Yatin, Keshav, Vishal, Dilpreet, Jatin, Chandrika, Deepak, Madhulika, Prajna and Sanchi.

Finally, I would like to express my heartfelt gratitude to my parents for the numerous sacrifices they have made to support me and my education. Their unwavering faith always helped me sail through all the ups and downs of my PhD journey. I would also like to thank my grandparents, husband, sister, brother and parent-in-laws, without their love, support and encouragement I would not have achieved this feat.

Abstract

Large Knowledge Bases (KBs) have been built to access a comprehensive collection of facts in a machine readable automatic format. Although these KBs are large, their coverage is far from complete. Most relations between entities are found to be missing in many widely-used KBs. Inference can be used to improve the coverage of such KBs and hence make them more suitable for practical applications like search, dialogue and question answering. This inference process is called Knowledge Base Completion (KBC).

In this dissertation, we analyze existing KBC systems and propose various new KBC methods and models. In particular, we exploit various attributes of KBs — language, structure and temporal attributes to improve KBC performance. We also extensively study the evaluation of KBC models and propose fair evaluation policies.

First, we studied different aspects of modeling KB structure and their impact on KBC performance:

- KBC models can be categorized on the basis of the way they represent entities. Matrix factorization (MF) models have a vector defined for each entity-*pair*, while tensor factorization (TF) models maintain a vector for each *entity*. We compared the effectiveness of the MF and TF paradigms for the general task of KBC. We recognized that special care is needed to handle out-of-vocabulary entity-pairs when evaluating MF against TF. We also propose the first fair unified KBC evaluation protocol to compare MF and TF approaches for KBC.

- Our analysis of KBC models reveals that they often make entity predictions that are incompatible with the type required by the relation. For example, DistMult incorrectly predicts *'Akira Isida'* (type-person) for the query *'Chief Phillips (type-film), released_in_region, ?'*. We propose an unsupervised typing gadget, which enhances KBC models (like DistMult and Complex) with type-compatibility checkers. The enhanced models (TypeDM and TypeComplEx) showed improved KBC performance over the base models. Further analysis revealed that our models better represent the latent types of entities and their embeddings also predict supervised types better than the embeddings learned by baseline models.

While implementing the above models, we found that the norm (L1, L2, or L3) used for regularization or measuring distances, and the rate of negative sampling used to train the models, can have significant consequences for KBC accuracy, sometimes overturning conventional wisdom about how various models compare with each other. Through our investigation of these issues, we implemented a very competitive version of ComplEx KB embedding, better than some follow-up systems.

Next, we study temporal KBs, which associate a relational fact (s, r, o) with a valid set of times (often an instant or interval). We propose TIMEPLEX a Temporal Knowledge Base Completion (TKBC) model, primarily targeted to the link-prediction and time-interval prediction tasks. To the best of our knowledge, this is the first work that predicts the time interval in which the given fact is valid in a general model-independent manner. Also, this is the first work that proposes a time-aware evaluation strategy for TKBC.

Finally, we study Open KBs where entities and relations are represented via textual schema-free strings. Open KBC is generally performed using an inference rule corpus. Using linguistic insights, we develop an algorithm —- Knowledge Guided Linguistic Rewrites (KGLR) — which provides independent verification for statistically-generated Open KB inference rules. The generated high precision rule corpus eventually helps in improving the KBC task performance.

सार

तथ्यों के व्यापक संग्रह को एक मशीन पठनीय स्वचालित प्रारूप मैं लाने के लिए बड़े ज्ञानकोष (नॉलेज बेस - के.बी.) का निर्माण किया गया है। हालांकि ये के.बी. बड़े हैं, लेकिन इनका कवरेज अधूरा है। कई व्यापक रूप से उपयोग किए जाने वाले के.बी. में इकाइयों (एंटिटी) के बीच अधिकांश संबंध (रिलेशन) गायब पाए जाते हैं। ऐसे के.बी. के कवरेज में सुधार के लिए अनुमान (इनफ्रेंस) का उपयोग किया जा सकता है और इससे उन्हें व्यावहारिक अनुप्रयोगों (जैसे खोज, संवाद और प्रश्न उत्तर) के लिए अधिक उपयुक्त बनाया जा सकता है। यह इनफ्रेंस प्रक्रिया को नॉलेज बेस कंप्लीशन (के.बी.सी.) कहा जाता है।

इस थीसिस में, हम मौजूदा के.बी.सी. सिस्टम का विश्लेषण करते हैं और विभिन्न नई के.बी.सी. विधियों का प्रस्ताव करते हैं। विशेष रूप से, हम के.बी.सी. प्रदर्शन में सुधार करने के लिए के.बी. की विभिन्न विशेषताओं का फायदा उठाते हैं - भाषा, संरचना और समय। हम के.बी.सी. मॉडल के मूल्यांकन का भी बड़े पैमाने पर अध्ययन करते हैं और निष्पक्ष मूल्यांकन नीतियों का प्रस्ताव रखते है।

सबसे पहले, हमने के.बी. संरचना के मॉडलिंग के विभिन्न पहलुओं और के.बी.सी.पर उनके प्रभाव का अध्ययन किया।

- के.बी.सी. मॉडल को उनके द्वारा एंटिटीज का प्रतिनिधित्व करने के तरीके के आधार पर वर्गीकृत किया जा सकता है। मैट्रिक्स फ़ैक्टराइज़ेशन (एम.एफ.) मॉडल में प्रत्येक इकाई-जोड़ी (एंटिटी पेयर) के लिए एक वेक्टर परिभाषित होता है, जबकि टेंसर फ़ैक्टराइज़ेशन (टी.एफ.) मॉडल प्रत्येक इकाई (एंटिटी) के लिए एक वेक्टर बनाए रखते हैं। हमने एम.एफ. और टी.एफ. की प्रभावशीलता की तुलना के.बी.सी. में प्रदर्शन से की। हमने माना कि एम.एफ. और टी.एफ. की प्रभावशीलता की तुलना करते समय आउट-ऑफ-वोकैबुलरी इकाई-जोड़े को विशेष देखभाल की आवश्यकता होती है। और हम एम.एफ. और टी.एफ. दृष्टिकोणों की तुलना करने के लिए पहले निष्पक्ष एकीकृत के.बी.सी. मूल्यांकन प्रोटोकॉल का प्रस्ताव करते है।
- के.बी.सी. मॉडल के हमारे विश्लेषण से पता चलता है कि वे अक्सर ऐसी भविष्यवाणी करते हैं जो असंगत होती हैं संबंध द्वारा आवश्यक प्रकार के साथ। उदाहरण के लिए, DistMult गलत भविष्यवाणी करता है - ''अकीरा इसिदा (टाइप-पर्सन)" की, प्रश्न - ''चीफ फिलिप्स (टाइप-फिल्म)", "क्षेत्र में रिलीज", ?' के पूछे जाने पर।

- हम एक गैर-पर्यवेक्षित टाइपिंग गैजेट का प्रस्ताव करते हैं, जो के.बी.सी. मॉडल (जैसे DistMult और ComplEx) को बेहतर करता है टाइप-संगतता चेकर्स के साथ। एन्हांस्ड मॉडल (TypeDM और TypeComplEx) ने बेस मॉडल की तुलना में बेहतर के.बी.सी. प्रदर्शन दिखाया। आगे विश्लेषण से पता चला कि हमारे मॉडल अव्यक्त प्रकार की एंटिटीज का बेहतर प्रतिनिधित्व करते हैं और उनके एम्बेडिंग भी पर्यवेक्षित प्रकारों की भविष्यवाणी बेसलाइन मॉडल द्वारा सीखी गई एम्बेडिंग से बेहतर है ।

उपरोक्त मॉडलों को लागू करते समय, हमने पाया कि नॉर्म के लिए प्रयुक्त मानदंड (L1, L2, या L3) या दूरियों को मापने, और मॉडलों को प्रशिक्षित करने के लिए उपयोग किए जाने वाले नकारात्मक नमूने की दर, केबीसी सटीकता के लिए महत्वपूर्ण परिणाम हो सकते हैं, यह कभी-कभी विभिन्न मॉडलों की एक दूसरे के साथ तुलना के पारंपरिक ज्ञान को उलट देते हैं। इनकी जांच के माध्यम से, हमने ComplEx के.बी. एम्बेडिंग का एक बहुत ही प्रतिस्पर्धी संस्करण लागू किया है।

इसके बाद, हम टेंपोरल के.बी. का अध्ययन करते हैं, जो एक तथ्य (s, r, o) को एक वैध समय के सेट (अक्सर एक पल या अंतराल) के साथ जोड़ते हैं। हम टी.के.बी.सी. के लिए TimePlex मॉडल प्रस्तावित करते है, यह मॉडल मुख्य रूप से लिंक-भविष्यवाणी और समय-अंतराल भविष्यवाणी के लिए लक्षित है। हमारी सर्वोत्तम जानकारी के अनुसार, यह पहला काम है जो समय अंतराल की भविष्यवाणी करता है जिसमें दिया गया तथ्य सामान्य मॉडल-स्वतंत्र तरीके से मान्य है। साथ ही, यह पहला काम है जो टी.के.बी.सी.के लिए समय के प्रति जागरूक मूल्यांकन रणनीति का प्रस्ताव करता है।

अंत में, हम ओपन के.बी. का अध्ययन करते हैं जहां स्कीमा-मुक्त टेक्स्ट के माध्यम से एंटिटीज और रिलेशंस का प्रतिनिधित्व किया जाता है। ओपन के.बी.सी. आमतौर पर एक अनुमान (इनफ्रेंस) नियम कोष का उपयोग करके किया जाता है।

हम एक एल्गोरिथ्म विकसित करते हैं – ज्ञान निर्देशित भाषाई पुनर्लेखन (KGLR) – जो सांख्यिकीय रूप से उत्पन्न ओपन केबी अनुमान नियमों के लिए स्वतंत्र सत्यापन प्रदान करता है। भाषा की अंतर्दृष्टि के प्रयोग से हम एक एल्गोरिथम विकसित करते हैं - ज्ञान निर्देशित भाषाई पुनर्लेखन (के.जी.एल.आर.) -जो सांख्यिकीय रूप से उत्पन्न ओपन के.बी. अनुमान (इनफ्रेंस) नियमों के लिए स्वतंत्र सत्यापन प्रदान करता है। उत्पन्न उच्च परिशुद्धता नियम कॉर्पस अंततः के.बी.सी. कार्य को बेहतर बनाने में मदद करता है।

Contents

List of Figures

List of Tables

Part I

Prologue

Chapter 1

Introduction

The explosion of textual information on the Web provides unprecedented opportunities for acquiring structured knowledge [Etzioni et al., 2004, Etzioni, 2011]. An enormous amount of data — structured as well as unstructured — is readily available to users. Knowledge Bases (KBs) have been built to access this comprehensive collection of facts in a format suitable for manipulation by programs, toward applications like search and question answering.

KBs are collections of facts, about people, things, and places in the world, and relationships between them. Many human-curated and automatically generated KBs have been built, e.g., DBpedia [Auer et al., 2007], YAGO [Suchanek et al., 2007], Freebase [Bollacker et al., 2008], NELL [Carlson et al., 2010], OLLIE KB [Mausam et al., 2012], Google's Knowledge Graph [Singhal, 2012], Bing's Satori [Qian, 2013] and Wikidata [Vrandecic and Krötzsch, 2014]. These KBs are large: OLLIE KB has 5 billion extractions (facts) from over a billion web pages and Google's Knowledge Graph had grown to 500 billion facts on 5 billion entities by May 2020 [1].

KBs contain a wide variety of data types. Apart from relations connecting entities, KBs often include numerical attributes (such as ages, dates, time, financial, and geocoordinate information), textual attributes (such as names, descriptions, and titles/designations) and images (profile photos, flags, posters). These different types of data can play a crucial role as extra pieces of evidence for supporting a fact. E.g., the time information provides additional insights on the validity of temporal facts: (Barack Obama, is President of, USA) is valid only during 2009–2017.

KBs are suitable for various practical applications like semantic search [Bast et al., 2016], question answering systems [Rajpurkar et al., 2016, Unger et al., 2014, Lopez et al., 2013], recommendation systems [Guo et al., 2020], video and text analytics [Suchanek and Preda, 2014, Krishna et al., 2017], and conversational assistants [Adiwardana et al., 2020] like Amazon's

[1] https://en.wikipedia.org/wiki/Google_Knowledge_Graph

Relation	Percentage Unknown
Profession	68%
Place of birth	71%
Nationality	75%
Education	91%
Spouse	92%
Parents	94%
Children	94%
Siblings	96%
Ethnicity	99%

Table 1.1: Incompleteness of Freebase: The table lists the fraction of missing relations (commonly used) of entities (type PERSON) [Min et al., 2013, West et al., 2014]

.

Alexa, Apple's Siri, Google Assistant and Microsoft's Cortana. KBs have found widespread use as a source of distant supervision for a variety of natural language processing (NLP) tasks, such as fine typing [Ling and Weld, 2012], entity linking [Mintz et al., 2009, Fan et al., 2015] and information extraction [Mintz et al., 2009].

KBs often suffer from incomplete coverage. Some KBs are incomplete because they are maintained by human curators, who may miss important facts. Some KBs are extracted from natural language resources and prone to loss of recall. World knowledge is continuously evolving, making it hard to keep track of new updates.

Relations between entities are frequently missing, limiting the performance of systems that rely on the completeness of KBs to any extent. This incompleteness is evident from the percentage of facts missing from the KB. For example, 71% of records about people in Freebase have missing place of birth, 94% have missing parents and 99% have missing ethnicity information. (Table 1.1, compiled from Min et al. [2013], West et al. [2014], lists the fraction of entities of type PERSON who have missing information of nine commonly used relations.) This problem is not specific to Freebase; other knowledge repositories are similarly incomplete.

Another way in which KBs are incomplete is that the vast majority of facts about the world are not encodable in the space of canonical predicates defined in the KB. E.g., Freebase encodes information about which players play for which teams, but not about how many points each player scored in a particular game. This information may be present in natural language text. One approach to bypass this missing fact issue is to allow "open" or "ontology-free" information extraction to build *Open KBs*, where facts are in natural language (and thus have broad coverage) as opposed to the fixed set of entities and relations in structured KBs. However, Open KBs are also prone to incompleteness [Clark et al., 2003] — as they are mostly extracted from natural language resources and the extraction methods are prone to loss of recall. Also some information may not be explicitly mentioned in text, for example, the sentence 'India launched a meteorological satellite into orbit this Wednesday.' suggests to a human reader that (among other things) there was a rocket launch; India probably owns the satellite; the satellite is for monitoring weather; the orbit is around Earth; etc. Hence we need methods to improve the coverage of KBs available in various forms.

This dissertation focuses on new techniques for Knowledge Base Completion (KBC). KBC is the task of automatically inferring missing facts by reasoning about the information already present in the KB. We study two forms of KBC: micro inference and macro inference. We discuss these two tasks in the following section.

1.1 KB inference tasks

KBC is done using a variety of inference tasks. We broadly classify these tasks into two categories. *Macro inference* involves inferring new facts using all known facts from existing KB (it can also be seen as predictive inference). *Micro inference* involves inferring a novel fact from a single input fact, independent of other facts known in the KB (it can also be seen as deductive inference). We elaborate on these tasks in the rest of this section.

1.1.1 Macro inference task

The *macro inference task* involves automatically deriving new facts using all known facts from existing KB. Suppose we observe a tuple (*Barack Obama, was enrolled in, Harvard Law School*). A macro-inference model can use related facts about Barack Obama in the KB, such as (*Barack Obama, alma mater, Harvard Law School*), (*Neil Gorsuch, classmate of, Barack Obama*), (*Neil Gorsuch, was enrolled at, Harvard Law School*), (*Joe Biden, alma mater, Syracuse University*), (*Joe Biden, was enrolled at, Syracuse University*) to infer that this new fact is likely to be true and should be accepted. Notice that the facts relevant to the claim are not only about the entities *Barack Obama* and *Harvard Law School* but could also draw information from other related entities and relations like (*Joe Biden, alma mater, Syracuse University*) and (*Joe Biden, was enrolled at, Syracuse University*). Such an inference provides us a way to "grow" a KB automatically.

With the growing size of KBs, it is important to build a scalable model to infer unseen facts from the facts in KB. Path Ranking Algorithm [Lao and Cohen, 2010] and its variations were popularly used for macro-inference. The algorithm enumerates paths between entity pairs in a KB and use those paths as features to train a model for missing fact prediction. However, most existing PRA based methods suffer from poor scalability (high RAM consumption) [Kyrola, 2013] and feature explosion (trains on an exponentially large number of features) problems. Lately, deep learning models have shown good scalability on macro inference tasks. These methods view KB as a graph (also called Knowledge Graph, KG), where nodes are entities and edges are relations between entities. The methods compute score of a fact (s, r, o) and are evaluated on queries like $(s, r, ?)$ or $(?, r, o)$, where s and o are subject entity and object entity respectively and r is a relation between them. The deep learning methods learn continuous vector representation known as embeddings, which captures semantics of the input. These methods can be classified on the basis of features they use for inference — *paths* over the graph structure, *neighborhood* of query entity, and just the *atomic* triple[2]/fact. All three methods are discussed in detail in Section 2.3.1.

Factorization models are popular atomic models. In factorization models, for a given fact (s, r, o) in which the subject entity s is linked to the object entity o through the relation r, a score for the fact can be recovered as a multilinear product between the embedding vectors of s, r and o, or through more sophisticated composition (scoring) functions. Depending on the way entities are represented, factorization methods can be further subdivided into two broad categories: Matrix Factorization (MF) and Tensor Factorization (TF). MF models have a vector defined for each *entity-pair*, while TF models maintain *entity*-wise factors. Some popular TF models are E [Riedel et al., 2013], TransE [Bordes et al., 2013], DistMult [Yang et al., 2015], ComplEx [Trouillon et al., 2016, Lacroix et al., 2018], Rescal [Nickel et al., 2011], RotatE [Sun et al., 2019a]. MF method includes F [Riedel et al., 2013] and its extensions [Verga et al., 2016].

As the first contribution of the dissertation, in Chapter 4, we study MF's effectiveness for the general task of KBC and compare it against TF models [Jain et al., 2018b]. Note that the number of entity pairs grows quadratically as the number of entities in a KB. Therefore a large number of entity pairs are left unseen in training data. We call them Out-of-Vocabulary (OOV) entity-pairs. We recognize that special care is needed to handle OOV entity-pairs when evaluating MF against TF models, otherwise an MF algorithm may erroneously appear to perform better than it really does. We describe the first unified KBC evaluation protocol that can meaningfully compare MF and TF approaches for KBC. We found that MF models perform much worse than TF models on the task of KBC. Our analysis attributes poor performance of MF models to the high out-of-vocabulary (OOV) rate of entity pairs in test folds of commonly-used datasets. In response, we proposed a TF-augmented MF model. This hybrid model is robust and obtains strong results across various KBC datasets.

In our endeavor to improve the KBC performance, in Chapter 5 of the dissertation, we further analyze the predictions of recent factorization models to find that they often make entity predictions that are incompatible with the type required by the relation and hence we enhance them with type information in an unsupervised manner to improve their performance [Jain et al., 2018a].

[2]a KB fact is referred to as a triple

Type information of entities has shown to be useful for the task of relation extraction [Yaghoobzadeh et al., 2017], search [Metzler and Croft, 2007] and question answering [Welbl et al., 2018]. Our preliminary analysis of KBC model (DistMult (**DM**) and ComplEx (**CX**)) predictions reveal that they make frequent errors by giving high rank to entities that are not compatible with entity types, expected as gold argument for query relation. In **19.5%** of predictions made by DM on FB15K, the top prediction has a type different from what is expected. For a query like '*(Barack Obama, is married to, ?)*', if the model does not know the correct answer, it should still not predict answers of incompatible types like 'onion'. Such type information is sometimes not readily available with KBs. If available, it may be noisy or too fine-grained, limiting its usefulness. Recognizing types of entities is a core NLP problem; in Chapter 5 we explore how to learn types of entities and type signatures of relations in a KB. These *typing gadgets* are learnt without supervision of types of entities and are further used to improve KBC model performance [Jain et al., 2018a].

Further in Chapter 7, we focus on building models which leverage temporal validity of facts, to improve KBC performance [Jain et al., 2018b]. These models can also be used to predict the time-interval at which the fact holds.

Time information: Many relations are transient or impermanent. Temporal KBs annotate each fact (event) with the time period in which it holds or occurs. A person is born in a city in an instant, a politician can be a president of a country for several years, and a marriage may last between years and decades. Temporal KBs represent these by (s, r, o, T) tuples, where T is a span of time. In Chapter 7 we propose a method — TIMEPLEX, for the task of Temporal KBC (TKBC). TIMEPLEX exploits the recurrent nature of some facts/events and temporal interactions between pairs of relations. It is primarily evaluated by link prediction queries $(s, r, ?, T)$ and $(?, r, o, T)$. We also propose time span prediction queries $(s, r, o, ?)$. To the best of our knowledge, this is the first work which predicts the time interval in which a given fact is valid. Also, this is the first work that proposes a time-aware evaluation strategy for TKBC [Jain et al., 2020b] dealing with subtle issues that arise from the partial overlap of time intervals in gold instances and system predictions. The model and the evaluation scheme is described in Chapter7.

In the course of this dissertation work, several KBC models claiming state-of-the-art performance were proposed by various groups. In Chapter 6 we present a detailed study [Jain et al., 2020a] to compare various models across different datasets using the same evaluation framework. We implemented all the models in pytorch. Our study highlights how various recently proposed multiplicative KBC methods (where score of a fact is obtained by taking product of the constituent embeddings), benefit from using a training regime, which uses signal from all entities in loss computation. We find that various multiplicative KBC methods become indistinguishable in terms of performance on most datasets, when using these training regime. Our work calls for a reassessment of the recently proposed models' individual value.

1.1.2 Micro inference task

Micro inference involves inferring a novel fact from a single or a very small number of input facts, independent of other facts seen in the KB. The methods used for this task often rely on inducing inference rules from a corpus of facts [Schoenmackers et al., 2010, Nakashole et al., 2012, Berant et al., 2011, Galárraga et al., 2013, Berant, 2012, Pavlick et al., 2015, Hearst, 1992]. Note that usage of inference rule corpus enables complex reasoning for the validity of a generated fact. For example, given an inference rule corpus which has a rule, ((*X, is the President of, Y*) $\rightarrow$ (*X, is citizen of, Y*)), and a KB which has a fact (*Barack Obama, is the President of, USA*), then a query like (*Barack Obama, is citizen of, ?*) or even a more general query like 'How many people are citizens of USA?' to an inference engine can be correctly answered ('USA' for query 1 and a number ≥ 1 for query 2). Notice that this differs from macro inference which may use a larger neighborhood of related facts for inference.

Building an inference rule corpus, and performing inference using it, are challenging tasks, especially when we have an unbounded number of natural language relations in KBs (like OpenIE KBs[3]). Another challenge in developing such methods is overcoming the problem of language variability: each target meaning can be expressed in natural language in a myriad of ways, which computer programs must learn to recognize. In chapter 8 of the dissertation, we focus on improving the coverage of an OpenIE KB using inference. The task of inference on such KBs relies on (or benefits from) inference rules. These inference rules have a multiplicative impact, and one poor rule could potentially generate many bad KB facts. The research question is how to generate a *high precision* inference rule corpus, which has large number of relations from KB, while handling the *variability* of natural language.

To address the question, we propose an algorithm, Knowledge Graph Linguistic Rewrites (KGLR), which provides independent verification for statistically-generated OpenIE inference rules, using linguistic insights [Jain and Mausam, 2016]. KGLR obtains significant precision improvement on inference rule corpus generated using distributional semantics and multilingual features [Berant, 2012, Pavlick et al., 2015].

Note that the methods discussed in previous sections work well for structured datasets. However, these methods do not perform well on Open KB due to their sparsity and noisy nature [Pujara et al., 2017, Broscheit et al., 2020].

[3]Open Information Extraction (OpenIE) [Mausam et al., 2012] systems generate a schema-free KB where entities and relations are represented via normalized but not disambiguated textual strings. Such OpenIE KBs scale to web.

1.2 Contributions

In summary, we make the following contributions:

MF vs TF We study MF and TF method's effectiveness for the general task of KBC. We recognize that special care is needed to handle OOV entity-pairs when evaluating MF against TF. We also propose the first unified KBC evaluation protocol that can meaningfully compare MF and TF approaches for KBC, in an unbiased manner [Jain et al., 2018b].

Typing Gadget We propose an unsupervised typing gadget, which enhances base models for KBC (DistMult, ComplEx) with two type-compatibility functions, one between r and s and another between r and o [Jain et al., 2018a].

Training KBC models We also performed a comprehensive study which highlights how various multiplicative KBC methods, recently proposed in the literature, benefit from using a training regime, which uses signals from all entities in loss computation and become indistinguishable in terms of performance on most datasets. Our work calls for a reassessment of their individual value, in light of these findings [Jain et al., 2020a].

Temporal KBC (TKBC) : We propose TIMEPLEXa top of the line TKBC model, which is primarily evaluated by link-prediction and time-interval prediction tasks. To the best of our knowledge, this is the first work which predicts time interval in which the given fact is valid. Also, this is the first work which proposes a time aware evaluation strategy for TKBC [Jain et al., 2020b].

Micro Inference in Open KBs We propose Knowledge Graph Linguistic Rewrites (KGLR), which provides independent verification for statistically-generated OpenIE inference rules, using linguistic insights. This eventually helps in improving the precision of the rule corpus [Jain and Mausam, 2016].

Work done in Chapter 4 was done in collaboration with Shikhar Murty. Work done in Chapter 5 was done in collaboration with Pankaj Kumar. Work done in Chapter 6 and 7 was done in collaboration with Sushant Rathi. In each case, the part done by the collaborators appeared in their respective bachelor's theses.

1.3 Outline of the dissertation

The dissertation is organized into four parts: Part I sets up the introduction (Chapter 1) and discusses related work (Chapter 2). Parts II and III of the dissertation contains novel contributions of this dissertation. In Chapter 3 (Part II), we describe Macro Inference more formally, describe the datasets used for experiments and also touch upon evaluation strategy for macro inference models. In Chapter 4, we compare then popular Matrix Factorization models with Tensor Factorization models; we also propose an evaluation metric for a fair comparison of the two. Then we propose a joint MF-TF model which performs well or atleast as good as the best model on KBC datasets. In Chapter 5, we analyse the predictions of recent KBC models and propose an unsupervised typing gadget, which enhances KBC models (DistMult, ComplEx). In Chapter 7, we discuss KBC methods for temporal KBs and also propose a time-aware evaluation strategy. In Chapter 6, we perform an extensive re-examination of recent KBC techniques and find that multiplicative models significantly benefit from signals from all entities during training, making the relative performance gaps between different models trained in this manner small. This calls for reassessment of their individual value. In *Part* III of the dissertation we formally define the micro inference task and propose methods for inference in Open KBs. In *Part* IV we conclude, summarizing what we have learned and offer suggestions for future work.

Chapter 2

Related Work

This chapter will discuss how our work on Knowledge Base Completion (KBC) relates to other research efforts in similar directions. It first presents an overview of knowledge bases and then discusses related work on the Macro Inference task and the Micro Inference Task. Note that related work of a specific topic appears in the corresponding chapters.

2.1 Knowledge Bases

A Knowledge Base (KB) is a collection of machine-readable facts about the real world. The first large manually compiled general-purpose KB construction project, aiming to assemble human knowledge, dates back to the 1990s — The Cyc project [Lenat, 1995] and the WordNet project [Hirst and St Onge, 1998]. Similar projects started in 2000s are DBpedia [Auer et al., 2007], Freebase [Bollacker et al., 2008], KnowItAll [Etzioni et al., 2005], WebOf-Concepts [Dalvi et al., 2009], WikiTaxonomy [Ponzetto and Strube, 2007], YAGO [Suchanek et al., 2007], BabelNet [Navigli and Ponzetto, 2012], ConceptNet [Speer and Havasi, 2012], KnowledgeVault [Dong et al., 2014], NELL [Carlson et al., 2010], Probase [Wu et al., 2012], Wikidata [Vrandecic and Krötzsch, 2014] and many more.

Previous works ([Nickel et al., 2015]) classified KBs on the basis of the methods used to curate them: (1) In curated approaches, highly accurate KBs are built manually by a closed group of experts. The Cyc project [Lenat, 1995] and the WordNet project [Hirst and St Onge, 1998] involved manual curation. (2) In collaborative approaches, triples are created manually by an open group of volunteers. KBs like Freebase [Bollacker et al., 2008] and Wikipedia built this way scales well. (3) In automated semi-structured approaches, triples are extracted automatically from semi-structured text (e.g., infoboxes in Wikipedia) via hand-crafted rules, learned rules, or regular expressions. YAGO [Suchanek et al., 2007] and DBpedia [Auer et al., 2007] are accurate yet inclomplete KBs built this way. (4) In automated unstructured approaches, triples

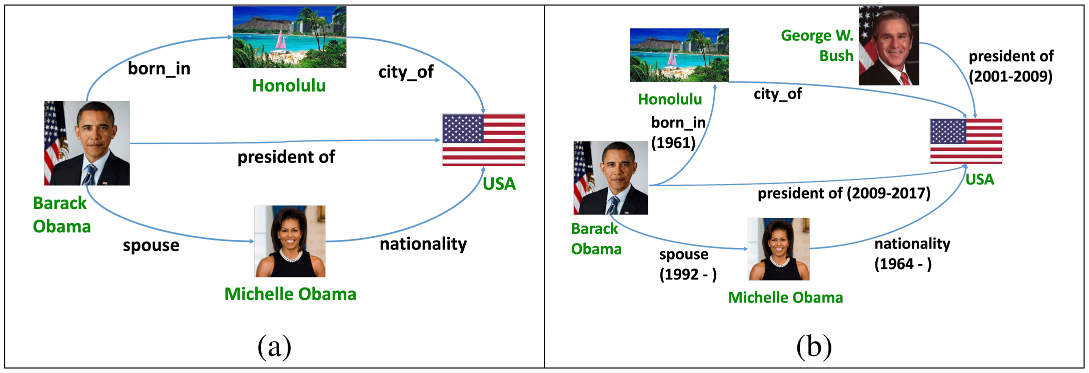


Figure 2.1: Knowledge graph fragments — (a) Regular KG (b) Temporal KG

are extracted automatically from unstructured text via machine learning and natural language processing techniques. Example the KnowledgeVault project [Dong et al., 2014] and the NELL [Carlson et al., 2010] project.

KBs vary widely in terms of the complexity and rigidity of their schemata, and the support they provide to automatic inference mechanisms. At one extreme, support may be provided for propositional logic with quantifiers, using a powerful theorem prover. At the other extreme, a 'KB' may be merely sets of object or entity names collected into types or categories. In some KBs, a 'fact' may be a full-fledged row in a table that is a member of a complete relational schema. In other KBs, the 'schema' may be simplified to just a single central table with three columns: subject, relation and object (loosely following the RDF standard[1]). Rows in this 3-column table are called *triples*. Such a KB may be regarded as a directed graph with typed edges (that represent relations) connecting nodes representing entities. In such a context, the KB may be called a Knowledge Graph or KG. Much of our work here concerns KGs, so we will refer to KGs and KBs interchangeably throughout. The 3-column 'schema' cannot naturally support relations with arity greater than 3. E.g., a fact may be associated with a period of validity — (*Barack Obama, president of, USA; 2009–2016*). One way to represent such a *temporal KG* is to associate each edge with not just a relation type but also start and end time epochs. A 'row' in such a 'table' is called a *tuple*. Figure 2.1 shows various examples illustrating the above notions.

KBs, by representing world knowledge in structured forms, have shown to be beneficial for low-level NLP tasks like prepositional phrase attachment [Nakashole and Mitchell, 2015], co-reference resolution [Rahman and Ng, 2011], named entity recognition [Ratinov and Roth, 2009], machine reading [Yang and Mitchell, 2017] and many more. As discussed in the introduction (Chapter 1), KBs are useful for an array of practical applications including semantic search [Bast et al., 2016], question answering systems [Unger et al., 2014], recommendation

[1]https://www.w3.org/TR/rdf-syntax-grammar/

systems [Guo et al., 2020], video and text analytics [Suchanek and Preda, 2014, Krishna et al., 2017], conversational assistants [Adiwardana et al., 2020] like chatbots — Amazon's Alexa, Apple's Siri, Google's Assistant and Microsoft's Cortana.

2.2 Open vs. Closed KBs

In 'open' KBs, names of entities and relations may be unrestricted text strings. The above example looks like an open KB fact. In 'closed' KBs, entities and relations are *canonicalized* to standardized IDs, and one or more textual aliases are provided. E.g., in Wikidata, the unique ID Q76 has been assigned to Barack Obama. In case a second Barack Obama is added to Wikidata, that person will be assigned a different ID. That entity will be connected to entirely different entities and relations in general. The entity Q76 has many *aliases*, perhaps over multiple languages. In English alone, Wikidata has registered aliases "Barack Hussein Obama II", "Barry Obama" and "BHO", among others. As of the time of writing, 235 aliases are listed in multiple languages. A canonicalized triple involving Obama in Wikidata is (*Q76, P103, Q1860*). Here the relation (a.k.a. 'property') P103 has English alias "native language" and the object entity Q1860 is the English language. Wikidata specifies about 7000 relations but lacks many potentially useful ones; e.g., in the context of musicians and songs, performedAt, coveredArtist, duetWith, songAbout, etc. might be very useful.

It may be useful to visualize the gradual structuring and canonicalization of knowledge by tracing the progression from free text to open KB records to closed KG tuples. The first step of distilling open tuples from free text is called Open Information Extraction (OpenIE). OpenIE methods automatically discover and extract all entities and relations of interest from input natural language sentences (perhaps from a large Web corpus). REVERB [Fader et al., 2011], OLLIE [Mausam et al., 2012] and OPENIE4 [Mausam, 2016] are three of many OpenIE systems that have been developed to generate Web-scale open KBs. The problem with such KBs is that their entities and relations are not canonicalized, which leads to the storage of redundant and ambiguous facts. For example, an open KB storing (*Barack Obama, was born in, Honolulu*) and (*Obama, is a native of, Honolulu*) does not record that Barack Obama and Obama refer to the same entity. Similarly, "was born in" and "is a native of" also refer to the same relation. Historically, many closed KBs were created before work on OpenIE took off. Nevertheless, inspection of high-confidence and high-support open tuples may trigger human curators to assign canonical IDs and create tuples in closed KBs as well.

