EMBEDDING BASED LINK PREDICTION FOR KNOWLEDGE GRAPH COMPLETION

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften

(Dr.-Ing.)

von der KIT-Fakultät für Wirtschaftswissenschaften

des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION
von

M.Sc. RUSSA BISWAS

Tag der mündlichen Prüfung: November 24, 2022

Referent: Prof. Dr. Harald Sack
Korreferent: Prof. Dr. Paul Groth

This thesis is dedicated to my parents
Kalpana and Pradyot Biswas

ABSTRACT

Knowledge Graphs (KGs) are the most widely used representation of structured information about a particular domain consisting of billions of facts in the form of entities (nodes) and relations (edges) between them. Besides, the KGs also encapsulate the semantic type information of the entities. The last two decades have witnessed a constant growth of KGs in various domains such as government, scholarly data, biomedical domains, etc. KGs have been used in Machine Learning based applications such as entity linking, question answering, recommender systems, etc. Open KGs are mostly heuristically created, automatically generated from heterogeneous resources such as text, images, etc., or are human-curated. However, these KGs are often incomplete, i.e., there are missing links between the entities and missing links between the entities and their corresponding entity types. This thesis focuses on addressing these two challenges of link prediction for Knowledge Graph Completion (KGC): **(i)** General Link Prediction in KGs that include head and tail prediction, triple classification, and **(ii)** Entity Type Prediction.

Most of the graph mining algorithms are proven to be of high complexity, deterring their usage in KG-based applications. In recent years, KG embeddings have been trained to represent the entities and relations in the KG in a low-dimensional vector space preserving the graph structure. In most published works such as the translational models, convolutional models, semantic matching, etc., the triple information is used to generate the latent representation of the entities and relations.

In this dissertation, it is argued that contextual information about the entities obtained from the random walks, and textual entity descriptions, are the keys to improving the latent representation of the entities for KGC. The experimental results show that the knowledge obtained from the context of the entities supports the hypothesis. Several methods have been proposed for KGC and their effectiveness is shown empirically in this thesis. Firstly, a novel multi-hop attentive KG embedding model MADLINK is proposed for Link Prediction. It considers the contextual information of the entities by using random walks as well as textual entity descriptions of the entities. Secondly, a novel architecture exploiting the information contained in a pre-trained contextual Neural Language Model (NLM) is proposed for Triple Classification. Thirdly, the limitations of the current state-of-the-art (SoTA) entity type prediction models have been analysed and a novel entity typing model CAT2Type is proposed that exploits the Wikipedia Categories which is one of the most under-treated features of the KGs. This model can also be used to predict missing types of unseen entities i.e., the newly added entities in the KG. Finally, another novel architecture GRAND is proposed to predict the missing entity types in KGs using multi-label, multi-class, and hierarchical classification by leveraging different strategic graph walks in the KGs. The extensive experiments and ablation studies show that all the proposed models outperform the current SoTA models and set new baselines for KGC.

The proposed models establish that the NLMs and the contextual information of the entities in the KGs together with the different neural network architectures benefit KGC. The promising results and observations open up interesting scopes for future research involving exploiting the proposed models in domain-specific KGs such as scholarly data, biomedical data, etc. Furthermore, the link prediction model can be exploited as a base model for the entity alignment task as it considers the neighbourhood information of the entities.

ZUSAMMENFASSUNG

Wissensgraphen (Knowledge Graphs, KGs) sind die am weitesten verbreitete Darstellung strukturierter Informationen über einen bestimmten Bereich, der aus Milliarden von Fakten in Form von Entitäten (Knoten) und Beziehungen (Kanten) zwischen ihnen besteht. Daneben beinhalten die KGs auch semantische Informationen der Entitäten. In den letzten zwei Jahrzehnten hat die Zahl der KGs in verschiedenen Bereichen wie Verwaltungsdaten, wissenschaftlichen Daten oder biomedizinischen Daten ständig zugenommen. KGs werden in auf maschinellem Lernen basierenden Anwendungen wie Entity Linking, Question Answering Systemen oder in Empfehlungssystemen verwendet. Offene KGs werden meistens heuristisch erstellt, automatisch aus heterogenen Ressourcen wie Text, Bildern usw. generiert oder von Menschenhand kuratiert. Diese KGs sind jedoch oft unvollständig, d.h. es fehlen Verknüpfungen zwischen den Entitäten sowie Verknüpfungen zwischen den Entitäten und ihren zugehörigen Typeninformationen. In der vorliegenden Arbeit liegt der Fokus darauf, diese beiden Herausforderungen bei der Vervollständigung von Wissensgraphen (Knowledge Graph Completion, KGC) zu bewältigen: **(i)**Link-Vorhersage (Link Prediction) in KGs, die die Vorhersage von Head und Tail und die Tripel-Klassifikation einschließt sowie die **(ii)** Vorhersage des Entitätstyps.

In den letzten Jahren wurden Knowledge Graph Embeddings trainiert, um die Entitäten und ihre Beziehungen untereinander in einem niedrig-dimensionalen Vektorraum darzustellen, der die Graphenstruktur bewahrt. Im Großteil der veröffentlichten Arbeiten, deren Fokus z. B. auf Übersetzungsmodellen, Faltungsmodellen oder semantischem Matching liegt, wird die RDF-Tripel-Information verwendet, um daraus eine latente Repräsentation der Entitäten und Relationen zu erzeugen.

In dieser Dissertation wird argumentiert, dass kontextuelle Informationen über die Entitäten eines KGs, die aus Random Walks und textuellen Entitätsbeschreibungen gewonnen werden, der Schlüssel zur Verbesserung der latenten Repräsentation für die Vervollständigung von Wissensgraphen (KGC) sind. Die experimentellen Ergebnisse belegen, dass das aus dem Entitätskontext gewonnene Wissen diese Hypothese stützt. Mehrere Methoden zur Knowledge Graph Completion werden vorgeschlagen und ihre Wirksamkeit wird empirisch nachgewiesen. Als erstes wird ein neuartiges Multi-Hop Attentive KG Embedding Modell, MADLINK, für die Link-Vorhersage vorgeschlagen. Es berücksichtigt kontextuelle Entitätsinformationen aus Random Walks und textuellen Entitätsbeschreibungen. In einem zweiten Schritt wird für die Tripel-Klassifikation eine neuartige Architektur vorgeschlagen, die die in einem vortrainierten neuronalen Sprachmodell (Neural Language Model, NLM) enthaltenen Informationen ausnutzt. Abschließend werden die Grenzen der aktuellen State-of-the-Art-Modelle (SoTA) zur Vorhersage von Entitätstypen analysiert und ein neuartiges Modell zur Typisierung von Entitäten, CAT2Type, vorgeschlagen, das die Wikipedia-Kategorien ausnutzt, die bislang nur wenig Beachtung fanden. Das vorgeschlagene Modell kann auch dazu verwendet werden, fehlende Typinformation von bislang unbekannten Entitäten vorherzusagen. Dies ist insbesondere relevant für neu in den KG hinzugefügte Entitäten. Schließlich wird eine weitere neuartige Architektur, GRAND, vorgeschlagen, um fehlende Typeninformation in KGs unter Verwendung von Multilabel-, Multiklassen- und hierarchischer Klassifikation vorherzusagen. Dabei werden verschiedene strategische Graphwalks in den KGs genutzt. Die umfangreichen Experimente und Ablationsstudien zeigen, dass alle vorgeschlagenen Modelle die aktuellen SoTA-Modelle übertreffen und neue Baselines für KGC setzen.

Die vorgeschlagenen Ansätze belegen, dass der Einsatz von NLMs in Verbindung mit kontextueller Information über die Entitäten im KG zusammen mit den verschiedenen neuronalen Netzwerkarchitekturen von entscheidendem Vorteil für die KGC ist. Die vielversprechenden Ergebnisse und Beobachtungen eröffnen interessante Möglichkeiten für die zukünftige Forschung, die die Nutzung der vorgeschlagenen Modelle in domänenspezifischen KGs vorsieht. Darüber hinaus kann das Link-Vorhersagemodell als Basismodell für das Alignment von Entitäten (Entity Alignment) genutzt werden, da es die Nachbarschaftsinformationen der Entitäten berücksichtigt.