2.3 Incompleteness and inference

OpenIE systems have been designed to continually scrape the Web corpus, harvesting new tuples and modifying or enhancing old ones. Despite such efforts, they continue lagging behind the production of new relational world knowledge. Canonicalization is an even slower process, because it need further human involvement, sometimes from experts in ontology management.

Consequently, automatic KB completion (variously called KG completion or KB inference) — the task of inferring facts missing from an incomplete KB — has witnessed vigorous investigation in recent years. In the rest of this chapter, we will discuss popular methods for macro and micro inference.

2.3.1 Macro inference

As discussed in section 1.1, *macro inference task* automatically deriving new facts using all known facts from existing KBs. Suppose we want to validate if '(*Barack Obama, has nationality, USA*)' and we are given a KB seen in Figure 2.1. A macro-inference model can use related facts about Barack Obama in the KB (such as (*Barack Obama, born in, Honolulu*), (*Barack Obama, president of, USA*), (*Barack Obama, spouse of, Michelle Obama*), (*Michelle Obama, nationality, USA*), (*Honolulu, city of, USA*)) to infer that this new fact is likely to be true and should be accepted. This provides us a way to "grow" a KB automatically.

We organize macro inference methods for KBC into three categories, based on the features they use for inference — Path based, Neighborhood based and Atomic Triple/fact based (See Figure 2.2) inference.

2.3.1.1 Path-based inference

These methods encode the knowledge base as a graph and leverage path (from query entity) information over the graph structure for inference. Preliminary work on Path-based inference

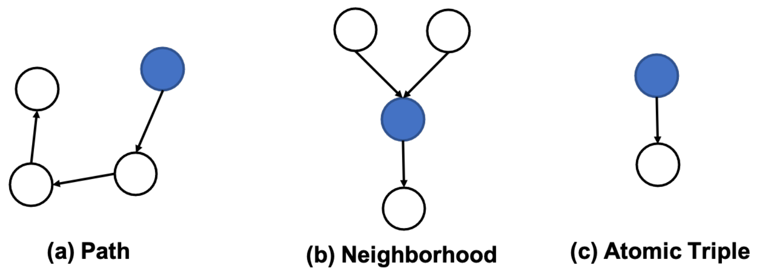


Figure 2.2: Macro Inference Task classified on the basis of features they use for inference: (a) Path over the graph structure, (b) Neighborhood of the query entity, (c) Atomic triple/fact. Note that the node highlighted in blue is query entity.

widely investigated Page Ranking Algorithm (PRA), which uses random walks to find paths that connect the source and target nodes of relation instances. These paths are used as features in a logistic regression classifier that predicts new instances of the given relation. Lao and Cohen [2010], Lao et al. [2011] uses random walk with restart mechanism on a knowledge graph to reliably infer new beliefs/facts for the knowledge base. Gardner et al. [2014] revisited the PRA and used pre-trained vector representations of relations and hence compute feature similarity in vector space. Neelakantan et al. [2015] use RNNs to obtain a dense representation of multi-hop paths for knowledge base completion; this also enabled zero-shot learning. Chains-of-Reasoning [Das et al., 2016] further added a neural attention mechanism to obtain improved performance. DeepPath [Xiong et al., 2017], MINERVA [Das et al., 2018], DIVA [Chen et al., 2018], Multi-Hop [Lin et al., 2018], M-Walk [Shen et al., 2018] and CPL [Fu et al., 2019] use reinforcement learning based approaches to explore paths in knowledge graphs.

2.3.1.2 Neighborhood-based inference

These methods leverage the neighborhood of the query entity (forming a graph) for inference. Graph Convolution Networks (GCN) are specialized neural networks for graphs [Bruna et al., 2014]. Therefore, GCNs become a natural building block for KBC models. An end-to-end model for KBC consist of an encoder (based on GCN) and a decoder (based on atomic models). Encoder uses a GCN to generate entity embedding by encoding its neighborhood. This embedding is then fed into a decoder to generate fact score. Atomic models are popularly used as decoders. Historically, GCNs are a generalization of Convolution Neural Networks (CNNs). An extension of GCNs for relational graphs is proposed by Marcheggiani and Titov [2017] but they ignored relations, later R-GCN [Schlichtkrull et al., 2018] extended the model and proposed relation specific filters (transformation). Intuitively, the R-GCN encoder accumulates transformed feature vectors of neighboring nodes through a normalized sum. R-GCN decoder uses the node embeddings to reconstruct a graph edge and hence can make link prediction. Weighted GCN [Shang et al., 2019] utilizes learnable relational specific scalar weights during GCN aggregation. Vashishth et al. [2020], Ye et al. [2019] further demonstrated improved performance by encoding the nodes and relations of the graph.

2.3.1.3 Atomic Triple based inference

These methods leverage the fact atomically for inference. Atomic model due to their simplicity are a popular choice for KBC models. Factorization models are used for atomic fact-based inference. Factorization models were initially popularized by the Netflix challenge [Koren et al., 2009]. In these models, a partially-observed matrix or tensor is decomposed into a product of embedding matrices with much smaller dimensions. The resultant low dimensional embeddings are viewed as vector representations for each entity and relation in the graph. When combined back, allow completion of the missing entries in the original partially complete matrix and tensor. For a given fact (s, r, o) in which the subject entity s is linked to the object entity o through the relation r, a score for the fact can be recovered as a multilinear product between the embedding vectors of s, r and o, or through more sophisticated composition (scoring) functions (denoted by ϕ). Various parametric forms of the function ϕ have been designed. We discuss the best-known ones, organized into two categories. Some of these scoring functions are summarized in Table 2.1 for ready reference.

Model (M)	Scoring function (ϕ^M)
TransE [2013]	$-\left\|\boldsymbol{e}_s + \boldsymbol{r} - \boldsymbol{e}_o\right\|_1$
RotatE [2019a]	$-\left\|\boldsymbol{e}_s \bullet \boldsymbol{r} - \boldsymbol{e}_o\right\|_1$
E [2013]	$\boldsymbol{v}_r^\top.\boldsymbol{e}_s + \boldsymbol{w}_r^\top.\boldsymbol{e}_o$
DistMult [2015]	$\langle \boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o \rangle$
ComplEx [2016]	$\mathbb{R}\langle \boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o^\star \rangle$
SimplE [2018]	$\frac{1}{2}(\langle \boldsymbol{h}_s, \boldsymbol{r}, \boldsymbol{l_o} \rangle + \langle \boldsymbol{l_o}, \boldsymbol{r}^{-1}, \boldsymbol{h}_s \rangle)$
TuckeR [2019]	$\mathcal{W} \times \boldsymbol{e}_s \times \boldsymbol{r} \times \boldsymbol{e}_o$
F [2013]	$\boldsymbol{r}^\top \cdot \boldsymbol{ep}_{so}$

Table 2.1: Scoring functions of several KBC models. First 1-2 rows list translation models, row 3-8 lists bilinear models . Note that row 8 is a matrix factorization model, while row 1-7 are tensor factorization models. Larger value of ϕ^M implies more confidence in the validity of the triple. $\bullet$ denotes rotation . $\star$ denotes the complex conjugate. $\mathbb{R}$ refers to the real part of the complex valued score returned by the ComplEx model.

Translation and rotation In these models, relations are represented as translation or rotation from the subject entity embedding to the object entity embedding. Each entity e (respectively, relation r) is represented as a vector $\boldsymbol{e}_s, \boldsymbol{e}_o, \boldsymbol{r}$. The relation transformed subject entity embedding is combined with the object entity embedding using an additive function. Row 1 and 2 of Table 2.1 represents translation and rotation model respectively. In TransE [Bordes et al., 2013], if (s, r, o) holds, then $\boldsymbol{e}_o$ should be close to $\boldsymbol{e}_s$ plus $\boldsymbol{r}$. TransE aims to model inversion and composition pattern. While RotatE [Sun et al., 2019a] models inversion, composition and symmetry.

Multiplication In multiplicative models [Riedel et al., 2013, Yang et al., 2015, Trouillon et al., 2016, Lacroix et al., 2018, Jain et al., 2018a, Balažević et al., 2019, Kazemi and Poole, 2018], the score of a fact (s, r, o) is obtained by taking a product of subject, relation and object embeddings. In Model E [Riedel et al., 2013], relation r is represented by two vectors $\boldsymbol{v}_r, \boldsymbol{w}_r \in \mathbb{R}^D$, and the tuple score $\phi^{\mathrm{E}}(s, r, o) = \boldsymbol{e}_s \cdot \boldsymbol{v}_r + \boldsymbol{e}_o \cdot \boldsymbol{w}_r$. The ComplEx [Trouillon et al., 2016] model is a popular multiplicative model, in this text we abbreviated it as CX. The model embeds s, r, o to vectors in a complex vector space, $\boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o \in \mathbb{C}^D$. CX defines the score ϕ of a fact (s, r, o) as $\mathbb{R}(\langle \boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o^\star \rangle)$ where $\langle \boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o \rangle = \sum_{d=1}^{D} \boldsymbol{e}_s[d]\ \boldsymbol{r}[d]\ \boldsymbol{e}_o^\star[d]$ is a 3-way inner product, $\boldsymbol{e}_o^\star$ is the complex conjugate of $\boldsymbol{e}_o$, and $\mathbb{R}(c)$ is real part of $c \in \mathbb{C}$. Such a composition of complex embeddings can handle a large variety of binary relations, among them symmetric and antisymmetric relations. In real valued case (i.e. embeddings of s, r, and o are real), CX model corresponds to DistMult model (see row 4 in Table 2.1). One may feel that the complex model gains over DistMult due to the use of larger embeddings. However, that is not the case. The main benefit from ComplEx comes from conjugating the object vector elements, thus allowing the model to capture asymmetry and antisymmetry in relations. DistMult, which uses real embedding, making the score function symmetric. It will give the same score to the facts – ⟨Joe Biden, fatherOf, Hunter Biden⟩ and ⟨Hunter Biden, fatherOf, Joe Biden⟩, which is clearly incorrect.

For SimplE [Kazemi and Poole, 2018], $\boldsymbol{h}_s \in \mathcal{H}$ and $\boldsymbol{l_o} \in \mathcal{L}$, where $\mathcal{H}$ and $\mathcal{L}$ are separate entity ($\mathcal{E}$) representation for entities in head and tail position respectively. It also learns seperate representation for $\boldsymbol{r} \in \mathcal{R}$ and its reciprocal $\boldsymbol{r}^{-1} \in \mathcal{R}^{inv}$. Lacroix et al. [2018] proposed a popular variant of CX model that we call ComplEx-N3 (CX-N3). This model uses a weighted L3 regularization in a modified training objective to accommodate reciprocal relations. F model has a vector representation of every entity pair, $\boldsymbol{ep}_{so} \in \mathbb{R}^D$ and scores a tuple: $\phi^{\text{F}}(s, r, o) = \boldsymbol{r}^\top \cdot \boldsymbol{ep}_{so}$.

Path based and neighborhood based inference methods generalize poorly (performance on KBC is poor) as compared to latest atomic triple based methods, which score a fact by only using the embeddings of subject, object and relation of the fact. Hence in this dissertation we will focus on atomic triple based inference methods.

2.3.2 Micro inference

Micro inference involves inferring a novel fact from a single or a very small number of input facts, independent of other facts seen in the KB [Schoenmackers et al., 2010, Nakashole et al., 2012, Berant et al., 2011, Galárraga et al., 2013, Berant, 2012, Pavlick et al., 2015, Hearst, 1992, Carlson et al., 2010]. For example, given an inference rule corpus which has a horn clause rule[2] $(BornIn(Person, City) \wedge CityOf(City, Country) \implies CitizenOf(Person, Country))$, and a KB which has facts (*Barack Obama, BornIn, Honolulu*) and (*Honolulu, CityOf, USA*), then a query like (*Barack Obama, is the citizen of, ?*) or even a more general query like ‘How many people are citizens of USA?’ to an inference engine can be correctly answered (‘USA’ for query 1 and a number ≥ 1 for query 2). The methods used for this task often rely on inducing inference rules from a corpus of facts. Note that usage of an inference rule corpus enables

[2]A Horn clause is a clause (a disjunction of literals) with at most one positive, i.e. unnegated, literal.

the possibility of complex reasoning for verifying a generated fact. We now discuss various traditional methods to learn inference rules.

2.3.2.1 Distributional Features

Inference rules are predominantly generated via extended distributional similarity — two relation pair having a high degree of argument overlap are similar, and thus are a candidate for a unidirectional inference rule or a paraphrase (bidirectional). The rule generation methods vary on the base representation of relations, e.g., KB relations [Galárraga et al., 2013, Grycner et al., 2015], Open IE relation phrases [Schoenmackers et al., 2010], syntactic-ontological-lexical (SOL) patterns [Nakashole et al., 2012], and dependency paths [Lin and Pantel, 2001]. Distributional similarity is modeled by symmetric as well as asymmetric measures.

Symmetric similarity measure: Lin and Pantel [2001] proposed the DIRT algorithm which is based on Lin similarity measure. For each relation r, the algorithm computes two sets of features, F_s^r and F_o^r, the nouns that instantiate the arguments s and o respectively in a large corpus. Every $f_s \in F_s^t$ is weighted by the Pointwise Mutual Information (PMI) between the relation and the feature. PMI is computed as: $w_s^r(f) = \log \frac{Pr(f_s|r)}{Pr(f_s)}$. Given two relations r_1 and r_2, Lin is computed for s as follows:

$$Lin_s(r_1, r_2) = \frac{\sum\limits_{f \in F_s^{r1} \cap F_s^{r2}} [w_s^{r_1}(f) + w_s^{r_2}(f)]}{\sum\limits_{f \in F_s^{r1}} w_s^{r_1}(f) + \sum\limits_{f \in F_s^{r2}} w_s^{r_2}(f)}$$

We can compute $Lin_o(r_1, r_2)$ similarly. Finally, $DIRT(r_1, r_2) = \sqrt{Lin_o(r_1, r_2)\dot{L}in_s(r_1, r_2)}$. These methods are more appropriate for detecting paraphrases than for entailment.

Asymmetric similarity measure: Almost all asymmetric (directional) distributional similarity approaches are based on the intuition that semantically-general relations occur in more contexts than semantically-specific relations. Thus, if the contexts of a relation r_1 is properly included in the contexts of another relation r_2, then $r_2 \rightarrow r_1$. One probabilistic interpretation is: r_2 probabilistically entails r_1 if: P(r_1 is true $\mid r_2$) $>$ P(r_1 is true), where P(r_1 is true $\mid r_2$) can be interpreted as entailment confidence. While learning Horn clauses, (Body $\implies$ Head) : ((Company, IsBasedIn, City) $\wedge$ (City, IsLocatedIn, Sate) $\implies$ (Company, IsHeadquarteredIn, State)) [Schoenmackers et al., 2010], r_2 can be safely replaced by body of the rule and r_1 with the head. Previously, Schoenmackers et al. [2010] use such a statistical significance/statistical relevance based approach. However Nakashole et al. [2012], Berant [2012], Galárraga et al. [2014], Pavlick et al. [2015] exploit support statistics. These methods are more suited for entailment as compared to the symmetric methods.

Distributional similarity methods have *high coverage* and are *domain independent*, and *unsupervised*. However these approaches have some fundamental limitations. First, they miss out on important recall because obvious commonsense facts are never stated explicitly in text, e.g., (s, married to, o) $\rightarrow$ (s, knows, o), will likely be missed — text will rarely say that a couple knows each other. Second, they are consistently affected by statistical noise and end up generating a wide variety of inaccurate rules. These methods sometimes result in distributionally similar antonyms (e.g., (Stock, rise in, January) $\rightarrow$ (Stock, fall in, January)) or terms that are mutually exclusive (e.g., boys and girls). Thirdly, these methods are blind to context. Hence techniques based only on distributional semantics do not perform well.

2.3.2.2 Multi-Lingual features

Distributional similarity methods use monolingual corpora statistics to generate paraphrases. This task is also performed using commonly available bilingual parallel corpora. Alignment techniques from phrase-based statistical machine translation are used; paraphrases in one language are identified using a phrase in another language as a pivot.

Multi-Lingual pivoting: Two relation phrases are considered paraphrases if they share a foreign translation [Bannard and Callison-Burch, 2005, Pavlick et al., 2015]. The most likely paraphrase (r̂) for a given relation phrase (r_1) is extracted as r̂= $arg \max_{r_1 \neq r_2} p(r_2|r_1) = arg \max_{r_1 \neq r_2} \sum_f p(f|r_1)p(r_2|f)$, where f and r are different language strings and $p(r_2|r_1)$ is the confidence that r_1 is a paraphrase of r_2. Table 2.2 compiled from [Bannard and Callison-Burch, 2005], lists a set of example paraphrase extracted from gold standard alignments.

Under control	Checked, curb, curbed, *in check*, limit, slowdown
At work	at the workplace, employment, in the work sphere, operate, held, holding, organized, taken place, took place, *working*
Sooner or later	*At some point*, eventually
Green light	Approval, call, *go-ahead*, indication, message, sign, signal, signals, formal go-ahead

Table 2.2: Sample paraphrases extracted from a manually aligned parallel corpus. Italicized words are best paraphrases according to paraphrase probability estimates [Bannard and Callison-Burch, 2005].

Such methods utilize the abundant bilingual parallel data to generate paraphrases. The methods are complementary to monolingual distributional methods. Multilingual pivoting approaches are adversely affected by *polysemy* in a foreign language. For example, when using an aligned parallel corpus of Hindi and English, we observe 'Yesterday' gets translated to 'कल ' which generates 'Tomorrow' in turn: Yesterday $\rightarrow$ कल $\rightarrow$ Tomorrow. Moreover, these methods can group morphological variants of a foreign language into the same paraphrase. For example: बैठ $\rightarrow$ sitting $\rightarrow$ बैठी (on using an aligned parallel corpus of Hindi and English).

2.3.2.3 Textual Patterns

Instead of comparing the typical contexts in which a pair of relations appear in a large corpus, co-occurrence methods focus on the co-occurrence of the pair of elements in a local scope such as a sentence. For example, from the sentence ‘He scared *and even* startled me’ one might infer that ‘startle’ is semantically stronger than ‘scare’ and thus ‘startle $\rightarrow$ scare’. Learning semantic relations using co-occurrence was first articulated in Hearst’s work [Hearst, 1992]. Hearst [1992] discussed methods for automatic acquisition of the *hyponym* or *is-a* relation between nouns from large corpora. The key insight is that the semantic relation is manifested in template patterns, for example the patterns ‘NP_y such as NP_x’ or ‘NP_x or other NP_y’ often imply that NP_x is a kind of NP_y (NP stands for Noun Phrase). This work was extended to acquire meronymy, equivalence, antonymy, similarity and hyponymy relations [Girju et al., 2006, Chklovski and Pantel, 2004]. Table 2.3 specifies sample pattern groups and provides one example pattern for each group.

Hypernymy (X $\rightarrow$ Y)	Delhi *and other* cities.
Hyponymy (X $\leftarrow$ Y)	Action-adventure games *including* GTA; planets *such as* Mercury
Equivalence (X $\equiv$ Y)	Vitamin D *known as* sunshine vitamin
Enablement	*to* win *by* fight*ing the*
Strength	hurt *and even* kill
Antonymy	*either* open *or* close
happens-before	*to* marry *and subsequently* divorce

Table 2.3: Sample textual patterns/path features (shown in italics), for rule extraction.

Naturally, these methods suffer from *low coverage*. Also, they are more *suited for nouns* than for verbs, since nouns tend to co-occur in sentences more often than verbs. Note that these rules are hand-built and generally have low recall.

For scalability [Niu et al., 2011, Domingos and Webb, 2012], the inference rule corpus generated using the above techniques is used along with probabilistic models such as Markov Logic Networks (MLN) [Schoenmackers et al., 2008] or Bayesian Logic Programs (BLP) [Raghavan et al., 2012] to produce human-interpretable proof chains. These inference methods are bound by the coverage and quality of the inference rules [Clark et al., 2014].

More recently, Rocktäschel et al. [2015], Guo et al. [2016], Demeester et al. [2016], Guo et al. [2018], Minervini et al. [2017b, 2020, 2021] proposed methods that incorporate first-order logic rules into KB relation and entity embeddings. IterE [Zhang et al., 2019] later jointly learns embedding as well as a rule learning system to enhance link prediction performance further. Neural Theorem Provers (NTP) [Rocktäschel and Riedel, 2017] and Neural LP [Yang et al., 2017] learn logic rules that are trained via end-to-end gradient based learning. Methods [Qu and Tang, 2019] which jointly learn the embeddings with a Markov Logic Network, also have the power to capture the uncertainty of logic rules.

In this chapter, we discussed various popular macro and micro inference methods. Next, we present the methods we propose for improving macro and micro inference performance.

Part II

Macro Inference

Chapter 3

Macro inference preliminaries

In this chapter, we will review macro inference methods to improve the coverage of an incomplete KB. We formally define the macro inference task and then describe standard datasets used to design and evaluate macro inference methods. Then we discuss commonly used loss functions and evaluation protocols.

3.1 Problem formulation

Macro inference task automatically derive new facts using all known facts from existing KBs. The methods generally rely on complete KB statistics for inferring novel facts. One approach is to learn a representation of real-world entities and the relations between them in a knowledge base; this enables inference of new facts from the knowledge base.

A fact tuple represents real-world knowledge in the form (*subject, relation, object*) or symbolically, (s, r, o). A concrete example is (*Narendra Modi, is the PM of, India*). We will often write readable strings in our examples, but the understanding is that s, r, o are canonical IDs. Macro inference is usually cast as a supervised classification or ranking problem. For this purpose, we will consider facts to come from two classes: valid and invalid, denoted by the variable $y \in \{0, 1\}$ (0 if not valid, 1 if valid). E.g., (*Barack Obama, is president of, India*) is an invalid tuple.

We are given an incomplete KB with entities $\mathcal{E}$ and relations (relation types) $\mathcal{R}$. Training data is presented as a set of N labeled training tuples

$$\mathcal{T}_{tr} = \Big\{ \Big((s_n, r_n, o_n); l_n = 1 \Big) : n \in [N] \Big\} \tag{3.1}$$

where $s, o \in \mathcal{E}$, $r \in \mathcal{R}$ and $[N] = \{1, 2, \ldots, N\}$. A validation fold $\mathcal{T}_v$ with similar labeled tuples may be provided for system tuning, after which it will be evaluated on a test fold $\mathcal{T}_{ts}$. Usually, KBs have only valid tuples, i.e., all $l_n = 1$, and algorithms must design various *negative sampling* strategies to obtain invalid tuples for effective training. The goal of the inference task is to predict the validity of any tuple not present in $\mathcal{T}_{tr}$. Note that micro-inference methods can also do the KBC task. But we loosely refer to macro inference methods as KBC models in this part of the dissertation. This dissertation focus on 'Atomic' KBC models, which fit continuous representations (loosely "embeddings") to entities and relations so that the belief in the veracity of (s, r, o) can be estimated as an algebraic expression (called a scoring function ϕ) involving those embeddings.

3.2 Evaluation protocol

To run the experiments at scale, macro-inference methods generally follow an automatic evaluation protocol. In this method the KB is split into train ($\mathcal{T}_{tr}$), validation ($\mathcal{T}_v$) and test ($\mathcal{T}_{ts}$) tuples. The system can access only $\mathcal{T}_{tr}$ during training. The validation set $\mathcal{T}_v$ is used for tuning the hyper-parameters of the system. For each test tuple, $(s^*, r^*, o^*) \in \mathcal{T}_{ts}$, query $(s^*, r^*, ?)$ and $(?, r^*, o^*)$ are issued to the trained model M. For query $(s^*, r^*, ?)$, the model ranks all entities $o \in \mathcal{E}$ by decreasing $\phi^M(s^*, r^*, o)$. A higher rank of o^* in this list suggests a better performance of the model. Common metrics used to compare algorithms are mean reciprocal rank (MRR) and the percentage of o^*s obtained in top k (HITS@k) results.

$$MRR = \frac{1}{|N|} \sum_{i=1}^{N} \frac{1}{rank_i} \tag{3.2}$$

These metrics are indicative but can be flawed: some of the os ranked higher than o^* may yield correct tuples (s^*, r^*, o). For the test query (*IIT Delhi, is alma-mater of, ?*) the gold answer of interest is *Padmasree Warrior*. But the query has more than one correct answer — *Vinod Khosla, Deepinder Goyal, Kiren Bedi, Raghuram G. Rajan, Mausam* and many more — which may appear above the answer of interest. It is unfair to penalize the model for predicting these. Some of the possible correct tuples might be seen in training set, validation set or test set. Hence, the *filtered* metrics remove the set $\{o|(s^*, r^*, o) \in \mathcal{T}_{tr} \cup \mathcal{T}_v \cup \mathcal{T}_{ts}\}$ from the ranked list [Bordes et al., 2013], thus creating a better evaluation metric. In the following chapters, we refer to the original metric as raw, while we refer to the newer as filtered.

3.3 Popular datasets

Many knowledge bases have been constructed, in recent times - WordNet, DBpedia, YAGO, and Freebase (now Google Knowledge Graph) are a few popular KBs. (a) Freebase[1] is a huge KB of general facts, with around 1.2 billion tuples and more than 80 million entities. (b) WordNet[2]

[1] developers.google.com/freebase
[2] wordnet.princeton.edu/download

is a lexical database of semantic relations between words. WordNet links words with semantic relations like synonyms[3], hyponyms[4], and meronyms[5]. WordNet has 1,55,327 words organized in 1,75,979 synsets for a total of 2,07,016 word-sense pairs. (c) YAGO[6] (Yet Another Great Ontology) is a large open-source knowledge base with general knowledge about people, cities, countries, movies, and organizations. YAGO3 knows more than 10 million entities and contains more than 120 million facts about these entities. (d) DBpedia[7] KB describes 6.0 million entities, out of which 5.2 million are classified in a consistent ontology, including 1.5M persons, 810k places, 135k music albums, 106k films, 20k video games, 275k organizations, 301k species, and 5k diseases.

Most KBC systems have used one or more of six popular datasets (subset of the KBs discussed above) for evaluation. This includes two subsets of Freebase. Bordes et al. [2013] built a subset of Freebase dataset, *FB15k*. It has entities and relations which are seen atleast 100 times in the original Freebase corpus and are also present in Wikilinks database. The resultant corpus is randomly split as shown in Table 3.1. Toutanova et al. [2015] found that FB15k has many test tuples which can be obtained by simply inverting training tuples. For example, the test set frequently contains tuples such as $(s$, hyponym, $o)$ while the training set contains its inverse $(o$, hypernym, $s)$. Hence they created a more difficult dataset, *FB15k-237*, where inverse relations are removed. Two subsets of Wordnet are also introduced. The *WN18* dataset has 18 relations scraped from WordNet for roughly 41,000 synsets, resulting in 1,41,442 tuplets [Bordes et al., 2013]. Dettmers et al. [2018] highlighted that a large number of the WN18 test tuples can be found in the training set with another relation or the inverse relation. Therefore, they introduced a new version of the dataset, *WN18RR* where inverse relations are removed. They also proposed a new dataset YAGO3-10. It is a subset of YAGO3 with entities that have a minimum of 10 relations each. Most of the tuples deal with descriptive attributes of people, such as citizenship, gender, and profession.

Riedel et al. [2013] proposed a dataset that has textual facts with aligned structured facts (structured KB facts) — NYT+FB. The textual data is extracted from the NYTimes corpus [Sandhaus, 2008] and structured facts are obtained from Freebase. The tuples in Freebase are aligned with the tuples extracted from NYTimes. Simple string-matching heuristic is used to find the link between text and Freebase entities. A freebase tuple is aligned to the text if the s and o of freebase tuple aligns with the mentions m_s and m_o seen in the same sentence from text corpus. Relations with fewer than 10 tuples with mentions in text are filtered out. The original

[3]Two words that can be interchanged in a context are said to be synonymous relative to that context.

[4]A word that is more specific than a given word. For example, spoon is a hyponym of cutlery.

[5]A word that names a part of a larger whole. Example, ‘brim’ and ‘crown’ are meronyms of ‘hat’.

[6]`mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/downloads/`

[7]`dbpedia.org`

test set has only 80 tuples. Since such a test set is rather small, and in keeping with other KBC datasets, we create our own train-valid-test splits by randomly sampling about 2% tuples from $\mathcal{T}$. Only tuples with FB relations are used in the test set similar to previous experiments on this dataset. The details of the discussed dataset statistics can be seen in Table 3.1.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	Train	Valid	Test
FB15K	14,951	1,345	483,142	50,000	59,071
FB15K-237	14,541	237	272,115	17,535	20,466
WN18	40,943	18	141,442	5,000	5,000
WN18RR	40,943	11	86,835	3,034	3,134
YAGO3-10	1,23,182	37	1,079,040	5,000	5,000
NYT+FB	24,528	4,111	78,041	22,298	11,149

Table 3.1: Number of distinct entities, number of relation types, and size of train/test/valid folds of popular KBC benchmarks.

3.4 Tuple belief score and loss objectives

Atomic KBC techniques overwhelmingly fit the following template:

- Associate entities with suitable continuous representations (loosely called ‘embeddings’). These can be real or complex vectors or matrices that represent translations in space, projections, rotations, etc. Let us say $\boldsymbol{r}$ is the continuous representation of relation r.

- Associate relations with suitable continuous representations (loosely called ‘embeddings’). These can be real or complex vectors or matrices that represent points in space, vectors starting at the origin. Let us say $\boldsymbol{e}$ is the continuous representation of entity e.

- Define the *score* of a tuple (s, r, o) through a designed ‘belief score’ function $\phi(s, r, o; \theta)$ that returns a real value. Note that θ represents model parameters which in our case are typically embeddings only. The bulk of KBC research is in better and better design of ϕ. The scoring function ϕ may use additional parameters inside itself.

- Based on the ‘gold’ training data provided, train all entity and relation embeddings, along with any parameters inside ϕ, using a loss function $\mathcal{L}$. Negative examples (not seen in training data) are generated for model training to improve the model’s ability to distinguish between valid and invalid tuples.

3.4.1 Negative sampling

To ease notation, a labeled positive tuple $\big((s, r, o); l = 1\big)$ may be denoted simply as (s, r, o) and a perturbed, presumed negative tuple $\big((s, r, o); l = 0\big)$ denoted as (s', r', o'), although only one of s, r, o may be perturbed. Also $\mathcal{T}'_{tr}$ represents the set of all negative samples used while training the model.

In literature the following strategies to construct negative tuples are discussed. Each strategy constructs two negative examples for each positive example (s, r, o) — one by replacing s, and another by replacing o.

- Random sampling: We sample a negative set $Neg(s, r)$ for every tuple, computed as $\{(s', r', o')|o' \in \mathcal{E}, s' = s, r' = r\}$ i.e., negative tuples are formed by uniformly sampling entities. Similarly, the set $Neg(r, o)$ is sampled.

- Unseen fact sampling: We sample a negative set $Neg(s, r)$ for every tuple, computed as $\{(s, r, o')|o' \in \mathcal{E} \wedge (s', r', o') \notin \mathcal{T}, s' = s, r' = r\}$. Similarly, the set $Neg(r, o)$ is sampled.

- Max sampling: We sample a negative set $Neg(s, r)$ for every tuple, computed as $\underset{(s,r,o')\in X}{\arg\max}\ \phi(s, r, o')$ where $X = \{(s', r', o')|o' \in \mathcal{E} \wedge (s, r, o') \notin \mathcal{T}, s' = s, r' = r\}$. Similarly, the set $Neg(r, o)$ is sampled.

- Mix sampling: Mix of above, using random sampling 50% of times and max sampling 50% of times.

- Typed sampling: Negative samples are obtained from entity set within type range of relation [Krompaß et al., 2015]. Basically, the negative set $Neg(s, r)$ for every tuple is computed as $\{(s', r', o')|o' \in \mathcal{E} \wedge (s, r, o') \notin \mathcal{T} \wedge type(o') == type(r_{obj}), s' = s, r' = r\}$, where the function $type(e)$ gives the type of the input entity. $type(o')$ is the type of entity o' and $type(r_{obj})$ is the type of the entity r expects in its object position. Similarly, the set $Neg(r, o)$ is sampled.

Randomly sampling negative tuples is the most efficient and popularly used technique amongst others.

3.4.2 Loss Objective

The models are trained such that tuples observed in the KB have higher scores than unobserved ones. Several loss functions have been proposed; we discuss three common ones in this dissertation.

3.4.2.1 Additive margin loss

A simple way to train embeddings and ϕ is to encourage

$$\phi(s,r,o) \gg \phi(s',r',o') \tag{3.3}$$

for positive tuple (s,r,o) and negative tuple (s',r',o'). A hinge loss with a tuned hyperparameter margin $\gamma > 0$ is the most common way to write the corresponding loss function:

$$\mathcal{L} = \sum_{(s,r,o)} \sum_{(s',r',o')} \max\Big\{0, \phi(s',r',o') + \gamma - \phi(s,r,o)\Big\} \tag{3.4}$$

This is similar to pairwise ranking loss as in RankSVM [Joachims, 2002].

3.4.2.2 Logistic loss

The second type of commonly used loss function minimizes the **negative log-likelihood of logistic loss** or **sigmoid cross entropy loss** as follows:

$$\mathcal{L} = \sum_{(s,r,o)\in\mathcal{A}} \log\left[1 + e^{-y_{sro}\phi(s,r,o)}\right] \tag{3.5}$$

Here y_{sro} is 1 if the fact (s,r,o) is true (for $l = 1$) and -1 otherwise (for $l = 0$). Also, $\mathcal{A}$ is the set of all positive facts ($\mathcal{T}_{tr}$) along with the negative samples ($\mathcal{T}'_{tr}$).

3.4.2.3 Log-likelihood or cross entropy loss

Another approach is to use ϕ to define *probabilities* for filling in blanks in incomplete tuples:

$$\Pr(o|s,r) = \frac{\exp(\beta\phi(s,r,o))}{\sum_{o'}\exp(\beta\phi(s'=s,r'=r,o'))} \quad \text{and} \quad \Pr(s|o,r) = \frac{\exp(\beta\phi(s,r,o))}{\sum_{s'}\exp(\beta\phi(s',r'=r,o'=o))}, \tag{3.6}$$

where ϕ is the scoring function of the inference method, $\beta(>0)$ is the temperature parameter (generally β=1) and $s,o \in \mathcal{E}$ and relation $r \in \mathcal{R}$. If summing over *all* s' or o' is not feasible because $\mathcal{E}$ is very large, estimates are built from smaller volumes of negative samples. (If the sampling is biased, the left hand side may not be a formal probability, but it may suffice for training.)