PUBLICATIONS

The thesis is based on the following publications

[1] Russa Biswas. "Embedding based Link Prediction for Knowledge Graph Completion." In: *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management*, ACM, 2020, pp. 3221–3224.

[2] Russa Biswas, Mehwish Alam, and Harald Sack. "MADLINK: Attentive Multihop and Entity Descriptions for Link Prediction in Knowledge Graphs." In: *Semantic Web* (2022).

[3] Russa Biswas, Yiyi Chen, Heiko Paulheim, Harald Sack, and Mehwish Alam. "It's All in the Name: Entity Typing Using Multilingual Language Models." In: *The Semantic Web: ESWC 2022 Satellite Events Proceedings*. Vol. 13384. Lecture Notes in Computer Science. Springer, 2022, pp. 36–41.

[4] Russa Biswas, Jan Portisch, Heiko Paulheim, Harald Sack, and Mehwish Alam. "Entity Type Prediction Leveraging Graph Walks and Entity Descriptions." In: *Accepted at ISWC*. 2022.

[5] Russa Biswas, Radina Sofronova, Mehwish Alam, Nicolas Heist, Heiko Paulheim, and Harald Sack. "Do Judge an Entity by Its Name! Entity Typing Using Language Models." In: *The Semantic Web: ESWC 2021 Satellite Events Proceedings*. Vol. 12739. Lecture Notes in Computer Science. Springer, 2021, pp. 65–70.

[6] Russa Biswas, Radina Sofronova, Mehwish Alam, and Harald Sack. "Contextual Language Models for Knowledge Graph Completion." In: *Machine Learning with Symbolic Methods and Knowledge Graphs co-located with ECML PKDD*. CEUR Workshop Proceedings. 2021.

[7] Russa Biswas, Radina Sofronova, Harald Sack, and Mehwish Alam. "Cat2type: Wikipedia Category Embeddings for Entity Typing in Knowledge Graphs." In: *Proceedings of the 11th on Knowledge Capture Conference*. 2021, pp. 81–88.

The author has further contributed to the following publications

[1] Mehwish Alam, Russa Biswas, Yiyi Chen, Danilo Dessì, Genet Asefa Gesese, Fabian Hoppe, and Harald Sack. "HierClasSArt: Knowledge-Aware Hierarchical Classification of Scholarly Articles." In: *Companion of The Web Conference 2021*. ACM / IW3C2, 2021, pp. 436–440.

[2] Russa Biswas, Maria Koutraki, and Harald Sack. "Exploiting Equivalence to Infer Type Subsumption in Linked Graphs." In: *The Semantic Web: ESWC 2018 Satellite Events*. Vol. 11155. Lecture Notes in Computer Science. Springer, 2018, pp. 72–76.

[3] Russa Biswas, Maria Koutraki, and Harald Sack. "Predicting Wikipedia Infobox Type Information using Word Embeddings on Categories." In: *Proceedings of the EKAW 2018 Posters and Demonstrations Session co-located with 21st International Conference on Knowledge Engineering and Knowledge Management*. 2018.

[4] Russa Biswas, Radina Sofronova, Mehwish Alam, and Harald Sack. "Entity Type Prediction in Knowledge Graphs using Embeddings." In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2020) co-located with the 17th Extended Semantic Web Conference*. Vol. 2635. CEUR Workshop Proceedings. 2020.

[5] Russa Biswas, Rima Türker, Farshad Bakhshandegan Moghaddam, Maria Koutraki, and Harald Sack. "Wikipedia Infobox Type Prediction Using Embeddings." In: *Proceedings of the First Workshop on Deep Learning for Knowledge Graphs and Semantic Technologies (DL4KGS) co-located with the 15th Extended Semantic Web Conerence*. 2018.

[6] Genet Asefa Gesese, Russa Biswas, Mehwish Alam, and Harald Sack. "A survey on knowledge graph embeddings with literals: Which model links better literally?" In: *Semantic Web* 12.4 (2021), pp. 617–647.

[7] Genet Asefa Gesese, Russa Biswas, and Harald Sack. "A Comprehensive Survey of Knowledge Graph Embeddings with Literals: Techniques and Applications." In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference*. Vol. 2377. CEUR Workshop Proceedings, pp. 31–40.