The **log-likelihood loss** or **cross-entropy loss** is then defined as

$$\mathcal{L} = -\sum_{(s,r,o)\in\mathcal{T}_{tr}} \Big(\log\Pr(o|s,r;\theta) + \log\Pr(s|o,r;\theta)\Big) \tag{3.7}$$

The summation is over $\mathcal{T}_{tr}$ which is the set of all positive facts. Note that typically, KBC models do not include a third term on $\Pr(r|s,o;\theta)$ as the evaluation is only on the tasks $(s,r,?)$ and $(?,r,o)$.

3.5 Model Regularization

Regularization of models helps speed up training, improves performance and prevents overfitting. KBC models are also regularized during training. $L1$ regularization is used in translation models like TransE and RotatE. TransE also normalized the entity and relation embeddings to unit norm after every update. $L2$ regularization is also popularly used for KBC models Yang et al. [2015]. While Lacroix et al. [2018] proposed $L3$ regularization. ConvE used dropouts in hidden layers of the network Dettmers et al. [2018]. In this dissertation, we regularize entity embeddings and relation embeddings at the end of every epoch. The regularizer is applied only to the parameters that are involved in the computation of the instantaneous loss $\mathcal{L}$. We use $L2$ regularizations for most models unless specifically mentioned.

In this dissertation, we regularize entity embeddings and relation embeddings at the end of every epoch. The regularizer is applied only to the parameters that are involved in the computation of the instantaneous loss $\mathcal{L}$. We use $L2$ regularizations for most models unless specifically mentioned.

Chapter 4

Matrix Factorization and Tensor Factorization

In the context of macro-inference, two popular KGC methods are matrix factorization (MF) and tensor factorization (TF). In this chapter we conduct a detailed analysis of their comparative strengths and weaknesses. This results in a new joint MF-TF method that performs at least as well as the better of the two base models. During the development of our joint algorithm, we also find and fix a significant problem with KGC evaluation in the literature.

4.1 Introduction

As discussed in Section 1.1.1, KBC models can be further subdivided into two broad categories: matrix factorization and tensor factorization. In both cases the models learn one or more embeddings of the relation r, however, they differ in their treatment of entities s and o. TF approaches (e.g., E [Riedel et al., 2013], TransE [Bordes et al., 2013], DistMult [Yang et al., 2015], ComplEx [Trouillon et al., 2016], TypedComplEx [Jain et al., 2018a], Rescal [Nickel et al., 2011] models) learn separate embeddings for s and o, whereas MF methods (e.g., F [Riedel et al., 2013] and extensions [Verga et al., 2016]) learn an embedding per entity-pair (s, o).

MF was one of the first neural technique for Relation Extraction (RE). In this chapter we study MF performance on the task of KBC across various datasets. The main goal of this chapter is to study MF's effectiveness for the general task of KBC and answer three key questions.

1. Does MF perform well for KBC? Our extensive evaluation reveals that MF has an unusually varied performance across various KBC datasets, achieving MRR scores as high as 74% but also as low as zero.

2. What makes MF performance so sensitive? We find that MF's performance can be ex-

plained by dataset sparsity, in the form of the fraction of entity-pairs that are outside the training vocabulary (OOV).[1]

3. Can we improve MF performance to obtain respectable scores in spite of high OOV rates? The original MF model has a rather *ad hoc* way of handling OOVs — each OOV entity pair gets a random embedding. We propose three enhancements.

Our first model learns a single OOV vector. This ignores signals from the constituent entities, as it uses the same vector for all OOV entity pairs. Our second model generates entity pair vectors from the constituent entity embeddings, on the fly, using a generator layer that is trained to output "MF-like" embeddings. While the second model has better performance on OOV test examples, it degrades[2] on test facts where the gold entity pair is seen — rendering it far inferior to basic TF models like ComplEx. In response, we propose a hybrid TF-MF model (inspired by Singh et al. [2015]). This model is robust and obtains strong results across all datasets. Verga et al. [2016] also proposed methods to generate embedding of a new entity-pair on the fly for the F model. However, their work is different from ours since, at test time, they expect knowledge of several tuples between the same entity pair.

Additionally, we recognize that special care is needed to handle OOV entity-pairs when *evaluating* MF against TF. Otherwise an MF algorithm may erroneously appear to perform better than it really does, as in the case of F's (MF model) performance on FB15K-237 [Toutanova et al., 2015]. We describe the first unified KBC evaluation protocol that can meaningfully compare MF and TF approaches for KBC.

We contribute open-source implementations[3] of all models and testing protocols for further research.

4.2 Standard Evaluation Protocol

Common metrics used to compare algorithms are mean reciprocal rank (MRR) and the percentage of o^*s obtained in top 1 (HITS@1) and 10 (HITS@10) results (See section 3.2 for details).

[1]We refer to such entity-pairs as OOV entity-pairs. A more robust definition of OOV entity pairs (based on their train set frequency, say) could have been used. We used the definition of OOV based on zero- vs. non-zero count because it was able to explain most of the difference between model performances. Future work could study the impact of a more general definition of OOV entity-pairs.

[2]Here we refer to a model which, instead of learning separate entity-pair embeddings (initialized by pre-trained embeddings of MF model), generates them from constituent entity embeddings (initialized by pre-trained embeddings of DM model). In principle, a network with sufficient capacity, trained with sufficient data, should be able to reconstruct an entity-pair embedding from single-entity embeddings. In practice, in case of non-OOV entity-pairs, direct training of an entity pair embedding can be superior if the entity pair appears reasonably often.

[3]`https://github.com/dair-iitd/KBI`

The testing procedure is typically run with two modifications. First, we use filtered evaluation metric, already discussed in section 3.2.

The second modification applies primarily to MF models. In MF, an embedding is learned only for entity pairs that appear in $\mathcal{T}_{tr}$. Therefore, it is futile to score every (s^*, r^*, o) over a large range of os, for most of which, entity-pair embedding $\boldsymbol{ep}_{so}$ is not even known. Instead, only those os in a smaller set

$$o = \{o|\exists r : (s^*, r, o) \in \mathcal{T}_{tr} \cup \mathcal{T}_v \cup \mathcal{T}_{ts}\} \tag{4.1}$$

are considered as candidates for ranking [Toutanova et al., 2015, Verga et al., 2016]. If entity pair (s^*, o) is not trained then a random vector is assumed for $\boldsymbol{ep}_{s^*o}$ (making evaluation non-deterministic, hence a problem).

4.3 Comparison Under Standard and Sanitized KBC Evaluation Protocol

In this section, we first present a detailed study of previous approaches under standard evaluation protocols, which exposes the limitations of existing evaluation protocols. We then propose a sanitized protocol which evaluates KBC models more accurately. We further add a few baselines overlooked in prior work, and present results adjusted for KBC evaluation.

Dataset and Models: To better understand the strengths and weaknesses of each model, we compare various KBC models (see Table 2.1) on a variety of datasets - WN18, FB15K, FB15K-237, NYT+FB (refer to Section 3.3 for dataset details). Note that our literature search reveals that no model has been tested on all datasets, prior to this work. To the best of our knowledge, no work reports results of E & F models on WN18 or FB15K, TransE on FB15K-237 or NYT+FB, and DM & ComplEx on NYT+FB (prior to this work).

Implementation details: We implement all algorithms in a common framework written using Keras/Theano [Chollet, 2015]. We use embeddings in $\mathbb{R}^d$, where d=100, throughout, trained using Adagrad. Negative samples are extracted through random sampling technique discussed in section 3.4.1. Note that since MF models operate over entity pairs, they do not need two Neg sets. They use one set where new entity pairs are sampled only from the entity pairs found in $\mathcal{T}$, since embeddings for only those pairs get learned. 200 random negative samples are drawn per positive tuple. Margin γ is 1 for max margin methods. Entity and entity-pair vectors are re-normalized to unit norm after each batch update [Yang et al., 2015]. Batch size is 20,000. Models are trained up to 200 epochs, but with early stopping on a validation set to prevent from overfitting. We train each model on each dataset using log-likelihood (LL), max-margin (MM) and logistic (L) loss functions and pick the best loss function (according to dev performance) for every setting. In particular, we find that TransE performs much better with MM loss. LL loss works better or at par in all other models except that MM outperforms LL for DistMult on WN18 dataset. L works best for ComplEx.

We follow the train-dev-test splits used in previous experiments for FB15K, WN18, and FB15K-237. The test sets $\mathcal{T}_{ts}$ are generally 3–10% random samples from $\mathcal{T}$. For NYT+FB, previous works had experimented on a test fold with only 80 correct tuples [Riedel et al., 2013]. Since such a test set is rather small, and in keeping with our other data sets, we create our own train-test splits by randomly sampling about 2% tuples from $\mathcal{T}$. Only tuples with FB relations are used in the test set similar to previous experiments on this dataset.

	Model	FB15K			FB15K-237			WN18			NYT+FB		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
1	ComplEx	**66.97**	**55.21**	**85.60**	**37.46**	**27.97**	55.95	**93.84**	**93.32**	**94.54**	69.43	64.84	76.55
2	DistMult	60.82	46.51	84.78	37.21	27.43	**56.12**	80.42	68.5	94.2	62.48	56.40	72.17
3	E	22.86	16.4	35.04	30.87	23.63	45.38	2.74	1.48	5.38	8.83	3.67	19.74
4	TransE	43.11	24.99	71.97	3.57	0.4	1.48	37.15	4.22	84.96	13.57	8.79	39.63
5	F (Old eval)	*33.62*	*22.27*	*60.20*	*28.01*	*13.21*	*64.76*	*82.95*	*71.76*	*98.84*	*89.28*	*83.48*	*97.84*
6	F (KBC eval)	13.35	9.45	17.03	0.0	0.0	0.0	0.14	0.04	0.20	74.34	68.96	80.01
7	MFreq($o\|r^*$)	24.91	18.84	36.03	33.05	25.45	47.60	3.10	1.92	5.28	11.42	6.23	20.39
8	MFreq($o\|s^*$)	8.22	15.61	15.61	0.01	0.0	0.0	0.17	0.38	0.38	**79.34**	**94.93**	**94.93**

Table 4.1: The first five rows compare 5 models on 4 datasets using the standard evaluation protocol - MRR, HITS@k (H@k) on test queries $(s^*, r^*, ?)$. The 6th row shows F's performance using our proposed KBC evaluation protocol. The last 2 rows report results of 2 most-frequent sanity-check baselines.

Model performance with the Standard protocol: The first five rows of Table 4.1 report standard protocol performance of all the models across the datasets. E has good performance on FB15K-237, whereas TransE gets good scores on FB15K, however ComplEx emerges the most robust. For TF models on three datasets (FB15K, FB15K-237, WN18) our experiments are able to replicate (or improve upon) most reported results [Yang et al., 2015, Bordes et al., 2013, Toutanova et al., 2015].[4] Since NYT+FB uses a new test fold, and F hasn't been tested on other datasets, those results cannot be directly compared against previous work.

We find that F outperforms ComplEx on NYT-FB dataset by wide margins and does not perform as well as ComplEx on the rest. It appears that a qualitative analysis of ComplEx vs F will shed light on their relative strengths and weaknesses. Our analysis reveals a limitation in the standard evaluation protocol that can inflate F's performance scores for OOV entity pairs.

[4]Since we use 100 dimensional embeddings throughout, we obtain slightly lower scores than Trouillon et al. [2016] for ComplEx, which uses 200 dimensional embeddings.

The problem with the Standard protocol: Recall the second modification from standard protocol. When ranking possible entities o using the score $\phi(s^*, r^*, o)$ from MF models, the standard protocol considers a subset o, instead of all entities in $\mathcal{E}$. This is because many entity pair embeddings (s^*, o) are not even trained in the model, and hence their scores will be meaningless. We call these **OOV entity pairs**. O contains all entities for which the entity pair (s^*, o) is trained. Additionally, all such o^*s that are gold entities for some test query $(s^*, r^*, ?)$ are also added to O. If these are not trained, a random vector is assumed for them.

a.

(Bill Gates, lives in, ?)	F (old)	F (new)
(Bill Gates, lives in, Seattle)	5.34	5.34
(Bill Gates, lives in, Medina)	0.04	-1.4
(Bill Gates, lives in, New York)	?	-1.4
⋮ ⋮ ⋮	?	-1.4
Reciprocal rank	0.5	∼0.0

b.

(Tina Fey, lives in, ?)	F (old)	F (new)
(Tina Fey, lives in, New York)	2.30	2.30
(Tina Fey, lives in, Seattle)	1.1	1.1
(Tina Fey, lives in, Medina)	?	-2.12
⋮ ⋮ ⋮	?	-2.12
Reciprocal rank	1	1

Table 4.2: Original F with old evaluation protocol vs. F (trained OOV vector) with KBC evaluation protocol. Gold tuple in bold, and italics means that entity-pair was not seen during training. (a) Bill Gates is seen with one o in training — not the gold answer; (b) Tina Fey is seen with two os including the gold answer.

Table 4.2(a) illustrates an extreme case where the gold entity pair (Bill Gates, Medina) is not seen in training, and only one o (Seattle) is seen with s^*. Here, MRR for F model will be computed as 0.5 — a gross overestimation. Implicitly, (s^*, o^*) is getting ranked higher than all other OOV (s^*, o)s, whereas they should all be equal. In other words, the mere presence of $\mathcal{T}_{ts}$ in Equation (4.1) leaks information.

Proposed Sanitized protocol: A correct KBC evaluation protocol must assume all OOV entities at the same rank, and output the average value over all possible rankings for them. In our sanitized protocol, we assume one random OOV entity pair (s^*, e_{oov}), identify *all* $o \in \mathcal{E}$ that are OOV, assign them all the same score from the model and compute aggregate scores based on all possible rankings of OOV candidates. In Table 4.2(a), the MRR should be computed as the average of $\frac{1}{2}$, $\frac{1}{3}$,. . . , which is very small.

We note that most existing MF models have used test folds in which none of the gold entity pairs are OOV (except FB15k-237). Hence, the results reported in most previous papers are not affected by our proposed fix. Also, if variants of MF models are being compared among themselves, while they may overestimate performance somewhat, the relative ordering of various models may not be affected. On the other hand, OOVs become a central issue when MF models are compared against or combined with TF models, since realistic levels of sparsity are very different in the two models.

Model performance with Sanitized protocol: When the sanitized protocol is used (Table 4.1 line 6), F's performance on all datasets drops drastically, to the extent that its performance is practically zero on two datasets, and extremely weak on the third. Also, its performance is worse that a simple baseline of MFreq$(o|r^*)$ (discussed in next paragraph) in all datasets. However, it continues to have the best numbers (among all trained models) for NYT+FB.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	ep OOV (%)
FB15K	14,951	1,345	68.70
FB15K-237	14,541	237	100.00
WN18	40,943	18	99.52
NYT+FB	24,528	4,111	0.75

Table 4.3: No. of distinct entities, no. of relations and entity pair OOV rate, i.e., percentage of tuples in test set, whose entity pairs (ep) weren't seen while training.

Why such a significant drop? The answer lies in entity pair OOV rates, i.e., the percentage of tuples in the test fold whose entity pairs were not seen while training. Table 4.3 reports some statistics about the datasets as well as their test sets. We notice that FB15K, FB15K-237 and WN18 all have a very high OOV rate, which is strongly correlated with poor performance of F. Dataset NYT+FB has a tiny OOV rate and F performs well on it. Because *single entity* OOVs are infrequent compared to entity pair OOVs, we expect TF methods to shine in large pair OOV regimes. Singh et al. [2015] highlight that while matrix factorization performs well for RE, it is not robust to sparse data and does not capture latent entity types that can be crucial for accurate relation extraction. On the other hand, although tensor factorization models are able to compactly represent entity types using unary embeddings, they are unable to adequately represent the pair-specific information that is necessary for modeling relations. Our data-driven analysis adds to Singh et al. [2015] understanding. We believe that OOVs, and more generally, data sparsity, offer a more practical insight into differences between two model classes.

Most-frequent baselines: To improve our understanding of the difficulty of each dataset and the quality of each model beyond trivial choices, we introduce two baselines for our task. Given a query, $(s^*, r^*, ?)$ our first baseline ranks all entities based on the frequency of their occurrence with relation r^*, i.e., it orders each entity o based on the cardinality of the set $\{t|\exists s : t =$

$(s, r^*, o) \wedge t \in \mathcal{T}_{tr}\}$. A similar baseline orders each entity o based on its frequency of occurence with s^*, i.e., based on cardinality of the set $\{t | \exists r : t = (s^*, r, o) \wedge t \in \mathcal{T}_{tr}, \}$. We name these baselines MFreq$(o|r^*)$ and MFreq$(o|s^*)$ respectively.

The last two rows of Table 4.1 report the performance of these baselines. It is satisfying to see that for FB15K and WN18 datasets, ComplEx outperforms the baselines by large margins. However, for FB15K-237, ComplEx is only marginally better than MFreq$(o|r^*)$. A closer analysis reveals that this dataset is constructed so that there is minimal entity-pair overlap between relations. How would any model, then, predict the best o for a query $(s^*, r^*, ?)$? If entity pairs have not been repeated much, a natural approach may just find the most frequent entities seen with the relation and order based on frequency. We checked some high MRR predictions made by ComplEx and found that often questions, like, what is the language of a specific website, were answered correctly as English. This is likely not because ComplEx figured out the language of each website, but because English was the most frequent one in the dataset.

We also observe that E's performance remains broadly similar to the performance of the baseline MFreq$(o|r^*)$. We attribute this to E's scoring function, since given s^* and r^*, the only term relevant for ranking os is $\boldsymbol{w}_r^\top \cdot \boldsymbol{e}_o$, i.e., the model looks for compatibility with r^* and ignores s^* completely.

Finally, for NYT+FB, under the sanitized protocol, MFreq$(o|s^*)$ beats F model significantly suggesting that while F is the best model on that dataset, it is not good enough. We explore this further in the next section.

4.4 Toward a Robust MF Model

The previous section highlights the importance of OOV entity-pairs in the performance of MF models. We now present a series of models to gracefully handle entity pair OOVs within MF.

4.4.1 MF with a Trained OOV Vector (F$_{vecOOV}$)

A natural extension to F is to explicitly model an OOV entity-pair vector. In particular, we represent a vector $\boldsymbol{ep}_{oov}$ for F and $\boldsymbol{e}_{oov}$ for TF[5]. This modification means that all facts with an OOV entity-pair will have the same score.

OOV vectors can be trained in many ways. We develop two baselines that don't train the vectors explicitly. The first baseline (Table 4.4, line 1) assigns a *random* value to $\boldsymbol{ep}_{oov}$. The

[5]The choice of parameterizing a single OOV vector is empirically motivated. We experimented with backoff-based parameterization, which learn a distinct OOV vector corresponding to each entity, but we did not observe any improvement likely due to overfitting.

Model	FB15K		WN18		NYT+FB	
	MRR	HITS@10	MRR	HITS@10	MRR	HITS@10
F (random)	13.35	17.03	0.14	0.20	74.34	80.01
F (average)	18.27	24.62	0.13	0.16	71.65	76.80
F (trained)	**20.21**	**27.42**	0.27	0.38	**81.51**	**93.67**
F (generated)	13.51	22.67	**0.80**	**1.22**	0.20	0.10

Table 4.4: Results on F model after explicitly modeling OOV vectors. OOV training outperforms other baselines, especially for NYT+FB. Results on FB15k-237 not reported, due to 100% OOV rate.

second one called the *average* baseline computes $\boldsymbol{ep}_{oov}$ as the average of all (s, o) pairs that occur only once in training (Table 4.4, line 2).

We also propose a procedure to *train* $\boldsymbol{ep}_{oov}$ (Table 4.4, line 3). The broad motivation is to score a known tuple higher than a tuple with an OOV. To ensure this, we add ep_{oov} in the Neg set for each train tuple. This encourages the model to learn embeddings such that $\phi^F(r, ep_{so}) > \phi^F(r, ep_{oov})$. Thus, we ensure that the performance of F is maintained when a gold test entity pair is seen during training. Table 4.2(b) illustrates an example where the correct answer (New York) is seen with Tina Fey and OOV training doesn't displace its position.

Model	OOV		Non-OOV	
	MRR	HITS	MRR	HITS
F$_{vecOOV}$	0.01	0	57.33	75.98
F$_{genEP}$	11.19	21.58	18.59	25.04
DM	55.34	80.13	72.87	94.99
ComplEx	61.22	80.76	79.58	96.22
F+ComplEx (AS)	1.12	1.23	51.37	81.74
F+ComplEx (RAL)	**61.37**	**80.95**	**80.82**	**96.45**

Table 4.5: Performance segregated by OOV and non-OOV test queries on FB15k. F+C (RAL) performs best on both OOV and non-OOV.

To assess the effectiveness of the F$_{vecOOV}$ model, we breakdown its performance on subset of test queries that have OOVs and non-OOV gold entity pairs. This analysis is meaningful only for FB15k, since other datasets have extreme entity-pair OOV rate (see Table 4.3). Clearly, as Table 4.5 shows, while F$_{vecOOV}$ has extremely poor performance on OOVs (and thus weak performance overall), it performs decently on non-OOVs. We attribute this to the fact that the F$_{vecOOV}$ is designed with the inductive bias that facts with OOV embeddings are worse than facts with seen entity-pair embeddings. Another shortcoming of this model is that it ignores all information present in constituent entities of an OOV entity pair.

4.4.2 Generate OOV Entity Pair Vectors (F$_{genEP}$)

To fix the above shortcomings, we propose an enhancement to aid F in performing well on facts with OOV entity pairs. The main insight is to generate an informed OOV entity pair embedding by leveraging the information in the constituent entities (Table 4.4, line 4). We would like these generated entity pair embeddings to be similar to what MF would have produced had these entity pairs been observed. To this end, we train a *generator* layer that is applied on the concatenation of an entity pair's constituent entities' vectors to produce entity pair embeddings.

We obtain such a generator layer as a by product of training a model with loss function $\mathcal{L}(\boldsymbol{ep}_{so}, \boldsymbol{r}, \boldsymbol{e}_s, \boldsymbol{e}_o) = \mathcal{L}^F(\boldsymbol{ep}_{so}, \boldsymbol{r}) + \mathcal{L}^C(\boldsymbol{ep}_{so}, \boldsymbol{e}_s, \boldsymbol{e}_o))$. Here $\mathcal{L}^F$ is the loss of the vanilla F model and $\mathcal{L}^C = \|\boldsymbol{ep}_{so} - \boldsymbol{ep}^g_{so}\|_2$, is regression loss, which we use to guide the generator layer to generate F like embeddings. We define $\boldsymbol{ep}^g_{so} = f\left(P\left[\begin{smallmatrix} \boldsymbol{e}_s \\ \boldsymbol{e}_o \end{smallmatrix}\right]\right)$ ($\boldsymbol{e}_s, \boldsymbol{e}_o$ are column vectors), the generator layer $P \in \mathbb{R}^{2d \times d}$ and f can be any activation function. At test time, we concatenate the trained embeddings of constituent entities and use the learnt generator layer to obtain an embedding for an entity pair. Note that the new model is no longer a pure MF model, as it also uses entity embeddings. However, entity pair embeddings are only generated for OOV entity-pairs; the trained embeddings of seen entity pairs from the MF model are used for other entity pairs.

At train time, we initialize the model with pre-trained embeddings — MF entity pair embeddings, MF relation embeddings and DistMult entity embeddings. We do not use any activation function (f) as it led to a slight decrease in performance. Table 4.5 shows that F$_{genOOV}$ indeed has improved performance on OOV test facts. However, the performance on seen test facts drops significantly, leading to a weak overall performance. We attribute poor performance of this model on seen test facts to the joint optimization and potential change in the relation embeddings, especially when dealing with OOV entity pairs.

4.4.3 Using TF to Guide MF

We now extend MF (from Section 4.4.1) based on the following two observations: (1) TF models like ComplEx perform robustly on both OOV and non-OOV test facts (see Table 4.5). (2) Singh et al. [2015] show that MF and TF models have complimentary strengths. Could we use a TF model (such as ComplEx) to guide MF to perform better on OOV test facts?

Background on TF augmented MF models: Recall that Singh et al. [2015] find that the two models have complimentary strengths (see section 4.3). In response, they developed a TF augmented MF model which outperforms all other models on artificial datasets and NYT+FB. Their best model (F+E) uses the scoring function $\phi^{E+F} = \sigma(\phi^E + \phi^F)$, where σ is the sigmoid function. We call this model an **additive score (AS)** model, since the scores (ϕ) of two models are added. Early works of Riedel et al. [2013] also experiment with a similar model for

NYT+FB. Later, Toutanova et al. [2015] implement an AS model F+E+DM and tested it on FB15K-237.

We are motivated to develop an extension to MF that leverages TF to perform gracefully on both OOV and non-OOV test facts. For brevity, we refer to these MF extensions as joint MF-TF models, since they involve a TF model to guide the training of MF. Does an additive score model meet this requirement?

Dataset	Δ MRR	Δ HITS@10
FB15K-237	-4.09	-5.32
FB15K	-62.18	-75.67
NYT+FB	-69.03	-76.0
WN18	-4.62	-14.36

Table 4.6: Change in performance of ComplEx initialized with corresponding embeddings extracted from ComplEx+F (AS).

Additive loss (AL) joint model: Preliminary investigations (Table 4.5) reveal that additive score models can suffer substantial loss in performance for both OOV and non-OOV test facts. Table 4.6 shows drop in performance in the ComplEx component when trained jointly in additive score F+ComplEx model. It clearly shows that ComplEx's performance can reduce drastically due to joint training. A primary reason is that F scores overshadow ComplEx scores, since the scoring function in F involves a product of 2 small numbers and ComplEx involves a product of 3 small numbers.[6] Moreover, the number of parameters in MF models (vectors for entity pairs) significantly outnumber those in TF models (vectors for entities). This can lead to significant overfitting.

In response, we develop a different class of joint models in which instead of adding the scores (ϕs), we add their loss functions: $\mathcal{L}^{MF+TF} = \mathcal{L}^{MF} + \mathcal{L}^{TF}$. We name these *additive loss* joint models (AL). We expect this to be more resilient to overshadowing, since the joint loss expects each model's individual loss to decrease as much as possible. One may note that AL style of training is equivalent to training the models separately. However, joint training makes other extensions possible, such as regularization.

Regularized additive loss (RAL): We extend the vanilla AL joint model to a *regularized* joint model in which the parameters of MF model are L2-regularized. We expect this regularization to encourage a reduction in overfitting caused by the larger number of MF parameters

[6]To calibrate them, we tried standardizing scores obtained from pre-trained models. We also tried to learn a linear function to push ComplEx and F model scores to the same range simultaneously. We also tried sharing of relation parameters to allow information to flow from ComplEx to MF. Unfortunately, none of the approaches were robust across datasets.

(as compared to TF model parameters)[7]. Overall, our final joint model has the loss function:

$$\mathcal{L}^{MF+TF}(\theta^{MF}, \theta^{TF}) = \mathcal{L}^{MF}(\theta^{MF}) + \mathcal{L}^{TF}(\theta^{TF}) + \lambda \left\| \theta^{MF} \right\|_2$$

At test time, for a query $(s^*, r^*, ?)$ an AL model cannot simply add the scores, since some entity-pairs may be OOVs. We develop various backoff cases, reminiscent of traditional backoff in language models [Manning and Schütze, 2001]. For every o:

- **Case 1:** $(s^*, o) \in \mathcal{T}_{tr}$. Score of tuple is $\phi^{TF}(s^*, r^*, o) + \phi^{MF}(s^*, r^*, o)$.
- **Case 2:** $(s^*, o) \notin \mathcal{T}_{tr}$, but o is seen in training.
  Score of tuple is $\phi^{TF}(s^*, r^*, o) + \phi^{MF}(e_{oov}, r^*, e_{oov})$.
- **Case 3:** o is not seen in training. Score of tuple is $\phi^{TF}(s^*, r^*, e_{oov}) + \phi^{MF}(e_{oov}, r^*, e_{oov})$.

Note that the additive loss models train an $\mathbf{e}_{oov}$ by adding it to the Neg set for each train tuple, as also done in F_{vecOOV} (trained) model in section 4.4.1.

Results: Table 4.5 shows that RAL F+ComplEx performs well on OOV and non-OOV test queries. Regularization penalty λ is tuned over a small devset from within the training set.

	Model	FB15K			FB15K-237			WN18			NYT+FB		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
1	F	20.21	16.26	27.42	0.01	0.0	0.0	0.27	0.2	0.38	81.51	74.74	95.67
2	DM	60.82	46.51	84.78	37.21	27.43	56.12	80.42	68.58	94.20	62.48	56.40	72.17
2	ComplEx	66.97	55.21	85.60	37.46	27.97	55.95	93.84	93.32	**94.54**	69.43	64.84	76.55
3	F+E (AS)	26.24	20.59	37.35	29.71	22.15	44.39	1.60	0.07	4.04	82.46	75.99	92.21
4	F+ComplEx (AS)	16.84	11.48	26.42	32.90	24.18	50.25	90.02	88.94	91.54	79.41	72.78	89.70
5	F+E+DM (AS)	29.89	23.42	42.00	33.65	24.04	49.26	22.92	14.54	39.26	81.41	74.37	91.41
6	F+ComplEx (AL)	59.61	49.39	78.77	11.21	4.16	25.96	79.90	68.7	93.82	25.92	20.94	35.56
7	F+ComplEx (RAL)	**67.46**	**56.00**	**85.80**	**37.93**	**28.03**	**57.46**	**93.99**	**93.64**	94.48	**84.21**	**77.25**	**95.63**
8	F+ComplEx (Oracle)	69.02	58.80	84.99	37.46	27.97	55.95	98.71	98.39	99.15	89.79	81.97	96.69

Table 4.7: Performance (MRR, HITS@k/H@k) of joint models. AL = additive loss. AS = additive score. F+ComplEx combined with regularized additive loss (RAL) is the highest scorer as well as most robust across all datasets.

Robustness: RAL F+ComplEx performs well for OOV & Non-OOV test cases, hence is robust to all testing conditions. To validate the hypothesis we evaluate the model on a variety of datasets — WN18, FB15K, FB15K-237, NYT+FB. Table 4.7 compares its performance with individual models as well as other joint models.

We find that different additive score models (rows 3–5) perform well on some datasets, but are not robust across them. For example, in FB15K none of these are able to match up to ComplEx's performance. We attribute this to overfitting by F, which makes the model believe that ϕ^F is predicting the tuple very well. This lets F override TF and reduces the joint model's

[7]Hence regularizing MF parameters is more critical than TF parameters.

Dataset	Singleton Rate	Doubleton Rate
FB15K	83.83%	12.19%
FB15K-237	90.52%	8.64%
WN18	99.80%	0.20%
NYT+FB	8.06%	59.04%

Table 4.8: The fraction of entity-pairs occurring exactly once and exactly twice. NYT+FB has an unusual distribution.

need to learn the best TF model(s). Note that row 3 and row 5 are the models reported in Singh et al. [2015]) and Toutanova et al. [2015], respectively.

Rows 6 and 7 report the results of additive loss F+ComplEx models, both without and with regularization. As anticipated, adding the losses improves performance since both models get trained well. Moreover, regularization also helps considerably since now the model is not overwhelmed by too many F parameters. The RAL version of F+ComplEx achieves the best scores in all datasets. The reported numbers were state-of-the-art when we did this research in 2017. Note that a basic MF (F model) added to TF (ComplEx model) isn't robust — only an "OOV trained" MF, when integrated with TF, attains good performance.

Row 8 of Table 7 also shows the accuracy of an oracle model that, for every test query, post-facto selects the model with the more accurate score (between ComplEx and F). This upper bounds the performance expected from a perfect joint ComplEx+F model, fixing the constituents. We find that the oracle is only 4-5 MRR percentage points better than our best model for two datasets, and the differences are much less for the other two. Overall, it suggests that our proposed joint model obtains a strong robust performance.

4.5 Discussion and Future Work

We now list two observations that suggest important directions for future research in KB inference.

Dataset Characteristics: Our work subjects datasets to natural sanity checks. First, we introduce two most frequent baselines (Table 4.1) to understand the nature of the KBs. Second, we compute entity-pair OOV rates (Table 4.3) as a rough predictor of the relative success of the TF and MF families. Finally, in Table 4.8, we report the singleton and doubleton percentages (for entity pairs). A singleton is an entity-pair occurring only once in the data ($\mathcal{T}_{tr} \cup \mathcal{T}_v \cup \mathcal{T}_{ts}$) and a doubleton is an entity pair that occurs exactly twice. Doubletons have a strong effect in the scenario painted in Table 4.2.

We find that most datasets have an idiosyncrasy, which raises the question whether they are good representatives for naturally occurring KBs. In particular, WN18 and FB15K-237 have near 100% entity-pair OOV rates, unlikely to be the case in real KBs. In FB15K-237 the best models are not much better than MFreq($o|r^*$) baseline. This is because the dataset is artificially constructed to avoid relations with entity-pair overlap. But, this reduces its ability to make many interesting inferences. For NYT+FB, MFreq($o|s^*$) performance has a strong performance with 95% score on HITS@10. Moreover, learned models are able to improve its MRR by only about four percentage points. Statistics in Table 4.8 reveal that this could be because the dataset has an unusually high number of entity-pair doubletons: it is the only data set where doubletons by far outnumber singletons. It is unlikely that such a distribution occurs in a naturally occurring dataset. FB15K appears to pass our sanity tests. We believe that focus on better datasets will likely help us in better progress on KB inference.

Note that many improved datasets like WN18RR came after this work. We leave explorations on those datasets as future work.

Experiments on Synthetic Data: KBs comprise of various inference patterns - synonymy e.g. (Narendra Modi, was born in, India) implies (Narendra Modi, took birth in, India); transitivity e.g. (Michael Jordan, teaches at, Berkeley) and (Berkeley, is located in, California) implies (Michael Jordan, teaches in, California). To assess the ability of MF and TF models to capture various inference patterns, we build a synthetic dataset. Following Singh et al. [2015], we construct a synthetic dataset containing 4 different kinds of relations: Black, Red, Green and Purple (see Figure 4.1).

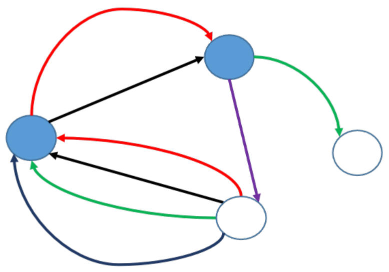


Figure 4.1: Synthetic dataset where, entities are represented as circles, color of the circle is its type. Relations are color coded links between entities.

- A *Black* relation exists between 2 entities with a probability 0.5
- A *Red* relation exists between 2 entities if a *black* relation exists between them (*red* and *black* relations are synonymous).
- A *Green* relation exists between 2 entities if the two have different latent types. A binary type is randomly assigned to each of the entity variables (half the entities are randomly sampled and assigned type zero).

- A *Purple* relation exists between 2 entities if there is a two-hop path between them. For example $(s_1, purple, o_2)$ holds iff both (s_1, red, o_3) and (o_3, red, o_2) holds true. This relation can be used to check the effectiveness of TF and MF models at performing multi-hop inference.

We add a new *purple* relation not discussed by Singh et al. [2015]. Also, we ensure a Zipf distribution in both entity and entity pair frequencies while Singh et al. [2015] do not. This is done to make the synthetic dataset as real as possible. Interestingly the FB15k-237 dataset has no *red* relation since all relations with a high entity overlap are filtered.

*MF/TF model performance on Synthetic Data:*In Table 4.9 we report the performance of TF model (ComplEx - CX) and MF model (F) on synthetic dataset. F model's ability to capture synonymy is demonstrated by its superior performance on *red* relations while CX's ability to capture latent types is demonstrated by its superior performance on *green* relations. The two models capture complementary aspects of KB inference and hence our final model combining the two: F+ComplEx (RAL) does well on both. We find none of the models show good performance on *purple* relation highlighting their inability to capture multi-hop inference. A study similar to ours comparing the models that train over relation paths [Guu et al., 2015, García-Durán et al., 2015, Toutanova et al., 2016] will benefit our understanding of multi-hop inference.

	Green		Purple		Red	
Model	MRR	HITS@10	MRR	HITS@10	MRR	HITS@10
ComplEx	**100**	**100**	6.21	15.43	10.26	23.31
F	55.17	60.32	2.04	5.21	**100**	**100**
F+ComplEx (RAL)	**100**	**100**	6.30	15.50	**100**	**100**

Table 4.9: Performance of TF model (CX) and MF model (F) and F+ComplEx (RAL) across various relations in Synthetic dataset.

Recent Progress on Evaluation Protocol: Any evaluation measure should know how to handle situations of score ties. Very recently, Sun et al. [2019b] highlighted that inability of an evaluation protocol to handle score ties is the key reason behind the unusual behavior of some recent NN-based methods. They further discuss three strategies to handle ties - rank of the correct entity is minimum amidst other same score entities (over-estimate model performance), rank of the correct entity is maximum amidst other same score entities (under-estimate model performance), or randomly assign one of many possible rankings of same scored candidate entities. Before their work, our sanitized evaluation protocol also highlighted that the same score candidates should be handled appropriately. In our context, all OOV entities have the same score.

These entities must be given the same score, and the average value over all possible rankings should be used for each of them (see Table 4.2).

4.6 Conclusion

We present the first study on the effectiveness of MF for KBC. After replacing the standard evaluation protocol with our sanitized proposal, we find that MF's performance is highly varied — it obtains MRR scores from 0 to 74% on different datasets. We also propose two simple frequency baselines and are surprised to find that MF's performance is worse than the better baseline in *all* domains! Further analysis reveals that MF performs poorly at high-OOV rates. We develop a series of extensions aimed at mitigating the effect of OOVs in MF. MF's performance improves by training OOV embeddings. Our most successful model uses ComplEx to augment our improved version of MF via a regularized additive loss. This hybrid model is highly robust and has the best performance on all datasets. Note that a basic MF added to TF isn't robust — only an "OOV trained" MF, when integrated with TF, attains good performance.

Relevant Publication

- [Jain et al., 2018b]: "Mitigating the Effect of Out-of-Vocabulary Entity Pairs in Matrix Factorization for KB Inference". Prachi Jain, Shikhar Murty, Mausam, Soumen Chakrabarti. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJ-CAI). Stockholm, Sweden. July 2018.

Chapter 5

Enhanced Knowledge Base Inference using unsupervised Type Induction

In previous chapter, we have reviewed a number of KG inference techniques. Even the best among them make mistakes. Inspection of these mistakes suggests that they fail to build a satisfactory internal representation of entity types. In this chapter, we set about to rectify this limitation. We propose a method for enhanced KBC, which learns the representation of the types of entities. We experimentally demonstrate that these models predict entities of compatible types more frequently than corresponding type-agnostic models.

5.1 Introduction

In the previous chapter (section 4.5), we found NYT+FB has an unusually high number of entity-pair doubletons, making it a bit too unreal. In the rest of the datasets, TF models outperformed MF models. Hence from now onwards we focus on TF models. Our preliminary analysis of popular TF models — DistMult (**DM**) and ComplEx (**CX**), reveals that they make frequent errors by ranking high incorrect entities that are not even compatible with types expected as arguments of r^*. In **19.5%** of predictions made by DM on FB15K, the top prediction has a type different from what is expected (see Table 5.1 for illustrative examples).

In response, we propose a modification to TF models (DM, ComplEx) by explicitly modeling type compatibility. Our modified function $\phi^M_{type}(s, r, o)$ is the product of three terms: a function of the original tuple score $\phi^M_{base}(s, r, o)$, subject type-compatibility between r and s, and object type-compatibility between r and o. Our type-sensitive models, **TypeDM** and **TypeComplEx**, do not expect any additional type-specific supervision — they induce all embeddings using only the original KB. An ideal way to train type embeddings would be to provide canonical type signatures for each relation and entity [Minervini et al., 2016]. Unfortunately, these aspects of realistic KBs are themselves incomplete [Neelakantan and Chang, 2015, Murty et al., 2018]. Our models train all embeddings using only the training tiples in the KG without any designated types and do not rely on any explicit type supervision.

Experiments over three datasets show that all typed models outperform base models by significant margins, obtaining then state-of-the-art results in several cases. We perform additional analyses to assess if the learned embeddings indeed capture the type information well. We find that embeddings from typed models can predict symbolic types (known to us but concealed from the system) better than base models.

We note that an older model called E [Riedel et al., 2013] can be seen as modeling type compatibilities. Observe that, in the DistMult score $\langle \boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o \rangle$ for fact triple (s, r, o), $\boldsymbol{r}$ mediates a direct compatibility between s and o for relation r, whereas, in E model, $\boldsymbol{v}_r^T \cdot \boldsymbol{e}_s + \boldsymbol{w}_r^T \cdot \boldsymbol{e}_o$, we are separately scoring how well s can serve as subject and o as object of the relation r. Thus, in the second case, $\boldsymbol{e}_e$ may be expected to encode the type(s) of entity e, where, by ‘type’, we loosely mean “information that helps decide if e can participate in a relation r, as subject or object”. Model E may be regarded as a relation prediction model that depends purely on type compatibility checking. Heuristic filtering of the entities that do not match the desired type at test time has been known to improve accuracy [Toutanova et al., 2015, Krompaß et al., 2015]. Our typed models formalize this within the framework of embeddings and allow for discovery of latent types without additional data. Krompaß et al. [2015] also use heuristic typing of entities for generating negative samples while training the model (see Section 3.4.2). Our experiment finds that this approach is not very competitive against our typed models.

Previous work has also explored additive combinations of DM and E [Garcia-Duran et al., 2015, Toutanova et al., 2015]. We directly compare against these models and find that, our proposal outperforms both E, DM and their linear combinations.

We contribute open-source implementations[1] of all models and experiments discussed in this paper for further research.

Subject s^*	**Relation** r^*	**Gold Object** o^*	**Prediction 1**	**Prediction 2**
Howard Leslie Shore	follows-religion	Jewism (religion)	Walk Hard (film)	21 Jump Street (film)
Spyglass Entertainment	headquarter-located-in	El lay (location)	The Real World (tv)	Contraband (film)
Les Fradkin	born-in-location	New York (location)	Federico Fellini (person)	Louie De palma (person)
Eugene Alden Hackman	studied	Rural Journalism (education)	Loudon Snowden Wainwright III (person)	The Bourne Legacy (film)
Chief Phillips (film)	released-in-region	Yankee land (location)	Akira Isida (person)	Presidential Medal of Freedom (award)

Table 5.1: Samples of top two DM predictions (having inconsistent types) on FB15K. TypeDM predicts entities of the correct type in top positions in the corresponding examples.

[1] https://github.com/dair-iitd/KBI

5.2 TypeDM and TypeComplEx

Representation: We start with M as the base model; where M can be DM or ComplEx or any TF model. The first key modification (see Figure 5.1) is that each entity e is now represented by *two* vectors: $\boldsymbol{u}_e \in \mathbb{R}^K$ to encode *type* information, and $\boldsymbol{e}_e \in \mathbb{R}^{D'}$ to encode residual information. Typically, $K \ll D'$. The second, concomitant modification is that each relation r is now associated with *three* vectors: $\boldsymbol{r} \in \mathbb{R}^{D'}$ as before, and also $\boldsymbol{v}_r, \boldsymbol{w}_r \in \mathbb{R}^K$. The vectors $\boldsymbol{v}_r$ and $\boldsymbol{w}_r$ encode the expected types for subject and object entities.

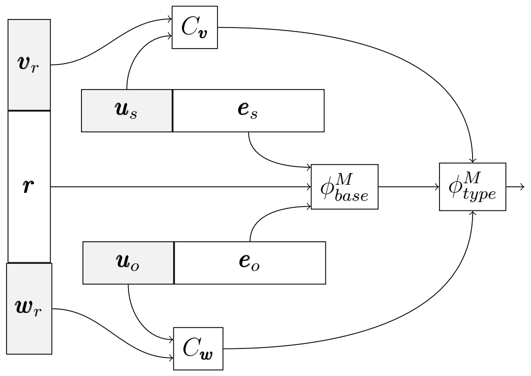


Figure 5.1: TypeDM and TypeComplEx.

Prediction: A nonlinearity function (sigmoid) is applied to the base model's prediction score:

$$\phi^M_{base}(s, r, o) = \sigma(\phi^M(s, r, o)), \tag{5.1}$$

and then combine with two additional terms that measure type compatibility between the subject and the relation, and the object and the relation:

$$\phi^M_{type}(s, r, o) = \phi^M_{base}(s, r, o)\, C_{\boldsymbol{v}}(s, r)\, C_{\boldsymbol{w}}(o, r), \tag{5.2}$$

where $C_{\boldsymbol{x}}(e, r)$ is a function that measures the compatibility between the type embedding of $\boldsymbol{e}$ for a given argument slot of $\boldsymbol{r}$:

$$C_{\boldsymbol{x}}(e, r) = \sigma(\boldsymbol{x}_r \cdot \boldsymbol{u}_e) \tag{5.3}$$

If each of the three terms in Equation 5.2 is interpreted as a probability, $\phi^M_{type}(s, r, o)$ corresponds to a simple logical AND of the three conditions.

We want $\phi^M_{type}(s, r, o)$ to be almost 1 for positive instances (tuples known to be in the KG) and close to 0 for negative instances (tuples not in the KG). For a negative instance, one or more of the three terms may be near zero. There is no guidance to the learner on which term to drive down.

5.3 Reflexivity aware model

A binary relation r on a set of entities $\mathcal{E}$ is reflexive if it relates every element of $\mathcal{E}$ to itself (i.e. $\langle e, r, e \rangle$).

Error analysis of TypeComplEx (best model – see Table 5.3) revealed that the model is unable to distinguish between reflexive and non-reflexive relations. It predicts gold subject entity (s^*) for query $(s^*, r^*, ?)$ even for non-reflexive relation (r^*) like `parentOf` and `hasColor`. Restricting all the top predictions of the test queries $(s^*, r^*, ?)$ not to be same as query subject entity s^*, such a hard constraint will disrupt model performance on reflexive relations like `educational_institution_campus` and `location_us_county_place`. We propose a reflexivity aware KBC model, whose scoring function is discussed below:

$$\phi^{ref}(s, r, o) = \epsilon_r * \left([\![s = o]\!] * \phi^{type}(s, r, o) \right) + \left([\![s \neq o]\!] * \phi^{type}(s, r, o) \right) \qquad (5.4)$$

The parameter ϵ_r captures the reflexivity property of each relation r. If relation r is fully reflexive, $\epsilon_r = 1$ and if relation r is never reflexive, $\epsilon_r = 0$. Relations with ϵ_r values between 0 and 1 have some reflexive and some non-reflexive facts. Value of $\phi^{ref} = 0$ only when the query relation is not reflexive and the query subject entity is same as predicted object entity. The proposed reflexivity aware model improved the performance of TypeComplEx by 1-2 point MRR on various datasets (see Table 5.3 for performance details of the TypeComplEx (reflexive) model). TypeDM (reflexive) also show improved performance on most datasets.

Similarly, [Minervini et al., 2017a] incorporated equivalence (relation between the relation pairs: partOf – componentOf) and inversion (relation between the relation pairs: partOf - hasPart) axioms in KBC models by using an auxiliary loss term. Note that these axioms are different from relation reflexivity, where a reflexive relation maps the same entity to itself ($\langle e, r, e \rangle$: $\langle ColumbiaUniversity, /education/educational\_institution\_campus/educational\_institution,$ $ColumbiaUniversity \rangle$).

5.4 Experiments

Model Size: DM uses $(E + R)D$ model weights for a KB with R relations and E entities, whereas TypeDM uses $E(D' + K) + R(D' + 2K)$. ComplEx case is analogous. To make comparisons fair, we set D' and K so that the total number of model weights (real or complex) are about the same for base and typed models.

Hyperparameters: We run AdaGrad for up to 1000 epochs for both logistic loss and log likelihood loss (described in Section 3.4), with a learning rate of 0.5 and with early stopping on the dev fold to prevent overfitting. All the models generally converge after 300-400 epochs, except TypeDM that exhausts 1000 epochs. E, DM, DM+E and ComplEx use 200 dimensional vectors. All except E perform best with logistic loss and 20 negative samples (obtained by randomly corrupting s and r) per positive fact. This is determined by doing a hyperparameter search on the values $\{10, 20, 50, 100, 200, 400\}$. With logistic loss, model weights θ are L2-regularized and gradient norm is clipped at 1.

Model	Embedding dimensions	Number of parameters
E	200	3,528,200
DM+E	100+100	3,393,700
DM	200	3,259,200
TypeDM	180+19	3,268,459
ComplEx	200	6,518,400
TypeComplEx	180+19	6,201,739

Table 5.2: Sizes were approximately balanced between base and typed models (FB15K).

For typed models we first perform hyperparameter search for size of type embeddings (K) such that total entity embedding size remains 200. We get the best results at $K = 20$, from among values in $\{10, 20, 30, 50, 80, 100, 120\}$. This hyperparameter search is done for the TypeDM model (which is faster to train than TypeComplEx) on FB15k dataset, and the selected split is used for all the typed models. To balance total model sizes (Table 5.2), we choose $K = 19$ dimensions for $\boldsymbol{u}_e, \boldsymbol{v}_r, \boldsymbol{w}_r$ and 180 dimensions for $\boldsymbol{e}, \boldsymbol{r}$. Notice that a typed model has a slightly higher number of parameters for relation embeddings, because it needs to maintain *two* type embeddings of size K, over and above $\boldsymbol{r}$. Using $K = 19$ reduced and brought the total number of parameters closer to that of the base model, for a fair direct comparison. The model performance did not differ by much when using either of the options (i.e., $K = 19$ or 20)

Typed models and E perform best with 400 negative samples per positive tuple while using log-likelihood loss (robust to a larger number of negative facts as opposed to logistic loss, which falls for class imbalance). FB15K and YAGO3-10 use L2 regularization coefficient of 2.0, and it is 5.0 for FB15K-237. Note that the L2 regularization penalty is applied to only those entities and relations that are a part of that batch update, as proposed by Trouillon et al. [2016]. β is set to 20.0 for the typed models, and 1.0 for other models if they use the log-likelihood loss. Note that, $\phi^M_{type} \in [0, 1]$ for typed models, we scale the log-likelihood-loss term with a hyperparameter $\beta > 0$ (a form of inverse temperature) to allow $\Pr(o|s, r)$ to take values over the full range $[0, 1]$ in loss minimization. Entity embeddings are unit normalized at the end of every epoch, for the type models. Also, we find that in TypeDM scaling the embeddings of the base model to unit norm performs better than using L2 regularization.

Results: Table 5.3[2] shows that TypeDM and TypeComplEx dominate across all datasets. E by

[2]Note that the models (written in keras/theano) discussed in Table 4.1 are 100 dimensional and trained using

Model		FB15K			FB15K237			YAGO3-10		
		MRR	HITS@1	HITS@10	MRR	HITS@1	HITS@10	MRR	HITS@1	HITS@10
E	Mean	20.51	14.45	33.24	16.38	10.32	29.22	4.45	3.6	5.66
	?,r,e2	17.7	11.75	30.2	6.39	2.44	14.26	0.22	0.04	0.36
	e1,r,?	23.33	17.15	36.29	26.47	18.2	44.18	8.67	7.16	10.96
DM+E	Mean	48.76	35.62	73.62	25.21	18.4	39.56	47.87	37.44	67.51
	?,r,e2	46.47	71.07	33.35	15.41	9.78	26.89	37.41	27.3	58.1
	e1,r,?	51.05	76.17	37.89	35.02	27.02	52.23	58.33	47.58	76.92
DM	Mean	68.09	57.66	84.82	24.74	15.42	45.37	41.87	31.72	62.85
	?,r,e2	66.79	56.35	83.24	18.52	10.54	36.46	24.56	13.74	50.02
	e1,r,?	69.39	58.97	86.39	30.96	20.3	54.28	59.19	49.7	75.68
TypeDM	Mean	72.43	62.96	86.63	30.11	22.12	47.38	44.3	35.35	61.9
	?,r,e2	70.12	60.71	84.23	20.14	13.13	35.65	29.85	19.56	51.76
	e1,r,?	74.73	65.21	89.03	40.08	31.11	59.1	58.76	51.14	72.04
TypeDM (reflexive)	Mean	75.73	69.51	86.87	29.61	21.49	46.76	47.04	38.26	63.73
	?,r,e2	73.32	67.11	84.32	20.11	12.74	36.02	33.13	22.36	54.94
	e1,r,?	78.15	71.92	89.42	39.11	30.25	57.51	60.95	54.16	72.52
ComplEx	Mean	70.18	60.36	86.02	24.25	15.04	44.47	44.61	34.26	64.74
	?,r,e2	68.74	59.1	84.14	18	10.1	35.05	31.79	21.14	54.46
	e1,r,?	71.63	61.63	87.9	30.51	19.97	53.88	57.43	47.38	75.02
TypeComplEx	Mean	75.72	68.1	**86.57**	30.41	22.16	47.74	47.55	39.52	63.12
	?,r,e2	73.21	65.65	**83.88**	20.39	12.99	36.35	34.56	24.84	54.62
	e1,r,?	78.22	70.54	**89.25**	40.43	31.33	59.13	60.54	**54.2**	71.62
TypeComplEx (reflexive)	Mean	**76.89**	**71.65**	86.3	**31.09**	**22.7**	**49.13**	**49.28**	**40.16**	**66.15**
	?,r,e2	**74.5**	**69.34**	83.85	**20.85**	**13.22**	**37.55**	**37.22**	**26.88**	**57.28**
	e1,r,?	**79.28**	**73.95**	88.74	**41.33**	**32.19**	**60.72**	**61.35**	53.44	**75.02**

Table 5.3: KBC performance for base, typed, and related formulations. Typed models outperform their base models across all datasets. Reflexive version of typed model further improves the performance on most datasets. Note that the mean performance reported in the table is comparable to the model scores reported in later chapters.

itself is understandably weak, and DM+E does not lift it much. Each typed model improves upon the corresponding base model on all measures, underscoring the value of type compatibility scores. For direct comparisons with published work, we choose 200 and 400 parameters per entity for DM and ComplEx respectively (ComplEx model has two 200 dimensional embeddings per entity). DM and TypeDM, on increasing the dimensionality to 400, yield MRR scores of 69.79 and 78.91, respectively, for FB15K.

The proposed reflexivity aware model further improve the performance of TypeComplEx by 1-2 point MRR on various datasets (see Table 5.3 for performance details of the TypeComplEx (reflexive) model). TypeDM (reflexive) also show improved performance on most datasets.

Note that the performance of the query $(s, r, ?)$ shows larger gains as compared to the performance of $(?, r, o)$.

In 2017 when this research was done, the results of the typed models were competitive with various reported results for models of similar sizes that do not use any additional information, e.g., soft rules [Guo et al., 2018], or textual corpora [Toutanova et al., 2015]. Note that TypeDM and TypeComplEx are also competitive on the WN18 dataset [Bordes et al., 2013], but we omit those results, as WN18 has 18 very generic relations (e.g., hyponym, hypernym, antonym, meronym), which do not give enough evidence for inducing types.

We also compare against the heuristic generation of type-sensitive negative samples [Krompaß et al., 2015]. For this experiment, we train a ComplEx model using this heuristically generated negative set, and use standard evaluation, as in all other models. On FB15k the model performed poorly with 10 MRR. We find that all the models reported in Table 5.3 outperform this approach.

5.5 Analysis of Typed Embeddings

We perform two further analyses to assess whether the embeddings produced by typed models indeed capture type information better. For these experiments, we try to correlate (and predict) known symbolic types of an entity using the unsupervised embeddings produced by the models. We take a fine catalog of most frequent 90 freebase types over the 14,951 entities in the FB15k dataset [Xie et al., 2016]. We exclude `/common/topic` as it occurs with most entities. On an average each entity has 12 associated types.

1. **Clustering Entity/Type Embeddings:** For this experiment we subselect entities in FB15k that belong to one of the 5 types (people, location, organization, film, and sports) from the freebase dataset. These cover 84.88% of FB15K entities. We plot the FB15K entities e using the PCA projection of $\boldsymbol{u}_e$ and $\boldsymbol{e}_e$ in Figure 5.2, color-coding their types. We observe

200 negative samples for 200 epochs only. Hence the difference in performance.

that $\boldsymbol{u}_e$ separates the type clusters better than $\boldsymbol{e}_e$, suggesting that $\boldsymbol{u}_e$ vectors indeed collect type information. We also perform k-means clustering (k=5) of $\boldsymbol{u}_e$ and $\boldsymbol{e}_e$ embeddings of these entities, as available from different models. We report cluster homogeneity (a clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class) and completeness (a clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster. This score is complementary to the previous one. Its purpose is to provide a piece of information about the assignment of samples belonging to the same class. More precisely, a good clustering algorithm should assign all samples with the same true label to the same cluster) scores [Rosenberg and Hirschberg, 2007] in Table 5.4. Typed models yield superior clusters.

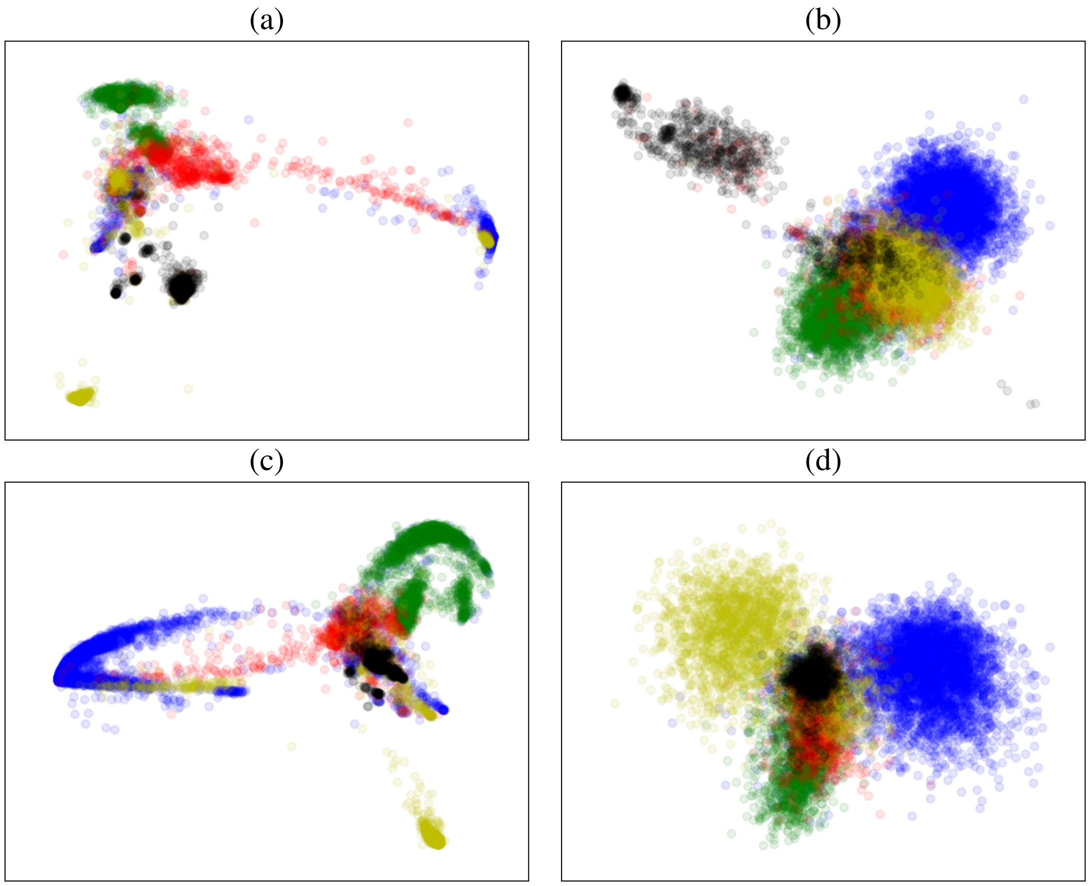


Figure 5.2: Projection of vectors representing entities belonging to frequent KB types- {**people**, **location**, **organisation**, **film**, **sports**}: a: TypeDM,$\boldsymbol{u}_e$; b: TypeDM,$\boldsymbol{e}_e$; c: TypeComplEx,$\boldsymbol{u}_e$; d: DM,$\boldsymbol{e}_e$.

2. **Prediction of Symbolic Types:** FB15k has 14,951 entities labeled with 232 types. We train a single-layer network that inputs pre-trained embeddings from various models and predicts a set of symbolic types (232 types) from the KB. The model layer's output is

passed through a softmax layer to make the final predictions. We use cross-entropy loss to train the network (Sx232) for 20 epochs; S is the size of input embedding. Ten percent of the FB15k entities are selected randomly for testing, and the rest are used for training. We train separate networks for different models. Note that S can be 19, 180, or 200 depending on the type of model from where the input embeddings are obtained. This experiment tells us the extent to which the embeddings capture KB type information (that was not provided explicitly during training). Table 5.4 reports average macro F1 score (5-fold cross validation). Embeddings from TypeDM and TypeComplEx are generally better predictors than embeddings learned by ComplEx, DM and E. $\boldsymbol{u}_e \in \mathbb{R}^{19}$ is often better than $\boldsymbol{e}_e \in \mathbb{R}^{180}$ or more, for typed models. DM+E with 199 model weights narrowly beats TypeDM with 19 weights, but recall that it has poorer KBC scores.

Method	Embed -ding	Size	H	C	Type F1
TypeDM	$\boldsymbol{u}_e$	19	66.72	66.29	81.77
TypeDM	$\boldsymbol{e}_e$	180	57.89	59.67	75.96
TypeDM	Both	199	**66.75**	**66.29**	82.57
DM	$\boldsymbol{e}_e$	200	51.40	48.12	81.34
TypeComplEx	$\boldsymbol{u}_e$	19	65.90	62.97	82.70
TypeComplEx	$\boldsymbol{e}_e$	180x2	50.76	48.57	74.75
TypeComplEx	Both	379	66.03	63.09	84.14
ComplEx	$\boldsymbol{e}_e$	200x2	51.56	47.20	81.58
DM+E	$\boldsymbol{u}_e$	19	0.48	2.05	74.66
DM+E	$\boldsymbol{e}_e$	180	49.62	47.24	82.72
DM+E	Both	199	49.66	47.26	82.68
E	$\boldsymbol{e}_e$	200	39.83	37.62	74.23

Table 5.4: Interpretation of embeddings w.r.t. supervised types: cluster homogeneity H, completeness C, and type prediction F1 score.

Note that the objective of experiments in this section is to check the quality of the typed embeddings learned i.e. if the model indeed learned the types better. Hence we use the pure type version of the base model and avoided polluting the experiment results from other models like the reflexive variants.

5.6 Conclusion and Future Work

We propose an unsupervised typing gadget, which enhances popular models for KBC (DistMult and ComplEx) with two type-compatibility functions, one between r and s and another between r and o. Without explicit supervision from any type catalog, our typed variants (with a similar number of parameters as the base models) substantially outperform base models, obtaining up to 7% MRR improvements and over 10% improvements in the correctness of the top result. To confirm that our models capture type information better, we correlate the embeddings learned without type supervision with existing type catalogs. We find that our embeddings indeed separate and predict types better. In future work, combining type-sensitive embeddings with a focus on less frequent relations [Xie et al., 2017], more frequent entities [Dettmers et al., 2018], or side information such as inference rules [Guo et al., 2018, Jain and Mausam, 2016] or textual corpora [Toutanova et al., 2015] may further increase KBC accuracy. It may also be of interest to integrate the typing approach here with the combinations of tensor and matrix factorization models for KBC [Jain et al., 2018b]. Also note that the current entity/relation representation, being only a point in space, is not equipped to handle type hierarchy. However, there has been recent work that uses box-embedding representation [Subramanian and Chakrabarti, 2018, Vilnis et al., 2018, Li et al., 2019], which can theoretically handle such a type hierarchy. But these models are harder to train.

Relevant Publication

- [Jain et al., 2018a]: "Type-Sensitive Knowledge Base Inference Without Explicit Type Supervision". Prachi Jain, Pankaj Kumar, Mausam, Soumen Chakrabarti. In Proceedings of the 2018 Annual Meeting of the Association for Computational Linguistics (ACL). Melbourne, Australia. July 2018.

Chapter 6

Knowledge Base Completion: Engineering Issues For Training

KBC datasets have a large number of (positive) training instances and an even larger number of negative training instances via negative sampling. Various KBC methods define diverse loss functions. These have different computational costs and memory footprints. Given the resource constraints of a computational environment (e.g. GPU size and practical training time available), these resource requirements dictate the extent of negative sampling that is feasible. Sometimes, apparently minor changes in the design of the loss function can change system accuracy to a surprising extent, and also open up paths to computational optimizations. To our knowledge, these trade-offs have not been studied adequately. One consequence is that baselines have remained unnecessarily weak [Kadlec et al., 2017, Ruffinelli et al., 2020]. During the development of various models described in this dissertation, we found that careful attention to these details can considerably improve old baselines, sometimes surpassing models and methods proposed since. Specifically, if large numbers of negative triples can be accommodated, the basic ComplEx model is extremely competitive. Our subsequent work uses this competitive baseline as a starting point.

6.1 Review of KBC score and loss functions

6.1.1 TransE

We recall that TransE [Bordes et al., 2013] embeds each entity e (variously, subject s or object o) to vectors $\boldsymbol{e}_s, \boldsymbol{e}_o \in \mathbb{R}^D$ and relations r to vectors $\boldsymbol{r} \in \mathbb{R}^D$ as well. The score function is defined as

$$\phi_{\text{TransE}}(s, r, o) = -\|\boldsymbol{e}_s + \boldsymbol{r} - \boldsymbol{e}_o\|_1. \tag{6.1}$$

As we shall see, the use of L_1 norm above is significant; if it is replaced by the L_2 norm, there are significant implications for memory footprint. To design the loss function, we observe that if $(s,r,o) \in \text{KB}$, we want $\phi_{\text{TransE}}(s,r,o)$ to be large; otherwise, if $(s,r,o) \notin \text{KB}$, we want it to be small. These considerations are combined into the loss function using a margin $\Delta > 0$ for the negative triples:

$$\mathcal{L}_{\text{TransE}} = -\sum_{(s,r,o)\in\text{KB}} \phi_{\text{TransE}}(s,r,o) + \sum_{(s',r',o')\notin\text{KB}} \big[\Delta + \phi_{\text{TransE}}(s',r',o')\big]_+, \tag{6.2}$$

where $[a] = \max\{0,a\}$ is the ReLU or hinge function. For each $(s,r,o) \in \text{KB}$, the number of perturbations (s',r',o') that are (assumed to be) not in the KB is astronomically large, possibly approaching E^2R, where the KB has E distinct entities and r distinct relation types. Usually, negative sampling is limited to perturbing only s or only o, resulting in $O(E)$ negative triples per positive triple, at most. But even this is considered computationally too burdensome, and a random sample is drawn.

6.1.2 RotatE

Computationally, RotatE [Sun et al., 2019a] is similar to TransE. It places $\boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o \in \mathbb{C}^D$ in the complex space, enforces unit complex modulus $|\boldsymbol{r}^d| = 1$ for each element of $\boldsymbol{r}$, and defines

$$\phi_{\text{RotatE}} = -\sum_{d\in[D]} |\boldsymbol{e}_s^d \boldsymbol{r}^d - \boldsymbol{e}_o^d|. \tag{6.3}$$

Here $|c|$ is the modulus of complex number $c \in \mathbb{C}$ and $\boldsymbol{e}_s^d \boldsymbol{r}^d$ is the product of two complex numbers. Note that the sum is similar to an L_1 distance. RotatE can learn symmetry vs. antisymmetry, inversion and composition, and generally performs better than TransE. Because of the difference to be computed for each dimension d, we will regard TransE and RotatE as members of the *additive* family of KBC methods.

6.1.3 DistMult and ComplEx

In contrast to additive KBC methods, DistMult and ComplEx [Yang et al., 2015, Trouillon et al., 2016, Lacroix et al., 2018, Jain et al., 2018a, Balažević et al., 2019, Kazemi and Poole, 2018] can be regarded as *multiplicative* methods, for reasons clarified below. For DistMult, $\boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o \in \mathbb{R}^D$ and

$$\phi_{\text{DM}}(s,r,o) = \sum_{d\in[D]} \boldsymbol{e}_s^d \boldsymbol{r}^d \boldsymbol{e}_o^d. \tag{6.4}$$

For ComplEx, $\boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o \in \mathbb{C}^D$ and

$$\phi_{\text{CX}}(s,r,o) = \Re\left[\sum_{d\in[D]} \boldsymbol{e}_s^d \boldsymbol{r}^d \boldsymbol{e}_o^{d\star}\right], \tag{6.5}$$

where $c^\star$ is the conjugate of complex number c and $\mathbb{R}(c)$ is its real part. In both cases, observe that inside the sum over dimensions d there is no addition or subtraction operation between entity and relation embeddings. This has important implications (see Section 6.2).