[8] Sven Müller, Michael Brunzel, Daniela Kaun, Russa Biswas, Maria Koutraki, Tabea Tietz, and Harald Sack. "HistorEx: Exploring Historical Text Corpora Using Word and Document Embeddings." In: *The Semantic Web: ESWC 2019 Satellite Events - ESWC 2019 Satellite Events, Portorož, Slovenia, June 2-6, 2019, Revised Selected Papers*. Vol. 11762. Lecture Notes in Computer Science. Springer, 2019, pp. 136–140.

[9] Radina Sofronova, Russa Biswas, Mehwish Alam, and Harald Sack. "Entity Typing Based on RDF2Vec Using Supervised and Unsupervised Methods." In: *The Semantic Web: ESWC 2020 Satellite Events*. Vol. 12124. Lecture Notes in Computer Science. Springer, 2020, pp. 203–207.

ACKNOWLEDGMENTS

First and foremost, I'd like to thank Prof. Dr Harald Sack for giving me this opportunity to pursue my PhD under his supervision. Over the years, his persistent support, encouragement, guidance and the inspiring discussions we have had, have assisted me in exploring the research challenges in the right direction. I am thankful to Prof. Dr Paul Groth for reviewing my thesis as the second reviewer and providing me with valuable insights to improve the work. Many thanks also go to Dr Mehwish Alam who has been the guiding force. I found a mentor in her, who pointed me in the proper way while also boosting my confidence by recognizing my research efforts. I also consider myself fortunate to be able to collaborate a few times with Prof. Dr Heiko Paulheim during the course of my PhD. My research has always benefited greatly from the thought-provoking conversations we have had. Many thanks also to Dr Maria Koutraki, whose advice and assistance was invaluable throughout the first year of my PhD.

A heartfelt thanks to all my colleagues for making the past 5 years an unforgettable experience. It was a pleasure and I learned a lot from working with you! A special mention goes to Vivien, Genet, Yiyi, and Radina for being so supportive. Every discussion we have had has given me a different perspective to look at things, both professionally and personally. I am eternally thankful to my friends Ankush, Savina, Saptarshi, Debjit, Supratim and Saheli for making Germany a home away from home. I would not have made it through all these years without you. Thank you for being my biggest cheerleaders always.

Last but not the least, my constants, my parents - Kalpana and Pradyot Biswas, my cousin Abhiigyaan, and my family. Thank you for your unconditional love, and unwavering support, for bearing with me when I am far away for so many years, and for having my back always.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