For either DistMult or ComplEx, the loss is commonly defined as

$$\mathcal{L} = - \sum_{(s,r,o)\in\mathcal{T}_{tr}} \Big(\log \Pr(o|s,r;\theta) + \log \Pr(s|o,r;\theta) \Big) \tag{6.6}$$

where

$$\Pr(o|s,r) = \frac{\exp(\phi(s,r,o))}{\sum_{o'} \exp(\phi(s,r,o'))} \quad \text{and} \quad \Pr(s|o,r) = \frac{\exp(\phi(s,r,o))}{\sum_{s'} \exp(\phi(s',r,o))}, \tag{6.7}$$

Here, again, observe the potential performance bottleneck of the sums in the denominator ranging over E entities. Indeed, early implementations approximated the full sum in the denominator with a partial sum over a random subset of terms, suitably scaled, plus the numerator itself (to maintain consistency). As we shall see, this sampling approximation may not be necessary; the specific inner-product form Equation (6.5) lets us evaluate the full denominator sum efficiently.

6.2 Additive vs. multiplicative loss with all negative triples

Here we first describe how Equation (6.7) can be fully evaluated without sampling approximation, supported by highly efficient tensorized computation libraries. Then we describe why this appears more difficult for additive formulations (TransE and RotatE). Finally, we describe how tweaking the distance norm from L_1 to L_2 might open up TransE to the same efficiency. In experiments, however, we see that L_1 norm generally gives more accurate KBC models.

6.2.1 Inner product

Dettmers et al. [2018] suggested taking one (s,r) pair and scoring it against all E entities o in a batch method they called “1-N scoring”, instead of computing the score of one fact (s,r,o) at a time, that they called “1-1 scoring”. For any *multiplicative* method in general, the score for all entities can be computed in parallel via a simple matrix multiplication, which is both memory and time-efficient, thanks to the optimized implementations provided in BLAS libraries.

To get into more detail, let us ask how the computation in Equation (6.7) can be efficiently vectorized in case of DistMult. ϕ_{DM} is often written in the form $(\boldsymbol{e}_s \odot \boldsymbol{r}) \cdot \boldsymbol{e}_o$, where $\odot$ is elementwise product, and $\cdot$ is an inner product. Here $(\boldsymbol{e}_s \odot \boldsymbol{r}) \in \mathbb{R}^D$, as is $\boldsymbol{e}_o \in \mathbb{R}^D$. If we want to evaluate $\phi_{\text{DM}}(s,r,o)$ over all $o \in \mathcal{E}$, we can write it as a matrix-vector product between a

$E \times D$ entity embedding matrix $\mathbf{E}$ and a $D \times 1$ row vector $\boldsymbol{e}_s \odot \boldsymbol{r}$ followed by a sum aggregation:

$$\sum_{o \in [E]} \exp\left(\mathbf{E}_{E\times D}\,(\boldsymbol{e}_s \odot \boldsymbol{r})_{D\times 1}\right)_{E\times 1}. \tag{6.8}$$

(In reality we would use log-sum-exp for numerical stability, but the computation structure will remain the same.) The total intermediate space needed to compute the above expression is $O(DE)$. If we want to further batch up subject entities s in batches of size B, the space required is $O(BD + DE)$.

6.2.2 L_1 distance

Let us now shift focus to ϕ_{TransE}. As with $\boldsymbol{e}_s \odot \boldsymbol{r}$ in DistMult, pre-computation of $\boldsymbol{e}_s + \boldsymbol{r}$ creates no trouble, giving a D-dimensional vector. If we have a batch of B (s, r) pairs, this gives us a $B \times D$ matrix; call this $\mathbf{B}$. The other matrix of interest is $\mathbf{E} \in \mathbb{R}^{E\times D}$ as before. Effectively, we have a set of B points in D dimensions, another set of E points in D dimensions, and we wish to compile a $B \times E$ matrix $\mathbf{A}$ of pairwise L_1 distances.

In non-vectorized code, this is easily possible to compute within $O(BD+DE)$ input space, $O(BE)$ output space, and no (or $O(1)$) working space, using the following approach.

```
1: for $b \in [B]$ do
2:     for $e \in [E]$ do
3:         $\mathbf{A}[b,e] \leftarrow 0$
4:         for $d \in [D]$ do
5:             $\mathbf{A}[b,e] \leftarrow \mathbf{A}[b,e] + \underbrace{\left|\mathbf{B}[b,d] - \mathbf{E}[e,d]\right|}$
```

Structurally, this is identical to the non-vectorized code for matrix multiplication.

```
1: for $b \in [B]$ do
2:     for $e \in [E]$ do
3:         $\mathbf{A}[b,e] \leftarrow 0$
4:         for $d \in [D]$ do
5:             $\mathbf{A}[b,e] \leftarrow \mathbf{A}[b,e] + \underbrace{\mathbf{B}[b,d]\;\mathbf{E}[e,d]}$
```

SciPy defines a general library function `scipy.spatial.distance.cdist`[1], which allows the compilation of $B \times E$ distances using any L_p norm. L_1 corresponds to input parameter `metric='cityblock'`, but the default is L_2, or `metric='euclidean'`.

To our astonishment, despite the similarity in the above pseudocodes, the memory com-

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cdist.html

plexity of a vectorized version of L_1 distance is $O(BDE)$, instead of $O(BD + DE + BE) = O(BD + DE)$ achievable for matrix multiplication. It appears the structure of computation changes to the following.

```
1: allocate a $B \times E \times D$ tensor $\mathbf{X}$
2: for $d \in [D]$ do
3:     $\mathbf{X}[:,:,d] \leftarrow \mathbf{B}[:,d] \otimes \mathbf{E}[:,d] \in \mathbb{R}^{B \times E \times 1}$    ▷ Each layer $d$ gets a $B \times E$ outer product.
4: $\mathbf{A} \leftarrow \text{sum-reduce}_d(\mathbf{X}) \in \mathbb{R}^{B \times E}$    ▷ Aggregate across layers $d$.
```

Needless to say, this wastes a lot of space, making it difficult to deal with all negative subject or object entities, even for small batches.

6.2.3 The special case of L_2 distance

SciPy's default choice of L_2 distances in their `cdist` routine affords a space-efficient solution. Suppose we have a matrix $\boldsymbol{B} \in \mathbb{R}^{B \times D}$ and a matrix $\boldsymbol{Y} \in \mathbb{R}^{E \times D}$. If we want a $B \times E$ matrix $\boldsymbol{A}$ with dot products, i.e., $\boldsymbol{A}[b,e] = B[b,:] \cdot Y[e,:]$, we can efficiently compute $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{Y}^\top$. What if we want $\boldsymbol{A}[b,e] = \|\boldsymbol{B}[b,:] - \boldsymbol{E}[e,:]\|_2^2$? We can write this as

$$\begin{aligned}\boldsymbol{A}[b,e] = \|\boldsymbol{B}[b,:] - \boldsymbol{E}[e,:]\|_2^2 &= (\boldsymbol{B}[b,:] - \boldsymbol{E}[e,:]) \cdot (\boldsymbol{B}[b,:] - \boldsymbol{E}[e,:]) \\ &= \|\boldsymbol{B}[b,:]\|_2^2 + \|\boldsymbol{E}[e,:]\|_2^2 - 2\boldsymbol{B}[b,:] \cdot \boldsymbol{E}[e,:]\end{aligned}$$

We can compute the first two terms without any asymptotic increase in storage, and the third term is the same as in case of dot product. If we need

$$\boldsymbol{A}[b,e] = \|\boldsymbol{B}[b,:] - \boldsymbol{E}[e,:]\|_2 \qquad \text{(i.e., } L_2\text{, not } L_2\text{-squared)}$$

we can write this as

$$\boldsymbol{A}[b,e] = \sqrt{\|\boldsymbol{B}[b,:]\|_2^2 + \|\boldsymbol{E}[e,:]\|_2^2 - 2\boldsymbol{B}[b,:] \cdot \boldsymbol{E}[e,:]}.$$

I.e., there is a final elementwise square-root on the $B \times E$ matrix. The gradient changes, but not the basic space and time performance structure.

This trick can be used if we change the formulation of TransE and RotatE to use L_2 distances instead of L_1 distances:

$$\phi_{\text{TransE}}(s,r,o) = -\|\boldsymbol{e}_s + \boldsymbol{r} - \boldsymbol{e}_o\|_2 \quad \text{and} \tag{6.9}$$

$$\phi_{\text{RotatE}}(s,r,o) = -\|\boldsymbol{e}_s \odot \boldsymbol{r} - \boldsymbol{e}_o\|_2\,. \tag{6.10}$$

Without experimental evaluation, we do not know the impact of the change of norm on the predictive accuracy of these models. We end this chapter with such an evaluation, which shows that switching from L_1 to L_2 is unfortunately detrimental to predictive accuracy. However, ComplEx with a much larger negative sample set gives a very competitive baseline.

	FB15k			WN18			YAGO3-10		
Method	**MRR**	**HITS@1**	**HITS@10**	**MRR**	**HITS@1**	**HITS@10**	**MRR**	**HITS@1**	**HITS@10**
SimplE [2018]	0.73	0.65	0.86	0.95	0.94	0.95	—	—	—
SIMPLE-V2	0.85	0.82	**0.91**	**0.95**	**0.96**	0.95	0.56	0.49	0.69
ComplEx [2016]	0.81	0.75	**0.91**	0.94	0.93	0.95	0.51	0.40	0.63
COMPLEX-V2	**0.86**	**0.83**	**0.91**	**0.95**	0.95	**0.96**	**0.58**	**0.50**	**0.71**
ComplEx-N3 [2018]	**0.86**	**0.83**	**0.91**	**0.95**	0.94	**0.96**	**0.58**	**0.50**	**0.71**

(a)

	FB15k-237			WN18RR		
Method	**MRR**	**HITS@1**	**HITS@10**	**MRR**	**HITS@1**	**HITS@10**
SimplE [2018]	0.23	0.15	0.40	0.42	0.40	0.46
SIMPLE-V2	0.34	0.25	0.53	0.46	0.43	0.52
ComplEx [2016]	0.31	0.22	0.51	0.42	0.40	0.47
COMPLEX-V2	0.35	0.26	0.54	0.47	**0.46**	0.53
ComplEx-N3 [2018]	**0.37**	**0.27**	**0.56**	**0.49**	0.44	**0.58**

(b)

Table 6.1: Table (a) and (b) reports performance of popular KBC models along with their counterparts trained such that all entities contrast a positive fact while computing the loss: COMPLEX-V2 and SIMPLE-V2. Models are evaluated on five commonly used benchmark datasets for KBC. ComplEx-N3 is same as ComplEx except it adds new facts with inverse predicates in train set, use L3 regularization and 1-N training. All model parameters are of same range. COMPLEX-V2 shows near SOTA performance on all datasets. V2 models are indistinguishable or slightly worse.

6.3 Experimental study

Implementation details: We reuse the original implementations and the best hyper-parameters released for RotatE and TransE [Sun et al., 2019a]. We re-implement CX [Trouillon et al., 2016], CX-N3 [Lacroix et al., 2018], SimplE [Kazemi and Poole, 2018] in PyTorch [2]. AdaGrad is used for fitting model weights. Model is trained for up to 1000 epochs, with early stopping on the validation set to prevent overfitting. In our experiments, we calibrate all models to have a similar number of parameters across all datasets — CX, CX-N3, COMPLEX-V2, SIMPLE-V2 use 2000-dimension vectors, except on Yago3-10, we train 1000-dimensional models. All models except CX-N3 use L2 regularization, CX-N3 uses L3 regularization. The range of the hyperparameters for the grid search is as follows: regularization coefficient from {1, 0.1, 0.01, 0.001, 0.0001, 0.00001}, learning rate from {0.5, 0.1, 0.01, 0.001, 0.0001}, batch size from {100, 200, 500, 1000, 2000}.

Link prediction performance: This subsection demonstrates how KBC model performance improves when trained using all possible entities as negative samples.

Multiplicative models: Table 6.1 shows that when trained with all entities as the negative samples, multiplicative models (COMPLEX-V2, SIMPLE-V2) significantly improve over the same model trained with a small set of negative samples. COMPLEX-V2 trained with all entities as the negative sample shows near SOTA performance, making it a strong baseline.

[2] https://github.com/dair-iitd/kbc-baseline

	FB15k	WN18	YAGO3-10	FB15k-237	WN18RR
RotatE [2019a]	0.61	0.94	0.37	0.29	0.45
RotatE-V2	**0.64**	**0.95**	**0.40**	**0.32**	0.45

Table 6.2: Performance (MRR) improvement for RotatE (100-dim) by scoring against all entities while training (instead of negative sampling)

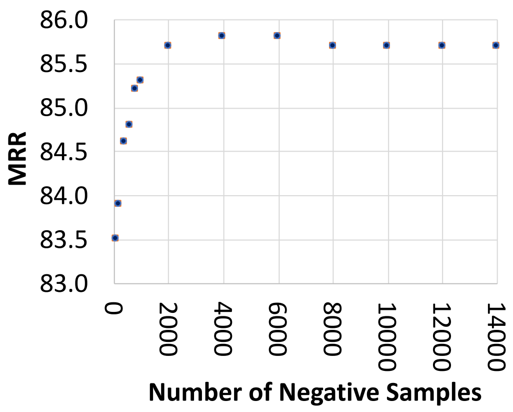


Figure 6.1: Influence on test MRR of number of negative samples used per positive training example. The peak is very mild and could be due to statistical variation.

Translation models: As pointed out in subsection 6.2, 1-N scoring is difficult to scale for translation models such as RotatE. To demonstrate the benefit of training all models such that all entities contrast a positive fact while computing the loss, we train RotatE for a reduced dimension (100). To overcome the memory challenges of training it on a single 12GB GPU, we train it by accumulating gradients over multiple batches, at the cost of increased training time. The results are reported in Table 6.2. Here, RotatE refers to the model trained with 256 negative samples, whereas RotatE-V2 refers to the model trained with all entities as the negative samples. We find that RotatE-V2 shows a significant improvement (up to 3 pt MRR) for FB15k, FB15k-237, and YAGO3-10, whereas for WN18 and WN18RR the model gives a slightly improved or similar performance.

Influence of Negative Samples: This experiment investigates the effect on model performance with the increasing number of negative samples per positive example. We report the performance of ComplEx model on FB15k dataset. We vary the number of negative samples in {100, 200, 400, 600, 800, 1k, 2k, 4k, 6k, 8k, 10k, 12k, 14k}. The 1-N scoring enables the computation of the score of all possible entities for query $(s, r, ?)$ and $(?, r, o)$ efficiently. For this experiment, we randomly subsample a smaller set of entities (negative examples) for each batch and compute approximate softmax scores. Figure 6.1 shows that the performance of ComplEx sharply improves with the increasing number of negative samples (in the beginning) and stabilizes around 2000 negative samples, with a slight dip at around 6000 negative samples (due to statistical variation).

We may also get benefit from generating negative samples of varying hardness but we leave that exploration to future work.

6.4 Discussion

The lessons we have learnt from our observations above may be summarized as:

- As long as memory footprint is manageable, all KBC models should use large (perhaps even exhaustive) negative samples and vectorized evaluation of contrastive loss — this most often leads to superior predictive accuracy.
- For models where vectorized evaluation of contrastive loss cannot be easily implemented for large negative samples (such as RotatE), gradient accumulation over multiple batches can be used at the cost of increased training time. This may still lead to better accuracy.
- Switching L_1 to L_2 norms is a tempting possibility to enable fast vectorized evaluation of contrastive loss while keeping memory footprint minimal. Unfortunately, training TransE and RotatE using L_2 norm resulted in visible drop in accuracy. The MRR score of TransE dropped by 7.8 points and RotatE by 6.5 points on FB15k. This takes away the possibility of optimizing these models for vectorized contrastive loss evaluation with low memory footprint.
- On the positive side, COMPLEX-V2 — plain old ComplEx with very large negative samples — is efficiently trainable and turns out to be an extremely competitive baseline. In fact, it largely wipes out the benefit from several models proposed since.

We are not alone in pointing out the last item above. Kadlec et al. [2017] and Ruffinelli et al. [2020] undertook an extensive exercise in tuning hyperparameters such as embedding dimensions, learning rate, batch size, regularization penalty, etc., and came to the same conclusion: the apparent accuracy gains from a new model architecture must be carefully assessed against relatively minor-looking but still significant modifications to hyperparameters in well-established models. Our experiments with negative sample size adds another weapon in the arsenal of old baselines that age well.

Relevant Publication

- [Jain et al., 2020a]: "Knowledge Base Completion: Baseline strikes back (Again)". Prachi Jain, Sushant Rathi, Mausam, Soumen Chakrabarti. arxiv 2020.

Chapter 7

Temporal Knowledge Base Completion

In chapter 5, we investigated the benefits of modeling types in KBs. In this chapter, we explore another important aspect of structured knowledge: time. We propose a new way to score temporal facts and capture statistical regularities between times associated with KB facts, resulting in more accurate KG inference.

7.1 Introduction

Many relations in KBs are transient or impermanent. Temporal KBs annotate each fact (or event) with the time instant or period in which it holds (or occurs). A person is born in a city in an instant, a politician can be a country's president for several years, and a marriage may last between years and decades. Temporal KBs may represent these by (s, r, o, t) tuples, where t is a time instant. The time instant t of the facts can be further generalized to discrete finite time sets T. Temporal KBC (TKBC) performs completion of temporal KBs. It is also primarily evaluated by link prediction queries $(s, r, ?, T)$ and $(?, r, o, T)$. Time prediction $(s, r, o, ?)$ has also been considered for predicting time instants, but not time sets or time intervals [Lacroix et al., 2020].

While KBC has been intensely researched, TKBC is only beginning to be explored. In this work we focus on a time-sets of size 1, i.e. time sets with only one time-interval. TKBC presents novel challenges in task definition and modeling. For instance, little is known about how best to predict time-intervals for $(s, r, o, ?)$ queries, or how to evaluate a system response interval. Moreover, we show that even for link prediction queries, evaluation faces subtle complications owing to the inclusion of time sets T in $(s, r, ?, T)$ and $(?, r, o, T)$ queries and requires careful rethinking of evaluation protocols. In this chapter, we propose improved evaluation protocols for both link and time prediction tasks in a TKBC.

TKBC also brings unique modeling opportunities. A TKBC system can learn typical durations of relation validity or distributions over time gaps between events from training data. E.g., a person must be born before becoming president, which must precede death. A nation rarely has two presidents at the same time. Such constraints can better inform both link and time predictions.

We present TIMEPLEX, a novel TKBC model, which beats all competitive baseline model performance on benchmark datasets for both link and time prediction . At a high level, TIMEPLEX performs tensor factorization of a temporal KB, using complex-valued embeddings for relations, entities and time points. It enables these embeddings to capture implicit temporal relationships across facts and relations, by providing temporal differences as explicit features. Our contributions are summarized below.

- We propose evaluation protocols for link and time interval prediction in TKBC. For link prediction, we highlight that existing evaluations seriously over/under-estimate system performance, and offer a time-aware filtering method for more reliable evaluation. For time interval prediction, we propose an evaluation metric that rewards a model for predicting an interval with partial overlap with gold interval, as well as for nearness to gold in case of no overlap.

- We present TIMEPLEX, a TKBC model that factorizes a temporal KB using entity, relation and time embeddings. It can learn and exploit soft ordering and span constraints between potentially all relation pairs (including that of a relation with itself). It beats competitive baseline models on several standard TKBC data sets.

We released an open-source implementation[1] of all models and experiments discussed here.

7.2 Preliminaries and Prior Work

7.2.1 Time-Agnostic KBC

TIMEPLEX builds upon ComplEx [Trouillon et al., 2016], which we abbreviate to CX here. As discussed earlier in Section 2.3.1, it embeds s, r, o to vectors of complex space $\boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o \in \mathbb{C}^D$. CX defines the score $\phi(s, r, o)$ of a fact (s, r, o) as $\mathbb{R}(\langle \boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o^\star \rangle)$ where

$$\phi^{CX}(s, r, o) = \mathbb{R}(\langle \boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o^\star \rangle) = \mathbb{R}(\textstyle\sum_{d=1}^{D} s[d]\ r[d]\ o^\star[d]) \qquad (7.1)$$

is a 3-way inner product, $\boldsymbol{e}_o^\star$ is the complex conjugate of $\boldsymbol{e}_o$, and $\mathbb{R}(c)$ is real part of $c \in \mathbb{C}$. We choose CX as our base model, because its performance is comparable to the best KBC models [Ruffinelli et al., 2020], at the time of writing this dissertation.

[1] github.com/dair-iitd/tkbi

7.2.2 Temporal KBC Problem Setup

A temporal KB associates the validity of a triple (s, r, o) with one or more time intervals $T \subseteq \mathfrak{T}$, where $\mathfrak{T}$ is the domain of "all time". Each interval T is represented as $[t_b, t_e]$, with begin and end time instants. Some event-style facts (e.g., born in) may have $t_b = t_e$. For simplicity, we assume that $\mathfrak{T}$ is discretized to a suitable granularity and is represented by a set of integers. Temporal KB facts have the form (s, r, o, T), and are partitioned into train, validation and test folds, abbreviated as tr, v, ts. System predictions are abbreviated as pr.

Given the train and validation folds, our goal is to learn a model that scores any unseen fact. A system is evaluated via link prediction queries $(?, r, o, T)$ and $(s, r, ?, T)$, and time interval prediction queries $(s, r, o, ?)$. In our setting, KB incompleteness exists at all times — the test fold may include instances from any interval in time, arbitrarily overlapping train and validation fold instances.[2]

7.2.3 TKBC Systems

Work on TKBC models adopts a common style for extending $\phi(s, r, o)$ in equation 7.1, to temporal score $\phi(s, r, o, t)$. Lacroix et al. [2020] embed each time instant t to vector $\boldsymbol{t}$ and use the form $\langle \boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o^\star, \boldsymbol{t} \rangle$ (called TNT-ComplEx). This can be interpreted as any *one* of $\boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o^\star$ becoming $\boldsymbol{t}$-dependent. Goel et al. [2020] make *both* subject and object embeddings time-dependent; the 'diachronic' embedding $\boldsymbol{e} \in \mathbb{R}^D$ of entity e is characterized by $\boldsymbol{e}_t[d] = a_e[d] \sin(w_e[d]\, t + b_e[d])$, where $d \in D$ and the sinusoidal nonlinearity affords the capacity to switch "entity features" on and off with time t. HyTE [Dasgupta et al., 2018] model $\boldsymbol{t} \in \mathbb{R}^D$, $\|\boldsymbol{t}\|_2 = 1$ and project *all* of $\boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o$ on to $\boldsymbol{t}$: $\boldsymbol{x} \downarrow \boldsymbol{t} = \boldsymbol{x} - (\boldsymbol{x} \cdot \boldsymbol{t})\boldsymbol{t}$, where $\boldsymbol{x} \in \{\boldsymbol{e}_s, \boldsymbol{r}, \boldsymbol{e}_o\}$. In all cases, time-dependent entity embeddings are plugged into standard scoring functions like DistMult, CX, or SimplE [Kazemi and Poole, 2018]. A very different approach [García-Durán et al., 2018] encodes the string representation of relation and time with an LSTM, which is used in TransE (TA-TransE) or DistMult (TA-DM).

These formulations do not directly model recurrences of a relation or interactions (e.g., mutual exclusion) between relations. There is some prior work on explicitly providing ordering constraints between relations (e.g., born, married, died) [Jiang et al., 2016]. In contrast, TIMEPLEX assumes no such additional engineered inputs; it has explicit components to enable learning of temporal (soft) constraints, as model weights, jointly with embeddings of entities, relations, and time instants.

[2]A different TKBC task studies only future fact predictions [Trivedi et al., 2017, Jin et al., 2019].

7.2.4 Standard Evaluation Schemes

Before designing TIMEPLEX, we discuss evaluation issues for TKBC.

Link Prediction: Link prediction queries in KBC are of the form $(s, r, ?)$ with a gold response o^*. Similarly, for TKBC they are of the form $(s, r, ?, T)$. The cases of $(?, r, o)$ and $(?, r, o, T)$ are symmetric and receive analogous treatment. "Unfiltered" link prediction performance is evaluated by finding the rank of o^* in the list of all entities ordered by decreasing score ϕ assigned by the model, and computing MRR. Other measures include the fraction of queries where o^* is recalled within the top 1 or top 10 ranked predictions (HITS@1 and HITS@10).

A query may have multiple correct answers. A model must not be penalized for ranking a different *correct* entity over o^*. In KBC this is achieved by filtering out all correct entities above o^* in ranked list before computing the metrics. In TKBC, filtering requires additional care, as depicted in Table 7.1. We develop time-aware filtering in Section 7.3.2.

Time Prediction: Time prediction queries of the form $(s, r, o, ?)$ will require comparing a gold time interval $T^* = [t^*_b, t^*_e]$ with a predicted interval $T^{\text{pr}} = [t^{\text{pr}}_b, t^{\text{pr}}_e]$. Since this task is relatively less studied, evaluation metrics have not yet been standardized. One might adapt the TAC metric popular in Temporal Slot Filling [Ji et al., 2011, Surdeanu, 2013]. Adapted to TKBC, TAC[3] will compute a score as $\frac{1}{2}\left[\frac{1}{1+|t^*_b - t^{\text{pr}}_b|} + \frac{1}{1+|t^*_e - t^{\text{pr}}_e|}\right]$. Unfortunately, TAC score is not entirely satisfactory for this task. For instance, TAC will assign the same merit score when gold interval [10,20] is compared with predicted interval [5,15], versus when gold [100,200] is compared with prediction [95,195]. However, a human would judge the latter more favorably, because a 5-minute delay in a 10-minute trip would usually be considered more serious than in a 100-minute journey. In response, we investigate alternative evaluation metrics inspired by bounding box evaluation protocols from Computer Vision, in Section 7.3.1.

7.3 Evaluation Metrics and Filtering

The preceding discussion motivates why we need clearly-thought-out filtering and evaluation schemes, not only for time interval prediction queries, but also because time affects link prediction evaluation in subtle but fundamental ways. This section addresses both issues.

7.3.1 Time Interval Prediction

One possible way to evaluate time prediction is to adapt measures to compare bounding boxes in computer vision, e.g., Intersection Over Union (IOU): $\text{IOU}(T^*, T^{\text{pr}}) = \frac{\text{vol}(T^* \cap T^{\text{pr}})}{\text{vol}(T^* \cup T^{\text{pr}})} \in [0, 1]$,

[3]TAC's original score compares gold and predicted *bounds* on begin and end of an interval. This formula is its adaptation, where begin and end are each a specific time point.

where vol for our case simply refers to the size of the interval. Unfortunately, IOU loses discrimination once $T^* \cap T^{\text{pr}} = \varnothing$; e.g., IOU$([1,2],[3,4])$ = IOU$([1,2],[30,40]) = 0$. This has been noticed in computer vision as well, and a metric called gIOU been introduced [Rezatofighi et al., 2019]:

$$\text{gIOU}(T^*, T^{\text{pr}}) = \text{IOU}(T^*, T^{\text{pr}}) - \frac{\text{vol}((T^* \Cup T^{\text{pr}}) \setminus (T^* \cup T^{\text{pr}}))}{\text{vol}(T^* \Cup T^{\text{pr}})} \in (-1, 1]. \qquad (7.2)$$

$T^* \Cup T^{\text{pr}}$ is the smallest single contiguous interval (**hull**) containing all of T^* and T^{pr}. E.g., $[1,2] \Cup [30,40] = [1,40]$.

gIOU can be negative, which is not ideal for a performance metric that is aggregated over instances. A simple fix (gIOU$'$) is to scale it to [0,1] via $(\text{gIOU}+1)/2$, but we notice that the tiniest overlap between T^* and T^{pr} yields gIOU$'$ to be at least half, regardless of vol(T^*) or vol(T^{pr}). Based on all these considerations, we propose a novel *affinity enhanced IOU*:

$$\text{aeIOU}(T^*, T^{\text{pr}}) = \frac{\max\{1, \text{vol}(T^* \cap T^{\text{pr}})\}}{\text{vol}(T^* \Cup T^{\text{pr}})} \qquad (7.3)$$

When $T^* \cap T^{\text{pr}} = \varnothing$, the denominator includes "wasted time", reducing aeIOU. The '1' in the numerator represents the smallest granularity of time in the data (see Section 7.2.2).

Comparison of Evaluation Metrics: A good time interval prediction metric (M) must satisfy the property (P) that: if two predicted intervals have intersections of the same size (possibly zero) with the gold interval, then the prediction that has a smaller hull with the gold interval should be scored higher by M. Formally, let T^{pr_1} and T^{pr_2} be two predictions made for T^*.

Property P**:** Let vol$(T^* \cap T^{\text{pr}_1})$ = vol$(T^* \cap T^{\text{pr}_2})$. Then, $M(T^*, T^{\text{pr}_1}) > M(T^*, T^{\text{pr}_2})$ if and only if vol$(T^* \Cup T^{\text{pr}_1})$ < vol$(T^* \Cup T^{\text{pr}_2})$.

Theorem: IOU and gIOU$'$ do not satisfy property P, whereas aeIOU satisfies it.

Proof that aeIOU satisifies P**:** For a fixed vol$(T^* \cap T^{\text{pr}})$, we have aeIOU$(T^*, T^{\text{pr}}) \propto 1/\text{vol}(T^* \Cup T^{\text{pr}})$ (see Eqn 7.3). Hence, aeIOU satisfies property P.

Proof that IoU and gIOU do not satisfy P**:**

IoU: This metric gives a score of 0, if the predicted interval does not intersect with the gold, irrespective of the hull. Hence IoU do not satisfy property P. Suppose the gold interval T^* [2002,2005], and consider two predictions, [1999,2001] and [1997,1999]. For both predictions, vol$(T^* \cap T^{\text{pr}}) = 0$, resulting in same scores for both predictions irrespective of the hull and hole.

gIoU: Let us look at the following example. Suppose the gold interval T^* [2002,2005], and consider two predictions, [1999,2001] and [1900,2001]. For both predictions, vol$((T^* \Cup T^{\text{pr}}) \setminus (T^* \cup T^{\text{pr}}))$ = 0, so the hull for the two predictions will be ignored (see Eqn 7.2), resulting in same scores for both predictions. Hence gIoU does not satisfy property P.

This suggests that aeIOU is a more defensible metric for our task, compared to other alternatives.

7.3.2 Link Prediction

Test query: $(s = \text{French National Assembly}, r = \text{has member}, o = ?, T^* = [2000, 2003])$							
Candidates o, system ordered	Known duration of o (any fold)	**Method 1** Unfiltered	**Method 2** Time-insensitive	**Method 3** Time-sensitive 2000	2001	2002	2003
Pierre	$[2002, 2003]$	1	0	1	1	0	0
Paul	$[2003, 2008]$	1	0	1	1	1	0
Alain	$[2008, 2009]$	1	0	1	1	1	1
Claude	$[2000, 2003]$	1	0	0	0	0	0
Jean	-	-	-	-	-	-	-
Time-sensitive rank of *Jean*		1+4=5	1+0=1	1+3=4	1+3=4	1+2=3	1+1=2

Table 7.1: *Jean* is the gold answer (o^*) to the query $(s, r, ?, T)$ shown above. Rows are ranked system predictions, which may be seen with same s and r for different intervals (Column 2). Columns 3–4 show the filtering of existing methods (1:unfiltered, 0:filtered). Columns 5–8 (Method 3, our proposal) show the filtering for each time instant. The bottom row shows ranks of *Jean* as computed by different methods. Existing methods over- or under-estimate performance. Method 3 assigns *Jean* a rank of 3.25, which is the average of the filtered ranks $\{4, 4, 3, 2\}$ for each time instant in $[2000, 2003]$.

We first illustrate the unique challenges offered by TKBC link prediction queries through an example in Table 7.1. The query asks for the name of a person who was a member of the French National Assembly in the time interval $[2000, 2003]$. Let the gold answer (object) o^* be Jean, which is ranked at the fifth position by the system that is being evaluated. All four entities above Jean are seen with the same subject and relation in the data, but for different time intervals. E.g., Pierre is also a member of the assembly, but during $[2002, 2003]$. The key question is: how should the four entities above Jean be filtered to compute its final rank?

We argue (Table 7.1) that existing filtering approaches are unsatisfactory. Dasgupta et al. [2018] underrate model performance by not performing any filtering (Method 1). In this example, the model is penalized for Claude, even though the time-interval for Claude exactly matches the query. On the other hand, García-Durán et al. [2018] and Jin et al. [2019] ignore time information altogether and filter out *all* entities seen with gold (s, r). This can greatly reduce the filtered rank of the system prediction, and thus overestimate system quality (Method 2). For instance, the model is not penalized for predicting Alain, even though its membership interval has no overlap with the query interval.

Ideally, filtering must account for the overlap between the query time interval and the time intervals associated with system-proposed entities. We propose such a filtering strategy (Method 3). We split the query interval into time instants, and compute a filtered rank for each time point independently. Entities that have full time overlap (or no overlap) will always (respectively, never) get filtered for a time instant. Partially overlapping entities will get filtered in only overlapping instants (e.g., 2 out of 4 for Pierre). After computing filtered ranks for each time instant, we output the final rank as an average of all such filtered ranks. In this example, this approach will compute the average of $\{4, 4, 3, 2\}$, which is 3.25. This average rank is used when computing standard metrics like MRR and HITS@10.

7.4 The Proposed TIMEPLEX Framework

Similar to TNT-ComplEx, TIMEPLEX learns complex-valued entity, relation and time instant embedding vectors. However, it differs from TNT-ComplEx in several ways. (1) Its base scoring function $\phi^{TX}(s, r, o, t)$ adds several products of three embeddings, instead of a single four-way product (Section 7.4.1). (2) It has a fully automatic mechanism to introduce additional features to capture potentially recurrent nature of a relation, as well as temporal interactions between pairs of relations (Section 7.4.2). (3) It uses a two-phase training (Section 7.4.3) curriculum that estimates first the embeddings and then novel additional parameters. (4) Its testing protocol can output a missing time-interval T for time-interval prediction queries (Section 7.4.4).

7.4.1 TIMEPLEX Base Model

We replace $\phi^{CX}(s, r, o)$ with:

$$\phi^{TX}(s, r, o, t) = \langle \boldsymbol{e}_s, \boldsymbol{r}^{\text{SO}}, \boldsymbol{e}_o^\star \rangle + \alpha \langle \boldsymbol{e}_s, \boldsymbol{r}^{\text{ST}}, \boldsymbol{t}^\star \rangle + \beta \langle \boldsymbol{e}_o, \boldsymbol{r}^{\text{OT}}, \boldsymbol{t}^\star \rangle + \gamma \langle \boldsymbol{e}_s, \boldsymbol{e}_o, \boldsymbol{t}^\star \rangle. \tag{7.4}$$

Here, $\boldsymbol{e}_s, \boldsymbol{e}_o, \boldsymbol{t} \in \mathbb{C}^D$, whereas each r is embedded as a concatenation of three such vectors $(\boldsymbol{r}^{\text{SO}}, \boldsymbol{r}^{\text{ST}}, \boldsymbol{r}^{\text{OT}})$, and hence requires three times the parameters. $\boldsymbol{r}^{\text{ST}}$ represents a relation which is true for entity s at time t (similarly for $\boldsymbol{r}^{\text{SO}}$ and $\boldsymbol{r}^{\text{OT}}$). And α, β and γ are hyperparameters.

Jiang et al. [2016] observed that several relations attach to a subject or object only at specific time points. E.g., subject Barack Obama was president in 2009, regardless of the object United States. In such cases, the formulation above is fully expressive.

To extend from single time instants t to an interval T, we propose

$$\phi^{TX}(s, r, o, T) = \textstyle\sum_{t \in T} \phi^{TX}(s, r, o, t). \tag{7.5}$$

7.4.2 Relation Recurrence and Pair Scores

We extend TIMEPLEX's base model via additional (soft) temporal constraints that can help in better assessing the validity of a tuple. We aim to capture three types of temporal constraints:

Relation Recurrence: Many relations do not recur for a given entity (e.g., a person is born only once). Some relations recur with fixed periodicity (e.g., Olympic games recur every four years, with rare exceptions). Recurrences of other relations may be distributed around a mean time period.

Ordering Between Relations: A relation precedes another, for a given entity. E.g., *personBornYear* must precede *personDiedYear* for a given subject entity (person), if the latter exists for the given entity.

Time Gaps Between Relations: The difference in time instants of two relations (wrt to an entity) may be distributed around a mean, e.g., *personDiedYear* minus *personBornYear* has a mean of about 70 with some observed variance.

The first constraint concerns single relations, whereas the latter two concern pairs of relations. Jiang et al. [2016] attempted to capture relation ordering constraints as model regularization, but their approach does not take into account time differences. Nor does it model relation recurrence.

Basic TIMEPLEX may not be able to learn these constraints from data either, since each time instant is modeled as a separate embedding with *independent* parameters — it has no explicit understanding of the difference between two time instants. In response, we augment TIMEPLEX with additional features that capture how soon an event recurs, or how soon after the occurrence of one relation, another relation is likely to follow. We define two scoring functions ϕ^{Rec} and ϕ^{Pair} for these two cases, to be aggregated with ϕ^{TX} (eqn. 7.4).

Inspired by García-Durán and Niepert [2018], we model time gaps as drawn from Gaussian distributions. We use $\mathcal{N}(x|\mu, \sigma)$ to denote the probability density of a Gaussian distribution with mean μ and std deviation σ at the time (difference) value x. We denote as $\mathcal{T}_{tr}$ all tuples in the train fold. While computing recurrence features, all training tuples of the form (s, r, o, T) are reduced to (s, r, o, t), i.e., with a singleton time interval, where $t = t_b$, the start time of T.

Recurrence Score: We say that (s, r, o) *recurs* if there are at least two distinct intervals T such that $(s, r, o, T) \in \mathcal{T}_{tr}$. If there are at least K^{Rec} distinct pairs (s, o) such that (s, r, o) recurs, then r is considered *recurrent*. K^{Rec} is a hyperparameter.

TIMEPLEX estimates a fact recurrence score, $\phi^{\text{Rec}}(s, r, o, T)$, as follows:

1. If $(s, r, o, \star) \notin \mathcal{T}_{tr}$, set $\phi^{\text{Rec}} = 0$.

2. Else, if r is not recurrent, set $\phi^{\text{Rec}} = b_r$. This allows the model to learn to penalize repetition of relations that do not recur.

3. Find time gap (δ) to its closest recurrence:

$$\delta = \min_{\{(s,r,o,t') \in \mathcal{T}_{tr}:\, t' \neq t\}} |t - t'|. \tag{7.6}$$

Then, set

$$\phi^{\text{Rec}}(s, r, o, T = [t_b, t_e]) = \phi^{\text{Rec}}(s, r, o, t_b) = w_r \mathcal{N}\big(\delta|\mu_r, \sigma_r\big) + b_r. \tag{7.7}$$

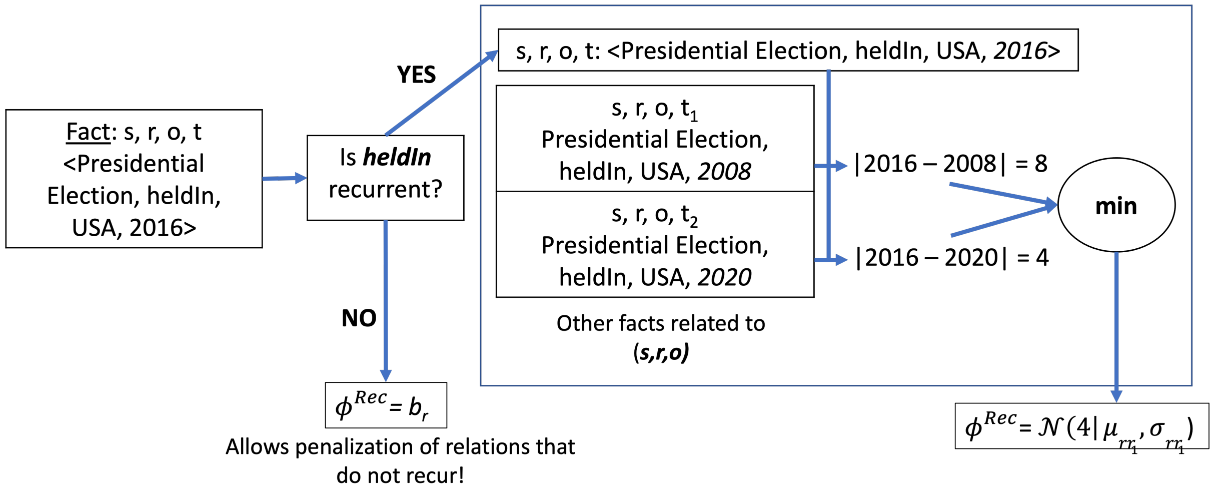

Fact: s, r, o, t <Presidential Election, heldIn, USA, 2016>
Is heldIn recurrent?
YES
NO
$\phi^{Rec} = b_r$
Allows penalization of relations that do not recur!
s, r, o, t: <Presidential Election, heldIn, USA, 2016>
s, r, o, t1 Presidential Election, heldIn, USA, 2008
s, r, o, t2 Presidential Election, heldIn, USA, 2020
Other facts related to (s,r,o)
|2016 – 2008| = 8
|2016 – 2020| = 4
min
$\phi^{Rec} = \mathcal{N}(4|\mu_{rr_1}, \sigma_{rr_1})$

Figure 7.1: This module attempts to capture fact recurrence. For tuple - *'Presidential Election, heldIn, USA, 2016'*, we first check if "*heldIn*" is a recurrent relation. If yes, we look for other facts of *'Presidential Election, heldIn, USA'* occurring at different time points. We identify the min time-difference and compute its validity w.r.t. typical relation recurrence time. For non-recurrent relations only a bias term is learnt.

For each recurrent relation r, our model learns three new parameters: μ_r, σ_r, and b_r. Intuitively, $\mathcal{N}(\cdot|\mu_r, \sigma_r)$ represents a distribution of typical durations between two recurring instances of a relation (with a specific subject and object entity) and b_r is the bias term. For non-recurrent relations, only the bias b_r is learnt.

Intuitively, ϕ^{Rec} should penalize the proposed (s, r, o, T) if δ is not close to the mean gap μ_r. For example, (Presidential election, held in, USA, 2017) should be penalized, if (Presidential election, held in, USA, 2016) is known, and the event reoccurs every 4 years ($\mu_r = 4, \sigma_r \approx 0$). See Figure 7.1 for an illustrative summary of score computation steps.

Relation Pairs Score: In an analogous manner, TIMEPLEX also learns soft time constraints between pairs of relations. We describe this mechanism for subjects; objects are handled analogously. For each relation pair (r, r'), we maintain four parameters, $\mu_{rr'}$, $\sigma_{rr'}$, $b_{rr'}$ and $w_{rr'}$, whose purpose we will describe presently. As with recurrence scores, all training tuples (s, r, o, T) are reduced to (s, r, o, t), where $t = t_b$, the start time of T. Given the candidate tuple (s, r, o, t) to score, we collect fact tuples $\{f_i = (s, r_i, o_i, t_i) \in \mathcal{T}_{tr}, r_i \neq r\}$ having the same subject but a different relation, into the set called $\text{KB}^{\text{Pair}}(s)$. The i^{th} tuple in $\text{KB}^{\text{Pair}}(s)$ is scored as $sc(f_i) = \mathcal{N}(t - t_i|\mu_{rr_i}, \sigma_{rr_i}) + b_{rr_i}$. This represents the contribution of f_i in the validity of candidate tuple, based on their (signed) time difference, and typical time differences observed between these two relations. $\phi^{\text{Pair}}_{\text{sub}}$ needs to aggregate these over f_i. The (trained) parameter $w_{rr'}$ measures how much the times associated with r' influence our belief in (s, r, o, t). Using these,

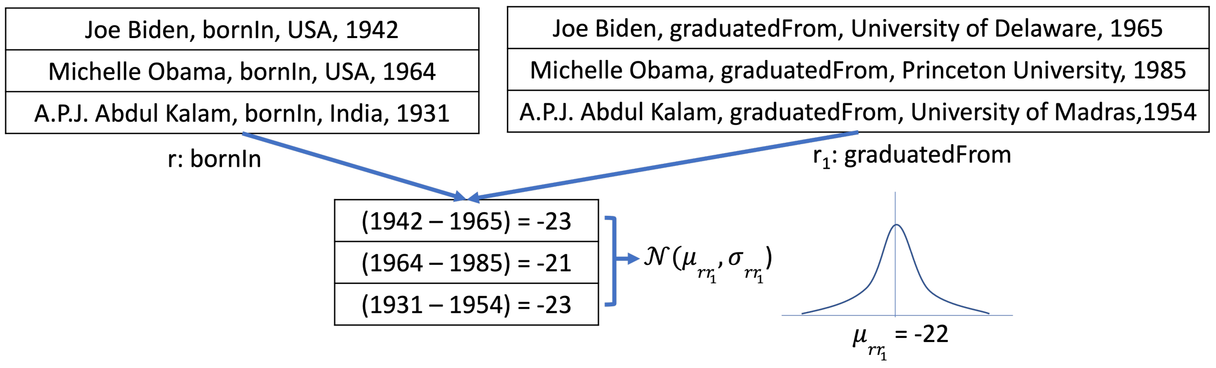

Joe Biden, bornIn, USA, 1942
Michelle Obama, bornIn, USA, 1964
A.P.J. Abdul Kalam, bornIn, India, 1931
Joe Biden, graduatedFrom, University of Delaware, 1965
Michelle Obama, graduatedFrom, Princeton University, 1985
A.P.J. Abdul Kalam, graduatedFrom, University of Madras,1954
r: bornIn
r1: graduatedFrom
(1942 – 1965) = -23
(1964 – 1985) = -21
(1931 – 1954) = -23
$\mathcal{N}(\mu_{rr_1}, \sigma_{rr_1})$
μ_{rr_1} = -22

Figure 7.2: The data statistics are pre-computed for model training. We model time gaps as drawn from gaussian distributions. The image shows pre-training data statistics collection strategy for a specific relation pair *'bornIn, graduatedFrom'*. *'Joe Biden'* was born in *'1942', while he graduated in 1965. While, 'Michelle Obama' was born in '1964', while she graduated in 1985. Similar facts can be used to compute the average age of a person when he graduates (here it is 22). Such statistics are computed for all relation pairs.*

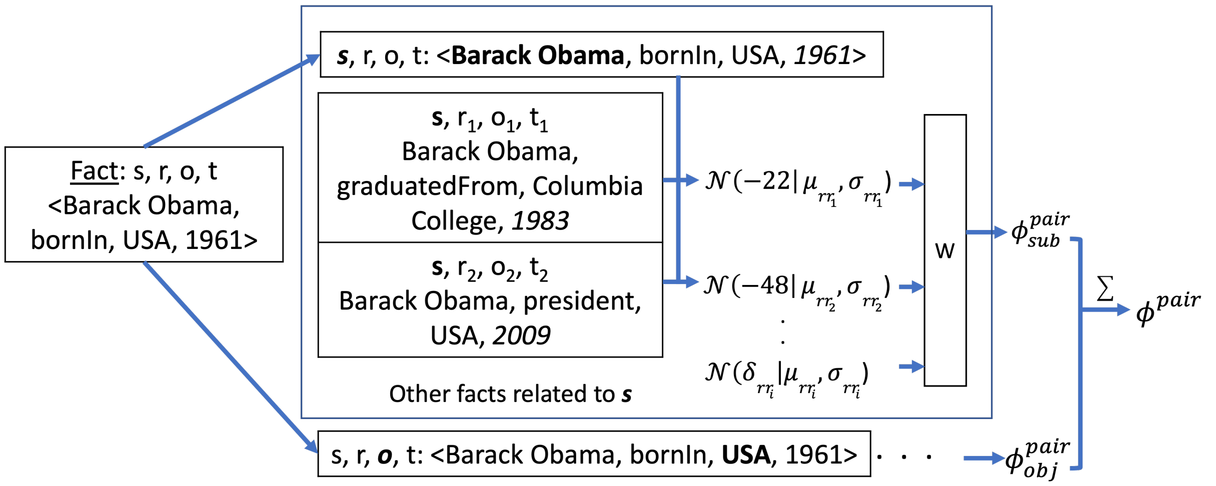

Fact: s, r, o, t <Barack Obama, bornIn, USA, 1961>
s, r, o, t: <Barack Obama, bornIn, USA, 1961>
s, r1, o1, t1 Barack Obama, graduatedFrom, Columbia College, 1983
s, r2, o2, t2 Barack Obama, president, USA, 2009
Other facts related to s
$\mathcal{N}(-22|\mu_{rr_1}, \sigma_{rr_1})$
$\mathcal{N}(-48|\mu_{rr_2}, \sigma_{rr_2})$
$\mathcal{N}(\delta_{rr_i}|\mu_{rr_i}, \sigma_{rr_i})$
w
ϕ^{pair}_{sub}
Σ
ϕ^{pair}
s, r, o, t: <Barack Obama, bornIn, USA, 1961>
ϕ^{pair}_{obj}

Figure 7.3: TIMEPLEX learns soft time constraints between pairs of relations. All training tuple with time intervals are reduced to facts with start time. For a dummy fact: *'Barack Obama, bornIn, USA, 1961'*. We first look at other facts related to the subject *'Barak Obama'* in dummy KB, like he graduated in *'1983'* and became president in year *'2009'*. We compute contribution of each relation-pair score in the validity of candidate tuple *'Barack Obama, bornIn, USA, 1961'*, based on their signed time difference and typical time differences observed between two relations. Here Barack graduated at the age of 22, which is very close to the mean age of graduation we computed from dummy data (see Figure 7.2). So, this is a positive signal. Here, we describe this mechanism for subjects, objects are handled analogously.

we define the weighted average

$$\phi^{\text{Pair}}_{\text{sub}}(s,r,o,t) = \sum_{f_i \in \text{KB}^{\text{Pair}}(s)} sc(f_i) \frac{\exp(w_{rr_i})}{\sum_{f_j} \exp(w_{rr_j})}.$$

A similar $\phi^{\text{Pair}}_{\text{obj}}$ score is computed for the object entity, and overall $\phi^{\text{Pair}} = \phi^{\text{Pair}}_{\text{sub}} + \phi^{\text{Pair}}_{\text{obj}}$. See Figure 7.2 and Figure 7.3 for an illustrative summary of score computation steps.

The **final scoring function** of TIMEPLEX is

$$\phi(s,r,o,T) = \phi^{TX}(s,r,o,T) + \kappa\phi^{\text{Pair}}(s,r,o,T) + \lambda\phi^{\text{Rec}}(s,r,o,T), \tag{7.8}$$

where κ and λ are hyperparameters.

7.4.3 Training

We train TIMEPLEX in a curriculum of two phases. In the first phase, we optimize embeddings for all entities, relations and time-instants by minimizing the log-likelihood loss using only the base model TX. We compute the probability of predicting a response o for a query $(s,r,?,T)$ as:

$$\Pr(o|s,r,T) = \frac{\exp(\phi^{TX}(s,r,o,T))}{\sum_{o'} \exp(\phi^{TX}(s,r,o',T))} \tag{7.9}$$

We can similarly compute the probability of predicting a response s for a query $(?,r,o,T)$ as:

$$\Pr(s|r,o,T) = \frac{\exp(\phi^{TX}(s,r,o,T))}{\sum_{s'} \exp(\phi^{TX}(s',r,o,T))} \tag{7.10}$$

We convert every $(s,r,o,T=[t_b,t_e]) \in \mathcal{T}_{tr}$ in time-instant format by enumerating all (s,r,o,t), for $t \in [t_b,t_e]$. We compute the probability of predicting a response t for a query $(s,r,o,?)$ as:

$$\Pr(t|s,r,o) = \frac{\exp(\phi^{TX}(s,r,o,t))}{\sum_{t'} \exp(\phi^{TX}(s,r,o,t'))} \tag{7.11}$$

Training of embeddings minimizes the **log-likelihood loss**:

$$-\sum_{\langle s,r,o,t\rangle \in KB^{\text{tr}}} \Big(\log\Pr(o|s,r,t;\theta) + \log\Pr(s|o,r,t;\theta) + \log\Pr(t|s,r,o;\theta) \Big) \tag{7.12}$$

In the second phase, we freeze all embeddings and train the parameters of the recurrence and pairs models. Here, too, we use the log-likelihood loss, except that ϕ^{TX} is replaced by the overall ϕ function. Parameters $\mu_{rr'}$ and $\sigma_{rr'}$ of the relation-pairs model component are not trained via backpropagation. Instead, they are fitted separately, using the difference distributions for the pair of relations in the training KB. This improves the overall stability of training.

7.4.4 Inference

At test time, for a link prediction query $(s,r,?,T)$ or $(?,r,o,T)$, TIMEPLEX ranks all entities in decreasing order of $\Pr(o|s,r,T)$ or $\Pr(s|r,o,T)$ scores. For time prediction, its goal is to output a predicted time duration T^{pr}. We first compute a probability distribution over time instants $\Pr(t|s,r,o) = \frac{\exp(\phi(s,r,o,t))}{\sum_{t' \in \mathfrak{T}} \exp(\phi(s,r,o,t'))}$. We then greedily coalesce time instants to output the best duration. For greedy coalescing, we tune a threshold parameter θ_r for each relation r using the validation fold. We then initialize the predicted interval T^{pr} as $\operatorname{argmax}_t \Pr(t|s,r,o)$. Then, as long as total probability of the interval, i.e., $\sum_{t \in T^{\text{pr}}} \Pr(t|s,r,o)$ is less than θ_r, we extend T^{pr} with the instant to its left or right, whichever has a higher probability.

7.5 Experiments

We investigate the following research questions. (1) Does TIMEPLEX convincingly outperform the best time-agnostic and time-aware KBC systems on link prediction and time interval prediction tasks? (2) Are recurrent and pairwise features helpful in the final performance? (3) Are TIMEPLEX's time embeddings meaningful, i.e., do they capture the passage of time in an interpretable manner? (4) Do TIMEPLEX predictions honor temporal constraints between relations?

7.5.1 Datasets & Experimental Setup

Datasets: We report on experiments with four standard TKBC datasets. WIKIDATA12k and YAGO11k [Dasgupta et al., 2018] are two knowledge graphs with a time interval associated with each triple. These contain relational facts like (David Beckham, plays for, Manchester United; [1992, 2003]). ICEWS14 and ICEWS05-15 [García-Durán et al., 2018] are two event-based temporal knowledge graphs, with facts from Integrated Crisis Early Warning System repository. These primarily include political events with timestamps (no nontrivial intervals). We consider the time granularity for interval datasets as 1 year, and for ICEWS datasets as 1 day.

See Table 7.2 for some salient statistics of the datasets we used for experiments. As already mentioned, Yago11k and Wikidata12k are interval based datasets. ICEWS14 and ICEWS05-15 are instant based datasets.

	YAGO11k	**WIKIDATA12k**	**ICEWS14**	**ICEWS05-15**
Entities	10622	12554	7128	10488
Relations	10	24	230	251
#Instants	251	237	365	4017
#Intervals	6651	2564	0	0
Train	16408	32497	72826	368962
Valid	2051	4062	8941	46275
Test	2050	4062	8943	46092

Table 7.2: Details of datasets used.

By experimenting across the spectrum, from 'point' events to facts with duration, we wish to ensure the robustness of our observations.

Methods compared: We compare against our reimplementations of CX, HyTE, TA-family, and TNT-ComplEx. In all cases we verify that our implementations give performance comparable to or better than what is reported in the literature. We combine HyTE and TA (see Section 7.2.3 for details on HyTE and TA), with scoring functions from TransE, DistMult and CX and present the best results. We compare against reported results in DE-SimplE (see Section 7.2.3 for details on DE-SimplIE).

Experimental Details: For all models, we optimize parameters with AdaGrad running for 500 epochs for all losses, with early stopping on validation fold. We control for an approximately comparable number of parameters and set dimensionality of 200 for all complex embeddings and 400 for all real embeddings. We follow other best practices in the literature, such as L2 regularization only on embeddings used in the current batch [Trouillon et al., 2016], adding inverted facts (o, r^{-1}, s, T) , using 1vsAll negative sampling [Dettmers et al., 2018] whenever applicable, and using temporal smoothing for ICEWS datasets [Lacroix et al., 2020].

Some instances in interval datasets have t_b or t_e missing. Following Dasgupta et al. [2018], we replace missing values by $-\infty$ or $+\infty$, respectively. For time prediction queries, we remove such instances from test sets. For ICEWS datasets we set $t_b = t_e$. For time interval prediction, all models use our greedy coalescing inference from Section 7.4.4.

For TIMEPLEX, we perform a grid search for all hyperparameters, and pick the best values based on MRR scores on valiations set.

Models	**Number of parameters**
HytE	d($\lvert E\rvert + \lvert T\rvert + \lvert R\rvert$)
DE-SimplE	2d($(3\delta + (1-\delta))\lvert E\rvert + \lvert R\rvert$)
TNTComplEx	2d($\lvert E\rvert + \lvert T\rvert + 4\lvert R\rvert$)
TIMEPLEX(base)	2d($\lvert E\rvert + \lvert T\rvert + 6\lvert R\rvert$)
TIMEPLEX	2d($\lvert E\rvert + \lvert T\rvert + 6\lvert R\rvert$) + 2($\lvert R\rvert^2$ + $\lvert R\rvert$)

Table 7.3: Number of parameters for each model. For HyTE we assume bucket size = 1 here. δ is the fraction of dimension to represent time in DA-SimplE model.

More details of hyperparameters and model training: The models discussed in this chapter train scalably on the datasets used for experiments (see Table 7.3 for all model parameter size) and can be easily fit into a single 12 GB NVIDIA Tesla K40 GPU. Our final model TIMEPLEX consist of a base model and two time-based gadgets. TIMEPLEX*(base)* takes less than 10 minutes to train on all datasets except for ICEWS05-15, where it takes 80 minutes. Table 7.4 lists best hyperparameters of TIMEPLEX(base) on various datasets. Both gadgets are trained independently in less than 10 minutes. The parameter λ=5.0 gave best results for interval datasets, while λ=1.0 gave best results on event datasets. On Yago11k κ=3.0, while for rest κ=0.0. The gadget weights are L2 regularized, with a regularization penalty of 0.002. In phase 2 training of the model we use 100 negative samples per correct fact (in this phase we train our time gadgets).

↓Datasets	**Learning Rate**	**Reg wt**	**Batch size**	**Temporal smoothing**	α	β	γ
YAGO11k	0.1	0.03	1500	0.0	5.0	5.0	0.0
WIKIDATA12k	0.1	0.005	1500	0.0	5.0	5.0	5.0
ICEWS05-15	0.1	0.005	1000	5.0	5.0	5.0	5.0
ICEWS14	0.1	0.005	1000	1.0	5.0	5.0	5.0

Table 7.4: Hyperparameters for training TIMEPLEX(base) model embeddings on various datasets, tuned on MRR for validation set. Temporal smoothing was found to help on ICEWS datasets, however it gave no improvement for interval datasets. We tuned the parameters in a staged manner - first we tune learning rate (lr), regularization weight (r), batch size(b), and temporal smoothing weight (ts). We performed a random search in the following ranges: $lr \in [0.0001, 1.0]$, $r \in [0.0001, 1.0]$, $b \in [100, 5000]$, and $ts \in [0.0001, 10.0]$. The models were most sensitive to regularization weight and learning rate. After finding best values for these parameters, we tuned α, β and γ weights for each dataset, doing a grid search over the set {0.0, 2.0, 5.0, 7.0, 10.0}

.

Time modelling details of TIMEPLEX*, HyTE:* Each dataset spans along a *time range*, with a certain *time granularity*, which can be year, month or day. TIMEPLEX learns a time embedding for every point in this time range, discretized on the basis of the dataset's granularity (years for the interval datasets WIKIDATA12k and YAGO11k, and days for ICEWS datasets). At training time, TIMEPLEX looks at a single time point at a time - for this, we sample a time point uniformly at random from the query interval $[t_b, t_e]$ associated with the fact. In contrast, HyTE maps each time point to bin (heuristically determined), making the data granularity coarser, and learns representation of these bins. HyTE looks at time points in an interval as well, but enumerates each interval fact to produce a separate fact for each time point beforehand.

Our method of sampling is efficient as the data size is unchanged. It also ensures each fact is sampled uniformly, not hurting link prediction performance by oversampling of long duration facts.

HyTE time prediction: HyTE can only predict a bin for the test fact. To convert predicted bins to years (or days), we take a mean of all years seen with the predicted bin and then do greedy coalescing to output time interval in years.

7.5.2 Results and Observations

Link prediction: Table 7.5 compares all algorithms for link prediction. We find that the best performing baseline among existing TKBC systems is TNT-ComplEx model. TIMEPLEX outperforms TNT-ComplEx by over 3 MRR points in ICEWS datasets. Its gains (3.25 and 5.6 pts) are even more pronounced in interval datasets. All differences are statistically significant using

Dataset→	WIKIDATA12k			YAGO11k			ICEWS05-15			ICEWS14		
↓Methods	MRR	HITS@1	HITS@10	MRR	HITS@1	HITS@10	MRR	HITS@1	HITS@10	MRR	HITS@1	HITS@10
CX	24.82	14.30	48.90	18.14	11.46	31.11	48.68	37.00	72.63	45.50	33.87	69.73
TA (CX)	22.78	12.69	46.00	15.24	9.36	26.26	49.23	37.6	72.69	40.97	29.58	63.87
HyTE (TransE)	25.28	14.70	48.26	13.55	3.32	29.81	23.73	3.11	62.76	24.91	2.98	65.30
DE-SimplE	25.29	14.68	49.05	15.12	8.75	26.74	51.30	39.20	74.80	52.60	41.80	72.50
TNT-ComplEx	30.10	19.73	50.69	18.01	11.02	31.28	60.58	51.14	78.50	56.72	47.04	75.40
TIMEPLEX (base)	32.38	22.03	52.79	18.35	10.99	31.86	63.91	**54.62**	81.42	60.25	51.29	77.05
TIMEPLEX	**33.35**	**22.78**	**53.20**	**23.64**	**16.92**	**36.71**	**63.99**	54.51	**81.81**	**60.40**	**51.50**	**77.11**

Table 7.5: Link prediction performance across four datasets. The last row reports results for TIMEPLEX(base) augmented with pair/recurrent features.

paired t-test with $p < 0.01$. These scores establish a new state of the art for link prediction on time-interval datasets, at the time of writing this dissertation.

Table 7.5 reports link prediction performance using the filtering strategy discussed in Section 7.3.2. We also report the performance of most competitive baseline and TIMEPLEX, using a filtering strategy that does not enumerate time points in an interval and filters out entities on exact matching time-interval (See Table 7.6). Note that our model consistently outperforms TNT-ComplEx, even with a stricter filtering.

Datasets→	**WIKIDATA12k**			**YAGO11k**		
↓Methods	MRR	HITS@1	HITS@10	MRR	HITS@1	HITS@10
TNT-ComplEx	27.35	17.59	48.51	15.78	10.21	28.64
TIMEPLEX	**30.61**	**20.79**	**51.78**	**22.77**	**16.33**	**36.3**

Table 7.6: Link prediction performance of the best models using a filtering strategy that does not enumerate time points in an interval, and filters on an exact match instead. We find that while TIMEPLEX convincingly outperforms the previous SOTA TNT-ComplEx using this filtering strategy as well.

Time prediction: We are the first to look at the task of predicting *time intervals*, and we report scores of the same on our novel aeIOU metric along with previously proposed evaluation metrics (Table 7.7). We see that TIMEPLEX outperforms TNT-ComplEx on both datasets, with a huge 11+ pt aeIOU jump on the Yago11K dataset. It is also noteworthy that even the base model of TIMEPLEX is consistently better than TNT-ComplEx across all experiments. Similar trends are observed in other metrics too.

On Pair/recurrent features: We find that recurrent features are very helpful in both interval datasets, and significantly improve link prediction performance. Relation pair features particularly help in YAGO11k – an over 5 pt aeIOU boost in time prediction, but on WIKIDATA12k

Datasets→	YAGO11k				WIKIDATA12k			
↓Methods	TAC	gIOU	IOU	aeIOU	TAC	gIOU	IOU	aeIOU
HyTE	5.59	15.96	1.91	5.41	6.13	14.55	1.40	5.41
TNT-Complex	9.90	20.78	3.99	8.40	26.98	36.63	11.68	23.25
TIMEPLEX (base)	16.57	26.22	5.48	14.21	30.36	39.2	**13.20**	26.20
TIMEPLEX	**22.66**	**32.64**	**8.24**	**20.03**	**30.71**	**39.34**	13.15	**26.36**

Table 7.7: Time prediction performance using - TAC, gIOU, IOU and aeIOU

they make only a marginal difference. On inspecting the datasets, we find that 78% of entities in WIKIDATA12k are seen with a single, recurring relation only (such as *award received*, or *member of sports team*) – hence relation pair features cannot help.

ICEWS datasets are scraped from news events. On inspecting the datasets, we find that the events do not follow any temporal ordering and are fairly non-regular in event recurrence as well. Hence, TIMEPLEX's improvements over the base model are limited. We further investigate the differing performance on datasets and the value of pair features in the next section.

7.5.3 Diagnostics

Time gap vs. embedding distances: Longevity of relations, or gaps between events, are often determined by physical phenomena that are smooth and continuous in nature. Therefore, we expect the embedding of the year 1904 to be closer to that of 1905 compared to the embedding of 1950.

To validate this hypothesis, we compute mean L2 distance between embeddings of time instants which are apart by a given time gap. To filter noise, we drop instant pairs with extreme gaps that have low support (less than 30). For WIKIDATA12k we used embeddings of years $[1984, 2020]$ and for YAGO11k we use embeddings of years $[1958, 2017]$.

Figure 7.4 shows that L_2 distance between pairs of time embeddings increases with the actual year gap between them. This strongly suggests that the time embeddings learnt by TIMEPLEX naturally represents physical time.

Temporal ordering of relation pairs: Both YAGO11k and WIKIDATA12k contain relations with temporal dependencies, e.g., *bornInPlace* should always precede *diedInPlace* for the same person. We now study whether TIMEPLEX models are able to learn these natural constraints from data.

We first exhaustively extract all relation pairs (r_1, r_2), where the existence of both $(s, r_1, \star, t_1)$ and $(s, r_2, \star, t_2)$ is accompanied by $t_1 < t_2$ at least 99% of the time, with a minimum support of 100 entities s. Table 7.8 and 7.9 lists automatically extracted high confidence relation orderings seen in Yago11k and Wikidata12k datasets respectively. These orderings are used to guide TIMEPLEX at the time of training.

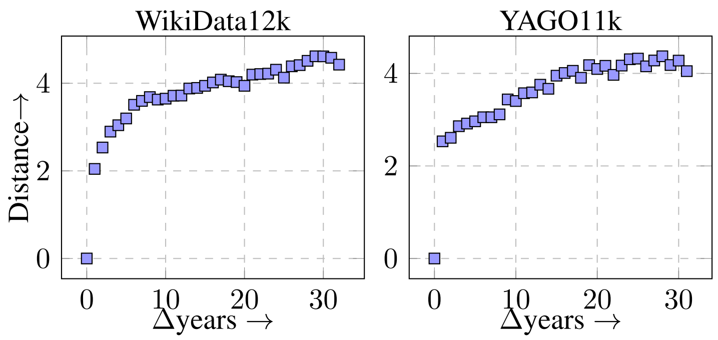


Figure 7.4: L_2 distances (y-axis) between TIMEPLEX time embeddings increase with time gap (x-axis).

graduatedFrom → diedIn
graduatedFrom → hasWonPrize
wasBornIn → graduatedFrom
wasBornIn → diedIn
wasBornIn → isAffiliatedTo
wasBornIn → hasWonPrize
wasBornIn → playsFor
wasBornIn → worksAt
wasBornIn → isMarriedTo
isAffiliatedTo → diedIn
worksAt → diedIn
isMarriedTo → diedIn

Table 7.8: High confidence (99%) relation orderings extracted from YAGO11k.

educated at → position held
educated at → employer
educated at → member of
educated at → award received
educated at → academic degree
educated at → nominated for
instance of → head of government
residence → award received
academic degree → nominated for
spouse → position held
located in the administrative territorial entity → award received

Table 7.9: High confidence (99%) relation orderings extracted from WIKIDATA12k

We now verify whether TIMEPLEX honors r_1 before r_2 when making predictions. For each query $(?, r, o, t)$ in the test set, we check whether the top model prediction violates any known temporal ordering constraint in this list. For example, for a query *(?, **hasWonPrize**, Nobel Prize, **1925**)*, if the model predicted *Barack Obama* and the KB already had Barack Obama born

	YAGO11k	**WIKIDATA12k**
CX	10.04	0.7
HyTE	7.2	0.4
TNT-ComplEx	8.82	0.3
TIMEPLEX (Base)	6.6	0.3
TIMEPLEX	**1.9**	**0.2**

Table 7.10: Ordering constraint violations among top predictions of various models (% of facts in test set).

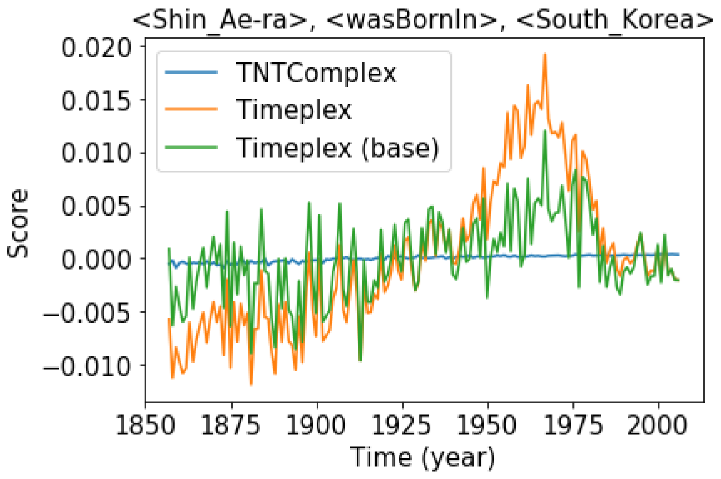
in Hawaii in 1961, then this will be considered as an ordering violation. Table 7.10 reports the number such violations as fraction of test set size. TIMEPLEX significantly reduces such errors for YAGO11k; this is also reflected in its superior time prediction performance. For WIKIDATA12k, the errors for TIMEPLEX (base) are already low, hence pair features are not found to be particularly helpful.

As an illustrative example, we consider the time prediction query *(Shinae-ra, wasBornIn, South Korea, ?)*, with the gold answer 1969. The only other fact seen for *Shinae-ra* in the train KB is *(Shinae-ra, isMarriedTo, ChaIn-Pyo, (1995, -))*. TIMEPLEX predicts 1967 for this query (earning an aeIOU credit of 33.33). However, TNTComplEx predicts 2013 (earning almost no credit) – this also highlights that it does not capture commonsense that a person can marry only after they are born.

Digging deeper, we plot the normalized scores for this query in time range $[1850, 2010]$ in Figure Table 7.11(a). The peak around 1967 for the TIMEPLEX plot can be attributed to the fact that mean difference for *isMarriedTo* and *wasBornIn* relations is around 30 in the dataset. Standard tensor factorization models like TNT-ComplEx are unable to exploit this, but our Pair features provide a way to the model to make very reasonable predictions.

In another time prediction query (see Table 7.11 (b)) *(Peter_Nowell, wasBornIn, Philadelphia, ?)*, with gold answer 1928, we show how, with limited background knowledge on the subject in question, TIMEPLEX can predict the gold time interval.

Ablation Study: To further explore the second research question – are recurrent and pairwise features helpful in the final performance? (discussed in Section 7.5), we perform an ablation study. In this study, we remove each component of TIMEPLEX (see equation 7.8) by making either κ=0 or λ=0, to understand the importance of each component. We report results on Yago11k dataset in Table 7.12. The study demonstrates that our recurrence feature significantly help in link prediction while out relation pair feature helps time-interval prediction.

	
Info about query e1 in train set:	
<Shin_Ae-ra, isMarriedTo, Cha_In-pyo>(1995, 3000)	
Gold answer	1969
Timeplex prediction	1967
Timeplex (base) prediction	1967
TNTComplEx prediction	2013
(a) TIMEPLEX, TIMEPLEX(base) both predict the correct answer but TNTComplEx cannot model that one cannot marry before birth.	

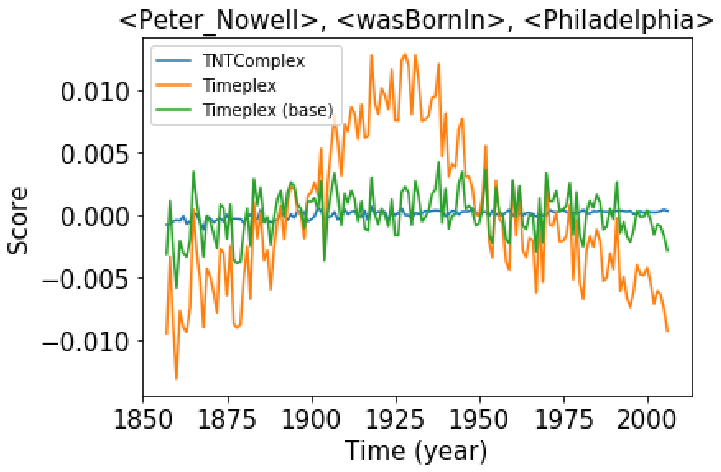
	
Info about query e1 in train set:	
<Peter_Nowell, graduatedFrom, Wesleyan_University>(1948, 3000) <Peter_Nowell, graduatedFrom, University_of_Pennsylvania>(1952, 3000)	
Gold answer	1928
Timeplex prediction	1928
Timeplex (base) prediction	1938
TNTComplEx prediction	1918
(b) TIMEPLEX(base) cannot model that one is unlikely to graduate at the age of 10. TIMEPLEX (base) and TNTComplEx do not have a clear vote like Timeplex.	

Table 7.11: Comparing time prediction performance of TIMEPLEX, TIMEPLEX(base) and TNTComplEx.

Prediction task→	**Link**			**Time interval**
↓Method	MRR	HITS@10	HITS@1	aeIOU@1
TIMEPLEX	23.64	16.92	36.71	20.03
TIMEPLEX-Pair	23.15	16.63	36.27	14.21
TIMEPLEX- Rec	18.93	11.46	32.74	20.03
TIMEPLEX- Pair - Rec	18.35	10.99	31.86	14.21

Table 7.12: Ablation study on Yago11k. Recurrence feature significantly help in link prediction while relation pair feature helps time-interval prediction.

7.6 Discussion

TIMEPLEX cannot exploit the influence that an entity can have on time difference distributions. For example, the life expectancy of a person (mean difference between *diedIn* and *bornIn* events) would be around 85 in Japan, but 54 in Lesotho. Extending our model to learn separate parameters for each $\langle$rel, entity$\rangle$ pair may be difficult due to sparsity. Also, recurrent facts may admit exceptions: Winter Olympics are held every 4 years except for 1992 and 1994. However, we do not expect even humans to do well in such cases. Exceptions like these are sparse and difficult to learn, except by rote.

7.7 Conclusion

We presented TIMEPLEX, a new TKBC framework, which combines representations of time with representations of entities and relations. It also learns soft temporal consistency constraints, which allow knowledge of one temporal fact to influence belief in another fact. TIMEPLEX exceeds the performance of existing TKBC systems. Diagnostics suggest that time embeddings are temporally meaningful, and TIMEPLEX makes fewer temporal consistency and ordering mistakes. We also argue that current evaluation schemes for both link and time prediction have limitations, and propose more meaningful schemes.

Relevant Publication

- [Jain et al., 2020b]: "Temporal Knowledge Base Completion: New Algorithms and Evaluation Protocols". Prachi Jain, Sushant Rathi, Mausam, Soumen Chakrabarti. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). November 2020.

Part III

Micro-Inference

Chapter 8

Improving the quality of inference rule-base for micro-inference

Thus far in this thesis, we have studied various methods to infer missing facts in structured canonical KBs; in particular macro inference methods. However, as discussed in Chapter 1, some facts are hard to express using structured KB predicates, but can be easily represented using natural language. Open KBs consist of fact tuples where one or more elements are free-form strings, eg. ('penicillin', 'kills', 'bacteria'). Open KBs are a viable alternative in situations where canonical KBs are difficult to curate and maintain. However, adding missing facts to Open KBs is challenging, due to their noisy nature. Also, macro-inference methods are hard to scale with the large number of predicates and entities in open KBs. Google's Knowledge Graph had grown to 500 billion facts on 5 billion entities by May 2020 [1]. Very large sized GPUs are required to just load the entity embeddings of any macro inference model. In this chapter, we discuss methods to improve the precision of such a rule base and eventually improve inference in open KBs.

8.1 Problem formulation

Micro inference involves inferring a novel fact from a single or a very small number of input facts, independent of other facts seen in the KB. The methods used for this task often rely on inducing inference rules from a corpus of facts [Schoenmackers et al., 2010, Nakashole et al., 2012, Berant et al., 2011, Galárraga et al., 2013, Berant, 2012, Pavlick et al., 2015, Hearst, 1992]. For example, given an inference rule base which has a rule, $[($ *X, has passport of, Y* $) \Rightarrow ($ *X, is citizen of, Y* $)]$, and a KB which has a fact (*Barack Obama, has passport of, USA*), then a query like (*Barack Obama, is citizen of, ?*) to an inference engine can be correctly answered

[1] `https://en.wikipedia.org/wiki/Google_Knowledge_Graph`

('USA' for query 1). Notice that this differs from macro inference which may use a larger neighborhood of related facts for inference.

In Table 8.1 we present a formal setup of the problem. In the problem formulation, a fact represents real-world knowledge in a triple format (s, r, o), example (Narendra Modi, is the PM of, India). Given a large amount of training data pairs ($x^{(m)}$,$y^{(m)}$) for m=1,2,3,...M, where $x^{(m)}$ is an instance of related facts $[(s_i, r_i, o_i), R, (s_j, r_j, o_j)]^{(m)}$; $R \in \{\equiv, \Rightarrow, \nRightarrow\}$ (micro inference setting) and y $\in$ [0,1] indicates the validity of x (note that the setup allows associating the likelihood of y taking a specific value). The goal is to learn a mapping function f:x $\rightarrow$ y to predict correctly on a new input x, using traditional machine learning or deep learning models by optimizing parameters on training data. Models to generate this form of input (inference rules) can also be built.

Training Data	($x^{(m)}$,$y^{(m)}$)$_{m=1,2,3,...M}$
Input	$x^{(m)}$ is a pair of related fact (s_i, r_i, o_i, R, s_j, r_j, o_j)$^{(m)}$; R $\in$ $\{\equiv, \Rightarrow, \nRightarrow\}$
Input example	(Narendra Modi, is PM of, India) $\Rightarrow$ (Narendra Modi, is citizen of, India)
Output	y $\in$ {0,1} 1 indicates x is valid and 0 indicated x is invalid.
Goal	f:x $\rightarrow$ y

Table 8.1: Micro inference task problem formulation.

8.2 Micro inference

A set of *inference rules* are used along with probabilistic models such as MLNs [Schoenmackers et al., 2008] or BLP [Raghavan et al., 2012] to produce proof chains and infer new facts. While scalable [Niu et al., 2011, Domingos and Webb, 2012], this is bound by the coverage and quality of the background knowledge – the set of inference rules [Clark et al., 2014].

This chapter will focus on generating a *high precision* subset of inference rules built over OpenIE KB [Etzioni et al., 2011]. Most existing large-scale corpora of inference rules are generated using distributional similarity, like argument-pair overlap [Schoenmackers et al., 2010, Berant et al., 2012]. Methods vary on the base representation, e.g., KB relations [Galárraga et al., 2013, Grycner et al., 2015], OpenIE relation phrases [Schoenmackers et al., 2010], SOL patterns [Nakashole et al., 2012], and dependency paths [Lin and Pantel, 2001]. Global transitivity (TNCF algorithm) in KBs is also exploited for improving recall [Berant et al., 2012]. The highest precision setting of TNCF ($\lambda = 0.1$) was released as a corpus (informally called CLEAN) of OpenIE inference rules.[2]

[2] *http://u.cs.biu.ac.il/~nlp/resources/downloads/predicative- entailment-rules-learned-using-local-and-global-algorithms*

	Antecedent	**Consequent**	**Y/N?**
1	(e_s, make a note of, e_o)	(e_s, write down, e_o)	Y
2	(e_s, offer wide range of, e_o)	(e_s, offer variety of, e_o)	Y
3	(e_s, make full use of, e_o)	(e_o, be used by, e_s)	Y
4	(e_s, be wounded in, e_o)	(e_s, be killed in, e_o)	N
5	(e_s, be director of, e_o)	(e_s, be vice president of, e_o)	N
6	(e_s, be a student at, e_o)	(e_s, be enrolled at, e_o)	N

Figure 8.1: Sample rules verified (Y) and filtered (N) by our method. Rules 4, 5 were correctly and 6 wrongly filtered.

Distributional similarity approaches have two fundamental limitations. First, they miss obvious commonsense facts, e.g., '(e_s, married, e_o) $\Rightarrow$ (e_s, knows, e_o)' — text will rarely say that a couple know each other. Second, they are affected by statistical noise and end up generating a variety of inaccurate rules (see rule 4 and 5 in Figure 8.1). Our early experiments with CLEAN revealed its precision to be about 0.49, not enough to be useful in practice. Precision is paramount for human-facing applications (such as IE-based demos). Also, inference rules have a multiplicative impact since one poor rule could generate many wrong KB facts.
Contributions: We investigate the hypothesis that *"knowledge-guided linguistic rewrites can provide independent verification for statistically-generated OpenIE inference rules"*. Our system KGLR's rewrites exploit the compositional structure of OpenIE relation phrases alongside knowledge in additional resources like Wordnet [Hirst and St Onge, 1998] and thesauri. KGLR independently verifies rules from inference rule corpora [Berant et al., 2012, Pavlick et al., 2015] and can be seen as an additional annotation on existing inference rules. The verified rules are 27 to 33 points more accurate than the original corpora and still retain a substantial recall. The inferred knowledge also has a precision boost of over 29 points. We release our KGLR implementation and its annotations on two popular rule bases — CLEAN and entailment/paraphrase subset of PPDB 2.0 (PPDB$_e$), for further use.[3]

8.3 Knowledge-based Linguistic Rewrites

Given a rule "$(e_s, r_1, e_o) \Rightarrow (e_s, r_2, e_o)$" or "$(e_s, r_1, e_o) \Rightarrow (e_o, r_2, e_s)$" we present KGLR, a series of rewrites of the relation phrase r_1 to prove r_2 (example in Fig 8.1). The algorithm is enabled by knowledge sources like Wordnet and Thesaurus.

Similar to this work, some past works used additional sources of knowledge. Weisman et al. [2012] studied inference between verbs (e.g., 'startle $\Rightarrow$ surprise'), but they get low (0.4) precision. Wordnet is used to generate inference rules for natural logic, improving noun-based

[3]https://github.com/dair-iitd/kglr

inference [Angeli and Manning, 2014]. They recognize relation entailments as a key missing piece. Natural logic semantics have also been added to a paraphrase corpus (PPDB 2.0).[4] Many of their features, e.g., lexical/orthographic, multilingual translation based, are complimentary.

KGLR performs the following rewrites to perform rule verification. Note that the last two rewrites deal with reversal of argument entity order in r_2; other rewrites use the original argument entity order.

Thesaurus Synonyms: Thesauri[5] typically provide a set of potential synonyms('produce' – 'make'), encompassing near-synonyms ('produce' – 'accomplish') and contextually synonymous words. Thesaurus synonyms are not that helpful for *generating* inference rules (or else we will end up with rules like 'produce ⇒ percolate'[6]). However, they are excellent in rule *v*erification as they provide evidence independent from statistical overlap metrics.

We allow any word/phrase w_1 in r_1 to be replaced by any phrase w_2 from its thesaurus synsets as long as w_2 is in r_2 and its POS and location in r_2 corresponds to r_1. To define a thesaurus synset, we tag w_1 with its POS and look for all thesaurus synsets of that POS containing w_1. We allow this rewrite if PMI$(w_1, w_2) > \lambda$ (=-2.5 based on a devset). We calculate PMI as $log\frac{(\#w_1 \text{ occurs in synsets of } w_2 + \#w_2 \text{ occurs in synsets of } w_1)}{(\# \text{ of synsets of } w_1 \times \# \text{ of synsets of } w_2)}$. Some words can be both synonyms and antonyms in different situations. For example, thesaurus lists 'bad' as a synonym and an antonym of 'good', likewise for 'cool' and 'hot'. We do not allow antonyms in these rewrites.

Thesaurus synonyms can verify 'offer a vast range of ⇒ provide a wide range of', since offer-provide, and vast-wide are thesaurus synonyms. We use Roget's 21^{st} Century Thesaurus in KGLR implementation.

Negating rules: We reject rules where r_2 explicitly negates r_1 or vice versa. We reject a rule if r_2 is same as r_1 if we drop 'not' from one of them. For example, the rule ⟨be the president of ⇒ be not the president of⟩, will be rejected.

Wordnet Hypernyms: We replace word/phrase w in r_1 by its Wordnet hypernym if it is in r_2. We prove 'be highlight of ⇒ be component of', as Wordnet lists 'component' as a hypernym of 'highlight'.

Dropping Adjectives/Adverbs/Superlatives: We drop any adjective, adverb, superlatives or comparatives (e.g., 'more', 'most') from r_1. This lets us verify 'be most important part of ⇒ be part of'.

Gerund-Infinitive Equivalence: We convert infinitive constructions into gerunds and vice versa. For example, 'starts to drink ≡ starts drinking'.

Deverbal Nouns: We use Wordnet's derivationally related forms to compute a verb-noun pair list. We allow back and forth conversions from "be noun of" to a related verb. So, we verify 'be

[4]http://paraphrase.org/#/download
[5]https://www.thesaurus.com
[6]https://www.thesaurus.com/browse/produce/12

cause of ⇒ cause'.

Light Verbs and Serial Verbs: If a light verb precedes a word with derivationally related noun sense, we delete it. Similarly, if a serial verb precede a word with derivationally related verb sense, we delete it. We identify light verbs via the verbs that frequently precede a '(a|an)(verb|deverbal noun)' pair in Wikipedia. Serial verbs are identified as the verbs that frequently precede another verb in Wikipedia. Thus we can convert 'take a look at ⇒ look at'.

Preposition Synonyms: We manually create a list of preposition near-synonyms such as in-to, in-at, at-near, in-by, as-to be. We replace a preposition by its near-synonym. This proves ⟨translated into ⇒ translated to⟩.

Be-Words & Determiners: We drop be-words ('is', 'was', 'be', etc.) and determiners from r_1 and r_2.

Active-Passive: We allow active form like $(e_s, verb, e_o)$ to be rewritten as its passive form $(e_o, be\ verb\ by, e_s)$.

Redundant Prepositions: We find that often prepositions other than 'by' often can be alternatively used with passive forms of some verbs. Moreover, some prepositions can be redundantly used in active forms too. For example, '(e_s, absorb, e_o) ↔ (e_o, be absorbed *in*, e_s)', or similarly, '(e_s, attack, e_o) ↔ (e_s, attack *on*, e_o)' To create such a list of verb-preposition pairs, we simply trust the argument-overlap statistics. Statistics here do not make many errors since the base verb on both relations is the same.

Implementation: KGLR allows repeated application of these rewrites to modify r_1 and r_2. If it achieves $r_1 = r_2$ it verifies the inference rule. For tractable implementation KGLR uses a depth first search approach where a search node maintains both r_1 and r_2. Search does not allow rewrites that introduce any lexical (lemmatized) entries not in original words(r_1) ∪ words(r_2). If it ca not apply any rewrite to get a new node, it returns failure.

Many implications are proved by a sequence of rewrites. E.g., to prove '(e_s, be a major cause of, e_o) ⇒ (e_o, be caused by, e_s)', the proof proceeds as: (e_s, be a major cause of, e_o) ⇒ (e_s, be major cause of, e_o) ⇒ (e_s, be cause of, e_o) ⇒ (e_s, cause, e_o) ⇒ (e_s, be caused by, e_o) by dropping determiner, dropping adjective, deverbal noun, and active-passive transformation respectively. Similarly, '(e_s, helps to protect, e_o) ⇒ (e_s, look after, e_o)' follows from gerund-infinitive conversion (helps protect), dropping support from serial verbs (protect), and thesaurus synonym (look after).

8.4 Experiments

KGLR verifies a subset of rules from CLEAN and PPDB$_e$ to produce, VCLEAN and VPPDB$_e$.

Through our experiments we seek answers to the following research questions: (1) What is the precision and size of the verified subsets compared to original corpora? (2) How does additional knowledge generated after performing inference using these rules compare with each other? and (3) Which rewrites are critical to KGLR performance?

Comparison of CLEAN and VCLEAN: The original CLEAN corpus has about 102K rules. KGLR verifies about 36K rules and filters 66K rules out. To estimate the precisions of CLEAN and VCLEAN we independently sampled a random subset of 200 inference rules from each. We asked two annotators (graduate-level NLP students) to label the rules as correct or incorrect. Rules were mixed, and the annotators were blind to the system that generated the rule. Our initial annotation guideline was similar to that of textual entailment — label a rule as correct if the consequent can usually be inferred given the antecedent, for most naturally occurring argument-pairs for the antecedent.

Our annotators faced one issue with the guideline: some inference rules were valid if (e_s,e_o) were bound to specific types and not valid for others. For example, '(e_s, be born in, e_o) $\Rightarrow$ (e_s, be birthplace of, e_o)' is valid if e_o is a location, and not a year. Even seemingly-correct inference rules, e.g., '(e_s, is the father of, e_o) $\Rightarrow$ (e_o, is the child of, e_s)', make unusual incorrect inferences: (Gandhi, is the father of, India) does not imply (India, is the child of, Gandhi). Unfortunately, these corpora do not associate argument-type information with their inference rules.

To mitigate this, we refined the annotation guidelines to accept inference rules as long as they are valid for *some* type-pair. The inter-annotator agreement with this modification was 94% (κ = 0.88). On the subset of the tags where the two annotators agreed, we find the precision of CLEAN to be 48.9%, whereas VCLEAN was 82.5% precise – much more useful for real-world applications. Multiplying the precision with their sizes, we find the effective yield[7] of CLEAN to be 50K compared to 30K for VCLEAN. Overall, we find that VCLEAN obtains a 34 point precision improvement with an effective fact yield of about 60%. We summarize all performance details in Table 8.2.

Error Analysis: Most of VCLEAN errors are due to erroneous (or unusual) thesaurus synonyms, example ('cast', 'drop'), ('talk', 'sing') are synonymous in thesaurus. For missed recall, we analyzed the rules in CLEAN but missed by VCLEAN. We find that only about 13% of those are world knowledge rules (e.g., rule 6 in Figure 8.1). Other missed recall is because of some missing rewrites, missing thesaurus synonyms ('let go of', 'free' are not synonymous in thesaurus) and spelling mistakes (the rule 'dont know about' $\Longrightarrow$ '*don* know' can't be verified), which can potentially be captured by using other resources and adding rewrite rules.

Comparison of PPDB$_e$ and VPPDB$_e$: Unlike CLEAN, the PPDB2.0 dataset associates a confidence value for each rule, which can be adjusted to obtain different levels of precision and

[7] Yield is proportional to recall

System	**CLEAN**	**VCLEAN**
Size	102,565	36,229
Rule Precision	48.9%	82.5%
Rule Yield	50,154	29,889
Fact Precision	49.1%	81.6%
Fact Yield	7 million	4.5 million

System	**PPDB$_e$(0.342)**	**VPPDB$_e$**
Size	85,272	85,261
Rule Precision	44.2%	71.4%
Fact Precision	22.16%	51.30%
Fact Yield	41 million	35 million

Figure 8.2: The precision and yield of inference rules after KGLR validation, and that of KB generated by inference using these rule-sets. Comparison with PPDB$_e$ is yield-controlled.

yield. We control for yield so that we can compare precisions directly.

We operate on the PPDB$_e$ subset that has OpenIE-like relation phrase on both sides; this was identified by matching to ReVerb syntactic patterns [Etzioni et al., 2011]. This subset is of size 402K. KGLR on this produces 85K verified rules (VPPDB$_e$). We find the threshold for confidence values in PPDB$_e$ that achieves the same yield (confidence > 0.342).

We perform annotation on PPDB$_e$(0.342) and VPPDB$_e$ using the same annotation guidelines as before. The inter-annotator agreement was 91% (κ = 0.82). On the subset of the tags where the two annotators agreed, we find the precision of PPDB$_e$ to be low: 44.2%, whereas VPPDB$_e$ was evaluated to be 71.4% precise. We notice that about 80% PPDB relation phrases (example, 'assist in', 'permit' and 'produce') are of length 1 or 2 (whereas 50% of CLEAN relation phrases are of length ≥ 3). This contributes to slightly lower precision of VPPDB$_e$, as most rules are proved by thesaurus synonymy and the power of KGLR to handle compositionality of longer relation phrases does not get exploited.

Comparison of Inferred Facts: New facts can be inferred by applying inference rules to a KB. We independently apply VCLEAN's and CLEAN's inference rules on a public corpus of 4.2 million ReVerb triples.[8] Since ReVerb itself has significant extraction errors (our estimate is 20%) and our goal is to evaluate the quality of inference, we restrict this evaluation to only the subset of accurate (high confidence) ReVerb extractions.

VCLEAN *and* CLEAN *facts:* We sampled about 200 facts inferred by VCLEAN rules and CLEAN rules (applied over accurate ReVerb extractions). We gave the original sentence and the inferred facts to the two annotators. We obtained a high inter-annotator agreement of 96.3% (κ = 0.92), and we discarded disagreements from the final analysis. Overall, facts inferred by CLEAN achieved a precision of about 49.1%, and those inferred by VCLEAN obtained an

[8] http://reverb.cs.washington.edu

System	Precision	Recall
KGLR (all rules)	85.4%	62.0%
w/o Negating Rules	85.4%	62.0%
w/o Antonyms	84.2%	62.0%
w/o Wordnet Hypernyms	86.1%	59.3%
w/o Dropping Modifiers	84.9%	59.6%
w/o Gerund-Infinitive Equivalence	85.2%	61.0%
w/o Light and Serial Verbs	85.0%	59.9%
w/o Deverbal Nouns	85.4%	62.0%
w/o Preposition Synonyms	86.9%	56.9%
w/o Active-Passive	85.0%	54.5%
w/o Redundant Prepositions	86.1%	61.6%

Figure 8.3: Ablation study of rule verification using KGLR rewrites on our devset of 600 CLEAN rules

81.6% precision. The estimated yields of fact corpora (precision×size) are 7 and 4.5 million for CLEAN and VCLEAN respectively. This yield estimate does not include the initial 4.2 million facts.

PPDB$_e$ *and* VPPDB$_e$ *facts:* As done previously, we sampled 200 facts inferred by PPDB$_e$ and VPPDB$_e$ rules, which two annotators annotated. We obtained a good inter-annotator agreement of 90.0%($\kappa = 0.8$), and we discarded disagreements from the final analysis. Overall, facts inferred by PPDB$_e$ achieved an inferior precision 22.2% and those inferred by VPPDB$_e$ obtained an improvement of about 29% (51.3% precision). Short relation phrases (mostly of length 1 or 2, which forms 80% of PPDB$_e$) contribute to low precision of VPPDB$_e$. Example low precision VPPDB$_e$ rules include $\langle$ (X, be, Y) $\Rightarrow$ (X, obtain, Y)$\rangle$, $\langle$ (X, include, Y) $\Rightarrow$ (X, come, Y)$\rangle$, which were inaccurately verified due to thesaurus errors. The estimated yields of fact corpora are 41 million and 35 million for PPDB$_e$ and VPPDB$_e$ respectively.

Ablation Study of KGLR rewrites: We evaluate the efficacy of different rewrites in KGLR by performing an ablation study (see Table 8.3). We ran KGLR by turning off one rewrite rule on a sample of 600 CLEAN rules (our development set) and calculating its precision and recall. The ablation study highlights that most rewrites add some value to the performance of KGLR, however *antonyms* and *dropping modifiers* are particularly important for precision, and *active-passive* and *redundant preposition* add a substantial recall.

The ablation study highlights that most rewrites add some value to the performance of KGLR, however *antonyms* and *dropping modifiers* are particularly important for precision and *active-passive* and *redundant preposition* add substantial recall.

Discussion: KGLR's value is in precision-sensitive tasks such as a human-facing demo or downstream NLP application (like question answering) where error propagation is highly undesirable. Along with high precision, KGLR still obtains acceptably good yield.

Our annotators observe the importance of type-restriction of arguments for inference rules (similar to rules in Schoenmackers et al. [2010]). Type annotation of existing inference rule corpora is important for obtaining high precision fact predictions. We also made a similar observation in the macro inference setting. In Chapter 5, a type-aware inference model shows improved KBC performance. Hence, building typed inference rule corpora is a promising future direction.

Generally, inference rules are of two types — linguistic/synonym rewrites (discussed in this work) and world knowledge rules (see rule 6 in Fig 8.1), which are not discussed here. We were surprised to estimate that about 87% of CLEAN rule base (statistically-generated) is just linguistic rewrites! Obtaining world knowledge or common-sense rules at high precision and scale, continues to be the key NLP challenge in this area.

8.5 Conclusions

We present KGLR, a linguistic rewrite system for inference rule verification, it exploits the compositionality of relation phrases, guided by existing knowledge sources (Thesaurus and Wordnet), to identify a high precision subset of existing inference rule base – CLEAN and PPDB$_e$. The validated CLEAN has a high precision of 82% (vs. 49%) at a yield of 59%. Validated PPDB$_e$ has a precision of 71% (vs. 44%) at the same yield. The precision of inferred facts also has a 32 % precision gain. We expect KGLR to be effective for precision-sensitive applications of inference.

Note that KGLR is a rule verification algorithm for rule bases generated from various algorithms discussed in background chapter (see Section 2.3.2).

Relevant Publication

- [Jain and Mausam, 2016]: "Knowledge Guided Linguistic Rewrites for Inference Rule Verification". Prachi Jain, Mausam. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016.

Part IV

Epilogue

Chapter 9

Conclusions and Future Directions

Knowledge bases are background assets for various applications, including – visual reasoning, reading comprehension, anomaly detection, information search, and retrieval. These knowledge bases are incomplete. Much research focused on building models that add missing facts in such KBs. Over time we also identified various ways to improve KBC performance by augmenting KBC models with different gadgets — type, time, language-oriented.

The underlying principles of the research work discussed in this thesis focus on (1) performing qualitative analysis to inspire model design; (2) adding appropriate modeling priors based on the task at hand; (3) verifying that the model meaningfully incorporates the priors; (4) carefully designing the evaluation criterion.

In chapter 4, an extensive qualitative analysis of Matrix Factorization (MF) model helps us identify that under-treatment of OOV-entity pairs is the reason behind its poor performance. We developed a series of extensions to mitigate the effect of OOVs in MF. Our most successful model uses ComplEx (Tensor Factorization – TF model) to augment our improved version of MF via a regularized additive loss. Note that a basic MF added to TF isn't robust – only an "OOV trained" MF when integrated with TF attains good performance. Next in chapter 5, we observe that KBC models often make entity predictions that are incompatible with the type required by the relation. In chapter 8, a careful analysis revealed that inference rules corpus (which are often generated using distributional similarity like argument-pair overlap) have a very low precision, not enough to be useful for many real tasks.

KBC models are customized based on the task at hand. In chapter 5, to incorporate type information into the base factorization model, we enhance it with two type-compatibility terms between entity relation pairs. In chapter 7, we build models for temporal knowledge bases in which entities, relations, and time are all embedded in a uniform, compatible space. Our model TIMEPLEX also exploits the recurrent nature of some facts/events and temporal interactions between pairs of relations. In chapter 8, we discuss KBC methods for OpenIE KBs, where entities and relations are represented via textual strings. We exploit the compositionality of the text and linguistic insights to improve the precision of the rule corpus and improve Open-KBC performance.

In the various models proposed in this thesis, we assess whether the embeddings produced by them meaningfully capture the modelling prior. In Chapter 5, we confirm that our type models capture type information better, we correlate the embeddings learned without type supervision with existing type catalogs. We find that our embeddings indeed separate and predict types better. In Chapter 7, our diagnostics suggest that the learnt time embeddings are temporally meaningful, and TIMEPLEX makes fewer temporal consistency and ordering mistakes.

This thesis also emphasizes carefully designing evaluation criteria for a fair estimation of model performance. In chapter 4, we propose a new evaluation protocol that makes comparisons between MF and TF models fair. The new properly designed evaluation protocol helped us identify that MF model performance is overestimated. In chapter 7 we found that existing TKBC models heavily overestimate link prediction performance due to imperfect evaluation mechanisms. In response, we propose improved TKBC evaluation protocols for both link and time prediction tasks, dealing with subtle issues that arise from the partial overlap of time intervals in gold instances and system predictions.

While the KBC models were developed, the quality of the underlying GPU architectures also improved, allowing us to study large models. Chapter 6 identified model training, particularly negative sampling, as crucial to model performance. We train large dimensional models by contrasting positive examples with all entities (very large number of entities) to improve the underlying model's performance. In particular COMPLEX-V2 (with base-model CX) obtain best or near best scores.

All the KBC models proposed in this thesis are rigorously evaluated on a variety of baseline datasets. Our TIMEPLEX model (discussed in Chapter 7) still has state-of-the-art link prediction performance on time-interval datasets – WIKIDATA12k and YAGO11k. We release all the datasets and code used in this thesis for further research [1].

Note that the best model (hybrid MF-TF) of Chapter 4 is not carried forward as the baseline model for the next work. The model performance reported in Chapter 5 are comparable to model performances reported in Chapter 6. We also trained the best model from Chapter 5 – TypeComplex with larger (all) set of negative examples and the performance of the model improved but it was still not competitive with the underlying base model ComplEx (when trained with larger number of negative examples) – COMPLEX-V2. Also, various methods from different chapters are implemented with different hyperparameters, eg. embedding dimensions or number of negative samples. Also the models are implemented in different frameworks – Chap-

[1] https://github.com/dair-iitd/

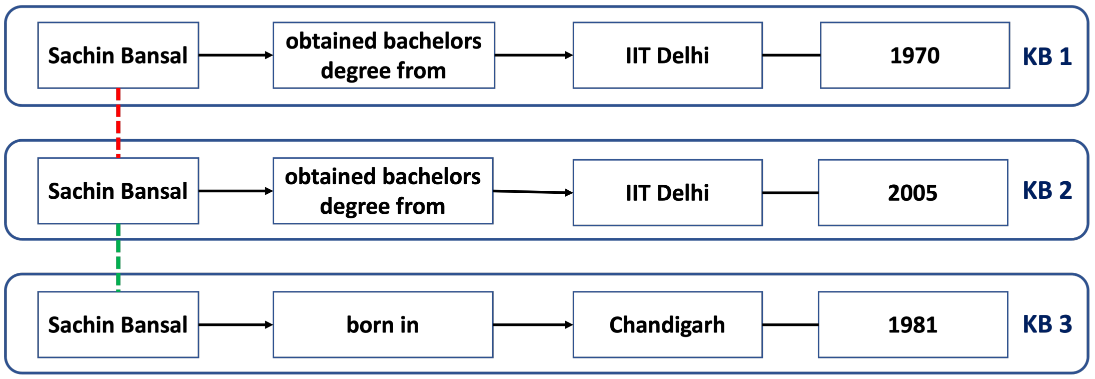


Figure 9.1: This figure demonstrates the benefits of additional temporal information for entity alignment and the multi-KBC task using dummy data. Following inferences would have been difficult if we ignored time information. (1) *Sachin Bansal* of KB1 and *Sachin Bansal* of KB2 are different entities as the two graduated at different times. (2) *Sachin Bansal* of KB1 and *Sachin Bansal* of KB3 cannot be the same as a person cannot graduate before he is born. (3) *Sachin Bansal* of KB2 and *Sachin Bansal* of KB3 are likely to be the same as his graduation age is 24 years which is close to the average age of graduation (as computed in Section 7.4.2).

ter 4 models are implemented in keras (theano), and the rest are implemented in pytorch. This results in the performance of a model different across different chapters, even on one fixed dataset. Both the issues arise because the methods were developed at different points in time, and the state of the art (or best practices) also evolved contemporaneously, leading to baselines that were moving targets themselves.

In addition, there are multiple research directions emanating from this thesis and contemporary research which can be pursued in the future:

- **Multi-Lingual Knowledge Base Completion:** Knowledge Bases are now expanding to multiple languages. Most KBC research has been focused on mono-lingual single KBs. However, building a KBC model which can leverage unique information available across various KBs of different languages is the need of the hour. Entity alignment (EA) and Relation alignment (RA) are important subtasks for multilingual KBC. On training a joint model for the three tasks – KBC, EA and RA, we obtained improved performance on all three tasks on DBP5L dataset [Singh et al., 2021]. DBP5L benchmark Chen et al. [2020] is derived from DBPedia in five languages: English (En), Greek (El), Spanish (Es), Japanese (Ja) and French (Fr). We need more datasets that have KBs in multiple languages. Besides the need for building more datasets for doing this research, we also identify the following directions to work on:

  Time information: is available in various KBs. This attribute can be used to improve inference and entity alignment across multiple KBs (which can be multilingual or monolingual). For examples see Figure 9.1, the facts (*Sachin Bansal, obtained bachelors degree* *from, IIT Delhi, in 1970*) and (*Sachin Bansal, obtained bachelors degree from, IIT Delhi, in 2005*) from *KB 1* and *KB 2* can be misinterpreted to be talking about the same *'Sachin Bansal'*, if we ignore the time information. Another fact *(Sachin Bansal, born in, Chandigarh, 1981)* from *KB 3* is more likely to be related to *Sachin Bansal* of *KB 2* and than of *KB 1*, as *KB 1*'s *Sachin Bansal* graduated before *KB 3*'s *Sachin Bansal* was born (also see *relation ordering* property discussed in Section 7.4.2). Also if *Sachin Bansal* of *KB 2* and *KB 3* are same, than *Sachin Bansal*'s age of graduation is 24 years, which is close to average age of graduation computed from data in Section 7.4.2 (time gaps between relations can be modeled). More ideas from Chapter 7 can be borrowed to build a time-aware multi-lingual and multi-KB completion model.

Type information: is very valuable for improving entity alignment and KBC performance. For example, facts from two different KBs — *(Holy basil, is used in, medicine)* and *(Thai food, is served at, Holy Basil)*, mentions two different entities with the same name *Holy basil*. Both entities have a different type; one is a plant, and the other is a restaurant. Also, a new fact *(Holy basil, is used in, Thai food)* is more related to the first fact we mentioned, as they talk about the same entity *Holy basil* the plant. Modeling such entity-type information across multiple KBs in an unsupervised manner is an exciting direction to pursue.

- **Multi-Modal Knowledge Base Completion:** Knowledge Bases can be heterogeneous, that is, they may comprise information in different modalities, including - text, numerical values, and images. Building methods that can encode various modalities and infer new facts (in various modalities) in KBs is an exciting direction to pursue. We discussed how to handle text and numeric attributes, but this thesis does not explore image attributes.

  Images naturally have a substantial type signal. For example, the Wikipedia page of all countries has the national flag of the respective country as the head image. Hence the image of different countries looks similar. Same way, the image of entities of type people look similar, but they are very different from the pictures of water bodies (see figure 9.2). Such a strong type signal may further improve KBC model performance.

  Recent image-transformers (like BEiT Bao et al. [2021]) can be used to obtain image embeddings. Distance between the image and entity type embeddings can be added to the loss function of the type model (see equation 5.2) for learning improved type embedding and hence improving KBC performance.

- **Typing for Open Knowledge Base Completion:** Open KBs are generally large. Its entities and relations are not canonicalized, leading to the storage of redundant and ambiguous facts. Open KBs also suffer from incompleteness. Building KBC models for Open

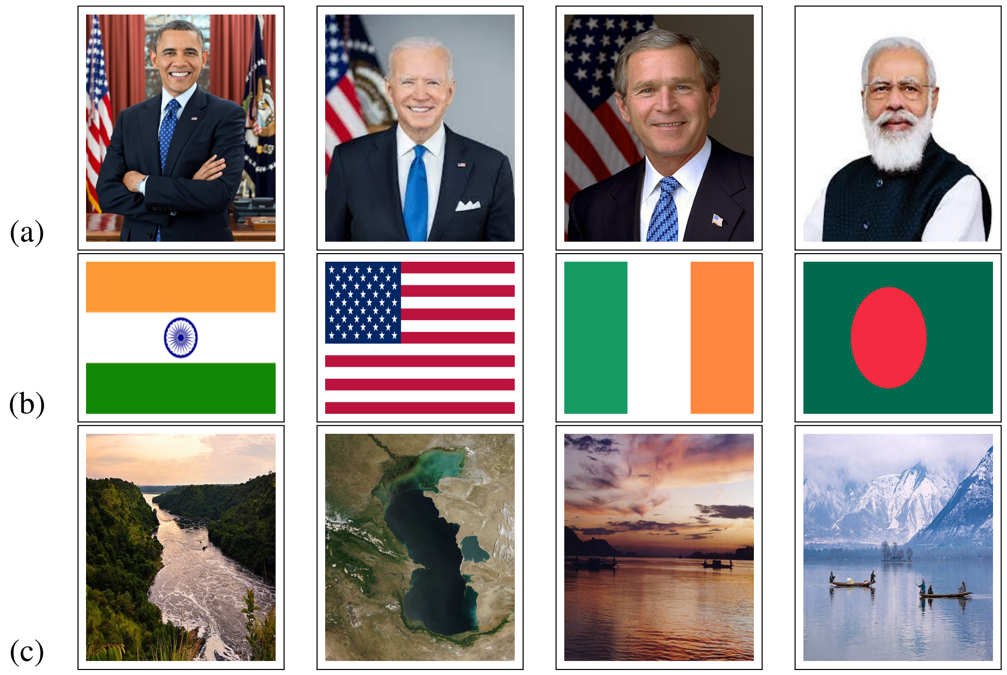


Figure 9.2: Images of entities of similar type look alike. Also, they look different from pictures of other entity types. In this figure, we show Wikipedia title images of entities of type (a) person (b) countries (c) water bodies.

KBs is an under-explored area. We can build off our previous work on typed models (discussed in chapter 5) to infer new type compatible facts in OpenKBs.

- **TKBC methods for OpenIE KBs:** A fraction of OpenIE KBs also has temporal knowledge of facts. Porting TKBC ideas to this new dataset is a promising future direction.

Bibliography

Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. Towards a human-like open-domain chatbot. *CoRR*, abs/2001.09977, 2020. URL https://arxiv.org/abs/2001.09977.

Gabor Angeli and Christopher D Manning. Naturalli: Natural logic inference for common sense reasoning. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2007. doi: 10.1007/978-3-540-76298-0_52. URL https://doi.org/10.1007/978-3-540-76298-0_52.

Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *Empirical Methods in Natural Language Processing*, 2019.

Colin Bannard and Chris Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604. Association for Computational Linguistics, 2005.

Hangbo Bao, Li Dong, and Furu Wei. Beit: BERT pre-training of image transformers. *CoRR*, abs/2106.08254, 2021. URL https://arxiv.org/abs/2106.08254.

Hannah Bast, Björn Buchhold, and Elmar Haussmann. Semantic search on text and knowledge bases. *Found. Trends Inf. Retr.*, 10(2-3):119–271, 2016. doi: 10.1561/1500000032. URL https://doi.org/10.1561/1500000032.

Jonathan Berant. *Global Learning of Textual Entailment Graphs*. PhD thesis, Tel Aviv University, 2012.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. Global learning of typed entailment rules. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 610–619. Association for Computational Linguistics, 2011.

Jonathan Berant, Ido Dagan, Meni Adler, and Jacob Goldberger. Efficient tree-based approximation for entailment graph learning. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the System Demonstrations, July 10, 2012, Jeju Island, Korea*, 2012.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*, pages 1247–1250, 2008. URL http://ids.snu.ac.kr/w/images/9/98/sc17.pdf.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *NIPS*, pages 2787–2795, 2013. URL http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.

Samuel Broscheit, Kiril Gashteovski, Yanjie Wang, and Rainer Gemulla. Can we predict new facts with open knowledge graph embeddings? A benchmark for open link prediction. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2296–2308. Association for Computational Linguistics, 2020. URL https://www.aclweb.org/anthology/2020.acl-main.209/.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6203.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial*

Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010. AAAI Press, 2010. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1879.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. Variational knowledge graph reasoning. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1823–1832. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1165. URL https://doi.org/10.18653/v1/n18-1165.

Xuelu Chen, Muhao Chen, Changjun Fan, Ankith Uppunda, Yizhou Sun, and Carlo Zaniolo. Multilingual knowledge graph completion via ensemble knowledge transfer. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3227–3238. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.290. URL https://doi.org/10.18653/v1/2020.findings-emnlp.290.

Timothy Chklovski and Patrick Pantel. Verbocean: Mining the web for fine-grained semantic verb relations. In *EMNLP*, volume 2004, pages 33–40, 2004.

François Chollet. Keras. https://github.com/fchollet/keras, 2015.

Peter Clark, Phil Harrison, and John Thompson. A knowledge-driven approach to text meaning processing. In *Proceedings of the HLT-NAACL 2003 workshop on Text meaning*, pages 1–6, 2003.

Peter Clark, Niranjan Balasubramanian, Sumithra Bhakthavatsalam, Kevin Humphreys, Jesse Kinkead, Ashish Sabharwal, and Oyvind Tafjord. Automatic construction of inference-supporting knowledge bases. In *4th Workshop on Automated Knowledge Base Construction (AKBC)*, 2014.

Nilesh N. Dalvi, Ravi Kumar, Bo Pang, Raghu Ramakrishnan, Andrew Tomkins, Philip Bohannon, S. Sathiya Keerthi, and Srujana Merugu. A web of concepts. In Jan Paredaens and Jianwen Su, editors, *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, pages 1–12. ACM, 2009. doi: 10.1145/1559795.1559797. URL https://doi.org/10.1145/1559795.1559797.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. *CoRR*, abs/1607.01426, 2016. URL http://arxiv.org/abs/1607.01426.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Syg-YfWCW.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha P. Talukdar. HyTE: Hyperplane-based temporally aware knowledge graph embedding. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2001–2011. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1225. URL https://doi.org/10.18653/v1/d18-1225.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1389–1399. The Association for Computational Linguistics, 2016. doi: 10.18653/v1/d16-1146. URL https://doi.org/10.18653/v1/d16-1146.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1811–1818. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17366.

Pedro M. Domingos and William Austin Webb. A tractable first-order probabilistic logic. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 601–610. ACM, 2014. doi: 10.1145/2623330.2623623. URL https://doi.org/10.1145/2623330.2623623.

O Etzioni, M Cafarella, et al. Web-scale information extraction in KnowItAll. In *WWW Conference*, New York, 2004. ACM. URL http://www.cs.washington.edu/research/knowitall/papers/www-paper.pdf.

Oren Etzioni. Search needs a shake-up. *Nature*, 476(7358):25–26, 2011.

Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1):91–134, 2005. doi: 10.1016/j.artint.2005.03.001. URL https://doi.org/10.1016/j.artint.2005.03.001.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10, 2011.

Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP Conference*, pages 1535–1545, 2011. URL http://ml.cs.washington.edu/www/media/papers/reverb_emnlp2011.pdf.

Miao Fan, Qiang Zhou, and Thomas Fang Zheng. Distant supervision for entity linking. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation, PACLIC 29, Shanghai, China, October 30 - November 1, 2015*. ACL, 2015. URL https://aclanthology.org/Y15-1010/.

Cong Fu, Tong Chen, Meng Qu, Woojeong Jin, and Xiang Ren. Collaborative policy learning for open knowledge graph reasoning. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2672–2681. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1269. URL https://doi.org/10.18653/v1/D19-1269.

Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M Suchanek. Canonicalizing open knowledge bases. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1679–1688. ACM, 2014.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422. International World Wide Web Conferences Steering Committee, 2013.

Alberto García-Durán and Mathias Niepert. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. In Amir Globerson and Ricardo Silva, editors, *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 372–381. AUAI Press, 2018. URL http://auai.org/uai2018/proceedings/papers/149.pdf.

Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. Composing Relationships with Translations. In *EMNLP*, pages 286–290, 2015.

Alberto Garcia-Duran, Antoine Bordes, Nicolas Usunier, and Yves Grandvalet. Combining two and three-way embeddings models for link prediction in knowledge bases. *arXiv preprint arXiv:1506.00999*, 2015. URL https://arxiv.org/pdf/1506.00999.

Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4816–4821. Association for Computational Linguistics, 2018. URL https://www.aclweb.org/anthology/D18-1516/.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom M. Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 397–406. ACL, 2014. doi: 10.3115/v1/d14-1044. URL https://doi.org/10.3115/v1/d14-1044.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1):83–135, 2006.

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. Diachronic embedding for temporal knowledge graph completion. In *AAAI*, 2020. URL https://arxiv.org/pdf/1907.03143.pdf.

Adam Grycner, Gerhard Weikum, Jay Pujara, James Foulds, and Lise Getoor. Relly: Inferring hypernym relationships between relational phrases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 971–981, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1113.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems. *CoRR*, abs/2003.00911, 2020. URL https://arxiv.org/abs/2003.00911.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 192–202. The Association for Computational Linguistics, 2016. doi: 10.18653/v1/d16-1019. URL https://doi.org/10.18653/v1/d16-1019.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Knowledge graph embedding with iterative guidance from soft rules. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4816–4823. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16369.

Kelvin Guu, John Miller, and Percy Liang. Traversing Knowledge Graphs in Vector Space. In *EMNLP*, pages 318–327, 2015.

Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

Graeme Hirst and David St Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 305–332. MIT Press, 1998.

Prachi Jain and Mausam. Knowledge-guided linguistic rewrites for inference rule verification. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 86–92.

The Association for Computational Linguistics, 2016. doi: 10.18653/v1/n16-1011. URL https://doi.org/10.18653/v1/n16-1011.

Prachi Jain, Pankaj Kumar, Mausam, and Soumen Chakrabarti. Type-sensitive knowledge base inference without explicit type supervision. In *Association for Computational Linguistics (ACL)*, 2018a.

Prachi Jain, Shikhar Murty, Mausam, and Soumen Chakrabarti. Mitigating the effect of out-of-vocabulary entity pairs in matrix factorization for KB inference. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4122–4129. ijcai.org, 2018b. doi: 10.24963/ijcai.2018/573. URL https://doi.org/10.24963/ijcai.2018/573.

Prachi Jain, Sushant Rathi, Soumen Chakrabarti, et al. Knowledge base completion: Baseline strikes back (again). *arXiv preprint arXiv:2005.00804*, 2020a.

Prachi Jain, Sushant Rathi, Mausam, and Soumen Chakrabarti. Temporal knowledge base completion: New algorithms and evaluation protocols. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 3733–3747. Association for Computational Linguistics, 2020b. doi: 10.18653/v1/2020.emnlp-main.305. URL https://doi.org/10.18653/v1/2020.emnlp-main.305.

Heng Ji, Ralph Grishman, Hoa Trang Dang, and Kira Griffitt. Overview of the TAC2011 knowledge base population (KBP) track. In *Text Analysis Conference (TAC)*, pages 14–15, November 2011. URL https://tac.nist.gov/publications/2010/presentations/TAC2010_KBP_Overview.pdf.

Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. Towards time-aware knowledge graph completion. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1715–1724. ACL, 2016. URL https://www.aclweb.org/anthology/C16-1161/.

Woojeong Jin, Changlin Zhang, Pedro A. Szekely, and Xiang Ren. Recurrent event network for reasoning over temporal knowledge graphs. *CoRR*, abs/1904.05530, 2019. URL http://arxiv.org/abs/1904.05530.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *SIGKDD Conference*, pages 133–142. ACM, 2002. URL http://www.cs.cornell.edu/People/tj/publications/joachims_02c.pdf.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines strike back. In Phil Blunsom, Antoine Bordes, Kyunghyun Cho, Shay B. Cohen, Chris Dyer, Edward Grefenstette, Karl Moritz Hermann, Laura Rimell, Jason Weston, and Scott Yih, editors, *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pages 69–74. Association for Computational Linguistics, 2017. doi: 10.18653/v1/w17-2609. URL https://doi.org/10.18653/v1/w17-2609.

Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 4289–4300, 2018. URL http://papers.nips.cc/paper/7682-simple-embedding-for-link-prediction-in-knowledge-graphs.

Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263. URL https://doi.org/10.1109/MC.2009.263.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Int. J. Comput. Vis.*, 123(1):32–73, 2017. doi: 10.1007/s11263-016-0981-7. URL https://doi.org/10.1007/s11263-016-0981-7.

Denis Krompaß, Stephan Baier, and Volker Tresp. Type-constrained representation learning in knowledge graphs. In Marcelo Arenas, Óscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Steffen Staab, editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, volume 9366 of *Lecture Notes in Computer Science*, pages 640–655. Springer, 2015. doi: 10.1007/978-3-319-25007-6_37. URL https://doi.org/10.1007/978-3-319-25007-6_37.

Aapo Kyrola. Drunkardmob: billions of random walks on just a PC. In Qiang Yang, Irwin King, Qing Li, Pearl Pu, and George Karypis, editors, *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 257–264. ACM, 2013. doi: 10.1145/2507157.2507173. URL https://doi.org/10.1145/2507157.2507173.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2869–2878. PMLR, 2018. URL http://proceedings.mlr.press/v80/lacroix18a.html.

Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. Tensor decompositions for temporal knowledge base completion. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rke2P1BFwS.

Ni Lao and William W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67, October 2010. ISSN 0885-6125. doi: 10.1007/s10994-010-5205-8. URL http://dx.doi.org/10.1007/s10994-010-5205-8.

Ni Lao, Tom Mitchell, and William W Cohen. Random Walk Inference and Learning in A Large Scale Knowledge Base. In *EMNLP*, pages 529–539. Association for Computational Linguistics, 2011.

Douglas B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):32–38, 1995. doi: 10.1145/219717.219745. URL https://doi.org/10.1145/219717.219745.

Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. Smoothing the geometry of probabilistic box embeddings. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=H1xSNiRcF7.

Dekang Lin and Patrick Pantel. Dirt@ sbt@ discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328. ACM, 2001.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3243–3253. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1362. URL https://doi.org/10.18653/v1/d18-1362.

Xiao Ling and Daniel S. Weld. Fine-grained entity recognition. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press, 2012. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5152.

Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. Evaluating question answering over linked data. *Web Semantics Science Services And Agents On The World Wide Web*, 21:3–13, 2013. ISSN 1570-8268. doi: 10.1016/j.websem.2013.05.006.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2001.

Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1506–1515. Association for Computational Linguistics, 2017. doi: 10.18653/v1/d17-1159. URL https://doi.org/10.18653/v1/d17-1159.

Mausam. Open information extraction systems and downstream applications. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4074–4077. IJCAI/AAAI Press, 2016. URL http://www.ijcai.org/Abstract/16/604.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics, 2012.

Donald Metzler and W. Bruce Croft. Latent concept expansion using markov random fields. In Wessel Kraaij, Arjen P. de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando, editors, *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July*

23-27, 2007, pages 311–318. ACM, 2007. doi: 10.1145/1277741.1277796. URL https://doi.org/10.1145/1277741.1277796.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL Conference*, pages 777–782, 2013. URL http://www.anthology.aclweb.org/N/N13/N13-1095.pdf.

Pasquale Minervini, Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Leveraging the schema in latent factor models for knowledge graph completion. In Sascha Ossowski, editor, *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*, pages 327–332. ACM, 2016. doi: 10.1145/2851613.2851841. URL https://doi.org/10.1145/2851613.2851841.

Pasquale Minervini, Luca Costabello, Emir Muñoz, Vít Novácek, and Pierre-Yves Vandenbussche. Regularizing knowledge graph embeddings via equivalence and inversion axioms. In Michelangelo Ceci, Jaakko Hollmén, Ljupco Todorovski, Celine Vens, and Saso Dzeroski, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18-22, 2017, Proceedings, Part I*, volume 10534 of *Lecture Notes in Computer Science*, pages 668–683. Springer, 2017a. doi: 10.1007/978-3-319-71249-9_40. URL https://doi.org/10.1007/978-3-319-71249-9_40.

Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Adversarial sets for regularising neural link predictors. In Gal Elidan, Kristian Kersting, and Alexander T. Ihler, editors, *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017b. URL http://auai.org/uai2017/proceedings/papers/306.pdf.

Pasquale Minervini, Matko Bosnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. Differentiable reasoning on large knowledge bases and natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5182–5190. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/5962.

Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. Learning reasoning strategies in end-to-end differentiable proving. In Pascal Hitzler and Md. Kamruzzaman Sarker, editors, *Neuro-Symbolic Artificial Intelligence: The*

State of the Art, volume 342 of *Frontiers in Artificial Intelligence and Applications*, pages 280–293. IOS Press, 2021. doi: 10.3233/FAIA210359. URL https://doi.org/10.3233/FAIA210359.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In Keh-Yih Su, Jian Su, and Janyce Wiebe, editors, *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 1003–1011. The Association for Computer Linguistics, 2009. URL https://aclanthology.org/P09-1113/.

Shikhar Murty, Patrik Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. Hierarchical losses and new resources for fine-grained entity typing and linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2018.

Ndapandula Nakashole and Tom M. Mitchell. A knowledge-intensive model for prepositional phrase attachment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 365–375. The Association for Computer Linguistics, 2015. doi: 10.3115/v1/p15-1036. URL https://doi.org/10.3115/v1/p15-1036.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145. Association for Computational Linguistics, 2012.

Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250, 2012. doi: 10.1016/j.artint.2012.07.001. URL https://doi.org/10.1016/j.artint.2012.07.001.

Arvind Neelakantan and Ming-Wei Chang. Inferring missing entity type instances for knowledge base completion: New dataset and methods. *In NAACL*, 2015.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. Compositional vector space models for knowledge base completion. In *AAAI Conference*, 2015. URL http://www.aaai.org/ocs/index.php/SSS/SSS15/paper/download/10254/10032.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A Three-Way Model for Collective Learning on Multi-Relational Data. In *ICML*, pages 809–816, 2011.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. *arXiv preprint arXiv:1503.00759*, 2015.

Feng Niu, Christopher Ré, AnHai Doan, and Jude W. Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an RDBMS. *PVLDB*, 4(6):373–384, 2011.

Ellie Pavlick, Johan Bos, Malvina Nissim, Charley Beller, Benjamin Van Durme, and Chris Callison-Burch. Adding semantics to data-driven paraphrasing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, 2015.

Simone Paolo Ponzetto and Michael Strube. Deriving a large-scale taxonomy from wikipedia. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1440–1445. AAAI Press, 2007. URL http://www.aaai.org/Library/AAAI/2007/aaai07-228.php.

Jay Pujara, Eriq Augustine, and Lise Getoor. Sparsity and noise: Where knowledge graph embeddings fall short. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1751–1756. Association for Computational Linguistics, 2017. doi: 10.18653/v1/d17-1184. URL https://doi.org/10.18653/v1/d17-1184.

Richard Qian. Understand your world with bing. *Microsoft Bing Blog, March*, 2013.

Meng Qu and Jian Tang. Probabilistic logic neural networks for reasoning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7710–7720, 2019. URL http://papers.nips.cc/paper/8987-probabilistic-logic-neural-networks-for-reasoning.

Sindhu Raghavan, Raymond J. Mooney, and Hyeonseo Ku. Learning to "read between the lines" using bayesian logic programs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL-2012)*, pages 349–358, July 2012. URL http://www.cs.utexas.edu/users/ai-lab/?raghavan:acl2012.

Altaf Rahman and Vincent Ng. Coreference resolution with world knowledge. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 814–824. The Association for Computer Linguistics, 2011. URL https://aclanthology.org/P11-1082/.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics, 2016. doi: 10.18653/v1/d16-1264. URL https://doi.org/10.18653/v1/d16-1264.

Lev-Arie Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In Suzanne Stevenson and Xavier Carreras, editors, *Proceedings of the Thirteenth Conference on Computational Natural Language Learning, CoNLL 2009, Boulder, Colorado, USA, June 4-5, 2009*, pages 147–155. ACL, 2009. URL https://aclanthology.org/W09-1119/.

Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 658–666. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00075. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Rezatofighi_Generalized_Intersection_Over_Union_A_Metric_and_a_Loss_for_CVPR_2019_paper.html.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 74–84, 2013.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3788–3800, 2017. URL http://papers.nips.cc/paper/6969-end-to-end-differentiable-proving.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting Logical Background Knowledge into Embeddings for Relation Extraction. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2015.

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP Conference*, 2007. URL http://aclweb.org/anthology/D/D07/D07-1043.pdf.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You CAN teach an old dog new tricks! on training knowledge graph embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BkxSmlBFvr.

Evan Sandhaus. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752, 2008.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer, 2018. doi: 10.1007/978-3-319-93417-4_38. URL https://doi.org/10.1007/978-3-319-93417-4_38.

Stefan Schoenmackers, Oren Etzioni, and Daniel S Weld. Scaling textual inference to the web. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–88. Association for Computational Linguistics, 2008.

Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, and Jesse Davis. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098. AssociaFrition for Computational Linguistics, 2010.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 -*

February 1, 2019, pages 3060–3067. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33013060. URL https://doi.org/10.1609/aaai.v33i01.33013060.

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. M-walk: Learning to walk over graphs using monte carlo tree search. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6787–6798, 2018. URL http://papers.nips.cc/paper/7912-m-walk-learning-to-walk-over-graphs-using-monte-carlo-tree-searc

Harkanwar Singh, Soumen Chakrabarti, Prachi Jain, Sharod Roy Choudhury, and Mausam. Multilingual knowledge graph completion with joint relation and entity alignment. In Danqi Chen, Jonathan Berant, Andrew McCallum, and Sameer Singh, editors, *3rd Conference on Automated Knowledge Base Construction, AKBC 2021, Virtual, October 4-8, 2021*, 2021. URL https://www.akbc.ws/2021/papers/FTzU__68rp8.

Sameer Singh, Tim Rocktäschel, and Sebastian Riedel. Towards combined matrix and tensor factorization for universal schema relation extraction. In *Workshop on Vector Space Modeling for Natural Language Processing*, pages 135–142, 2015. URL http://www.aclweb.org/anthology/W15-1519.

Amit Singhal. Introducing the knowledge graph: things, not strings. *Official Google Blog, May*, 2012.

R. Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 3679–3686. European Language Resources Association (ELRA), 2012. URL http://www.lrec-conf.org/proceedings/lrec2012/summaries/1072.html.

Sandeep Subramanian and Soumen Chakrabarti. New embedded representations and evaluation protocols for inferring transitive relations. In Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz, editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1037–1040. ACM, 2018. doi: 10.1145/3209978.3210150. URL https://doi.org/10.1145/3209978.3210150.

Fabian M. Suchanek and Nicoleta Preda. Semantic culturomics (vision paper). *Proc. VLDB Endow.*, 7(12):1215–1218, 2014. doi: 10.14778/2732977.2732994. URL http://www.vldb.org/pvldb/vol7/p1215-suchanek.pdf.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019a. URL https://openreview.net/forum?id=HkgEQnRqYQ.

Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha P. Talukdar, and Yiming Yang. A re-evaluation of knowledge graph completion methods. *CoRR*, abs/1911.03903, 2019b. URL http://arxiv.org/abs/1911.03903.

Mihai Surdeanu. Overview of the TAC2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *Text Analysis Conference (TAC)*, 2013. URL http://www.surdeanu.info/mihai/papers/kbp2013.pdf.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. *ACL Association for Computational Linguistics*, 2015.

Kristina Toutanova, Xi Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. Compositional Learning of Embeddings for Relation Paths in Knowledge Bases and Text. In *ACL*, pages 1434–1444, 2016.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3462–3471. PMLR, 2017. URL http://proceedings.mlr.press/v70/trivedi17a.html.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080, 2016. URL http://arxiv.org/abs/1606.06357.

Christina Unger, André Freitas, and Philipp Cimiano. An introduction to question answering over linked data. In Manolis Koubarakis, Giorgos B. Stamou, Giorgos Stoilos, Ian Horrocks, Phokion G. Kolaitis, Georg Lausen, and Gerhard Weikum, editors, *Reasoning Web. Reasoning on the Web in the Big Data Era - 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings*, volume 8714 of *Lecture Notes in Computer Science*, pages 100–140. Springer, 2014. doi: 10.1007/978-3-319-10587-1_2. URL https://doi.org/10.1007/978-3-319-10587-1_2.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. Composition-based multi-relational graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BylA_C4tPr.

Patrick Verga, Arvind Neelakantan, and Andrew McCallum. Generalizing to Unseen Entities and Entity Pairs with Row-less Universal Schema. *CoRR*, abs/1606.05804, 2016.

Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. Probabilistic embedding of knowledge graphs with box lattice measures. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 263–272. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1025. URL https://aclanthology.org/P18-1025/.

Denny Vrandecic and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014. doi: 10.1145/2629489. URL https://doi.org/10.1145/2629489.

Hila Weisman, Jonathan Berant, Idan Szpektor, and Ido Dagan. Learning verb inference rules from linguistically-motivated evidence. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012*, pages 194–204, 2012.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *TACL*, 6:287–302, 2018. URL https://transacl.org/ojs/index.php/tacl/article/view/1325.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In Chin-Wan Chung, Andrei Z. Broder, Kyuseok Shim, and Torsten Suel, editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, pages 515–526.

ACM, 2014. doi: 10.1145/2566486.2568032. URL https://doi.org/10.1145/2566486.2568032.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. Probase: a probabilistic taxonomy for text understanding. In K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 481–492. ACM, 2012. doi: 10.1145/2213836.2213891. URL https://doi.org/10.1145/2213836.2213891.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. An interpretable knowledge transfer model for knowledge base completion. *arXiv preprint arXiv:1704.05908*, 2017. URL https://arxiv.org/pdf/1704.05908.pdf.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*, pages 2965–2971, 2016.

Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 564–573. Association for Computational Linguistics, 2017. doi: 10.18653/v1/d17-1060. URL https://doi.org/10.18653/v1/d17-1060.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. Noise mitigation for neural entity typing and relation extraction. In Mirella Lapata, Phil Blunsom, and Alexander Koller, editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 1183–1194. Association for Computational Linguistics, 2017. doi: 10.18653/v1/e17-1111. URL https://doi.org/10.18653/v1/e17-1111.

Bishan Yang and Tom M. Mitchell. Leveraging knowledge bases in lstms for improving machine reading. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1436–1446. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1132. URL https://doi.org/10.18653/v1/P17-1132.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the Inter-*

national Conference on Learning Representations (ICLR) 2015, May 2015. URL http://research.microsoft.com/apps/pubs/default.aspx?id=241703.

Fan Yang, Zhilin Yang, and William W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2319–2328, 2017. URL http://papers.nips.cc/paper/6826-differentiable-learning-of-logical-rules-for-knowledge-base-reas

Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. A vectorized relational graph convolutional network for multi-relational network alignment. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4135–4141. ijcai.org, 2019. doi: 10.24963/ijcai.2019/574. URL https://doi.org/10.24963/ijcai.2019/574.

Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. Iteratively learning embeddings and rules for knowledge graph reasoning. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2366–2377. ACM, 2019. doi: 10.1145/3308558.3313612. URL https://doi.org/10.1145/3308558.3313612.

List of Publications

This thesis is based on the following publications:

1. "Knowledge Guided Linguistic Rewrites for Inference Rule Verification". Prachi Jain, Mausam. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). San Diego, California. June 2016.

2. "Mitigating the Effect of Out-of-Vocabulary Entity Pairs in Matrix Factorization for KB Inference". Prachi Jain, Shikhar Murty, Mausam, Soumen Chakrabarti. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI). Stockholm, Sweden. July 2018.

3. "Type-Sensitive Knowledge Base Inference Without Explicit Type Supervision". Prachi Jain, Pankaj Kumar, Mausam, Soumen Chakrabarti. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL). Melbourne, Australia. July 2018.

4. "Temporal Knowledge Base Completion: New Algorithms and Evaluation Protocols". Prachi Jain, Sushant Rathi, Mausam, Soumen Chakrabarti. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2020.

5. "Knowledge Base Completion: Baseline strikes back (Again)". Prachi Jain, Sushant Rathi, Mausam, Soumen Chakrabarti. Under submission.

Biography

Prachi Jain pursued Ph.D at the Department of Computer Science and Engineering, Indian Institute of Technology Delhi from 2014-2020. She obtained a Master's degree in Computer Science with a specialization in Information Security from Indraprastha Institute of Information Technology (IIIT), Delhi in 2013. She has a Bachelors' degree in Computer Science & Engineering from BMIET, Maharshi Dayanand University, Rohtak. Her research interests include Knowledge base completion techniques, and Natural Language Processing.