KG Knowledge Graph

KGC Knowledge Graph Completion

KGE Knowledge Graph Embeddings

NLP Natural Language Processing

SOTA State-of-the-art

ML Machine Learning

RELU Rectified Linear Unit

FNN Feed-forward Neural Networks

MLP Multi-Layer Perceptron

FCNN Fully Connected Neural Network

CNN Convolutional Neural Networks

GRU Gated Recurrent Unit

LSTM Long Short Term Memory

LM Language Model

NLM Neural Language Model

SLM Statistical Language Model

CBOW Continuous Bag of Words

OOV Out-of-Vocabulary

BERT Bidirectional Encoder Representations from Transformers

SBERT Sentence BERT

GPT-2 Generative Pre-trained Transformer

MRR Mean Reciprocal Rank

FB Freebase

WN WordNet

DB DBpedia

PF-ITF Predicate Frequency Inverse Triple Frequency

Part I

MOTIVATION

1 INTRODUCTION

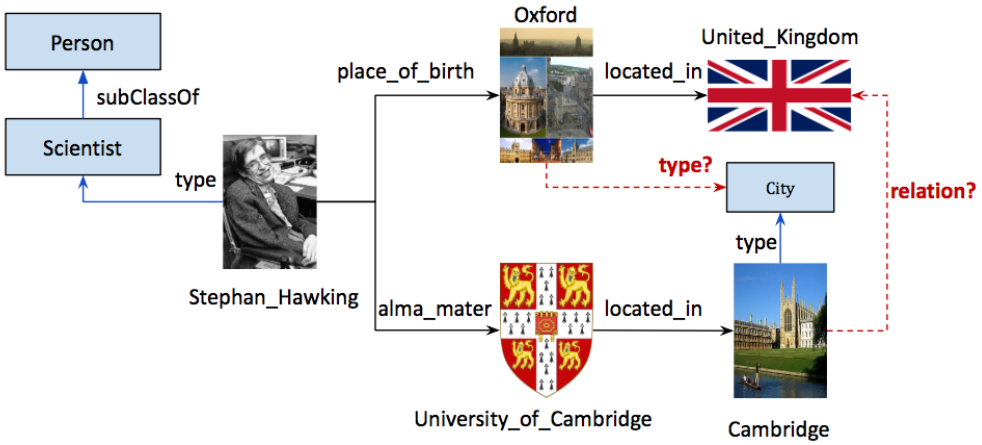
The term *"Knowledge Graph"* (KG) was first coined in literature in 1972 [120]; it was later reintroduced by Google in 2012 with the launch of Google Knowledge Graph. Since then it has emerged as one of the primary forces accelerating the progress in the field of Artificial Intelligence. YAGO [129], Freebase [17], Wikidata [140], DBpedia [4], etc. are some of the most eminent general purpose open KGs which form the backbone of various Machine Learning based applications such as named entity linking [53], question-answering [18], relation extraction [146], etc. Additionally, KGs are extensively used in a number of industrial sectors, such as e-commerce [70, 101], media [111], and life sciences [21], to name a few.

KGs are large networks consisting of huge amounts of facts organised as entities, represented as nodes and relations given by directed labelled edges connecting the nodes. The facts in a KG are presented in the form of a triple $< e_h, r, e_t >$, where e_h and e_t are the head and tail entities respectively and r is the relation between them. Figure 1.1 illustrates a KG, that encodes real-world information. For instance, the entities *Stephan_Hawking, Oxford*, etc. are represented as nodes and their relations are represented as directed edges, e.g., *place_of_birth*. <*Stephan_Hawking, place_of_birth, Oxford*> is a valid triple contained in the KG. Apart from the entities and the relations, the KGs also comprise a special kind of relation that denotes the semantic types of the entities. Semantic types of entities also referred to as classes, are used in KGs to group similar entities together. In Figure 1.1, the semantic type is given by the relation *type* (edges are marked in blue), hence the triple <*Stephan_Hawking, type, Scientist*> represents that *Stephan_Hawking* is an instance of class *Scientist*. The semantic types of the entities are represented in form of a hierarchy in a KG and are denoted by the *subClassOf* relation. For e.g., Figure 1.1 depicts that *Person* is a parent class of *Scientist*.

1.1 MOTIVATION

The cross-domain open KG such as DBpedia, Freebase, and Wikidata are either extracted automatically or semi-automatically from structured data, generated using heuristics, or are human-curated. Despite the huge amount of information encoded in the KGs, it is often observed that they are far from complete. For instance, in Freebase, 71% of 3 million person entities are without their place of birth information, 75% do not have a nationality [145], whereas in DBpedia 2016-10 version, 43% of person entities have their place of birth missing, 46% of the books lack their corresponding author information, director information is not available for 27% of the films, etc. Therefore,

Figure 1.1: Example of a KG consisting of real-world entities



there are many missing facts, and relationships between entities that have not been fully uncovered resulting in incomplete KGs. Furthermore, DBpedia version-2016-10 consists of 48 subclasses of *dbo:Person* and only 36.6% of the total number of entities belonging to dbo:Person is assigned to its subclasses. Also, 307,164 entities in the same DBpedia version are assigned to *owl:Thing*, which is the most generalised class (root node) in the type hierarchy.

As a result, for particular KG-based applications such as question-answering systems, incomplete KGs might not offer the right response to a correctly interpreted question. Given the KG in Figure 1.1, it would not be possible to answer the question, *"Where is Cambridge located?"*, even though both the entity *Cambridge* and the relation *located_in* are included in the KG. Therefore, there arises a necessity to predict the missing tail entity of the triple <*Cambridge, located_in, ?*> or to identify if <*Cambridge, located_in, United_Kingdom*> is a correct triple for the KG. Additionally, it would be impossible to provide a response to the question, *"Is Oxford a city?"*. Here also, the entity *Oxford* and the class *City* are there in the KG, but the information that *Oxford* is an instance of the class *City* is missing. Knowledge Graph Completion (KGC) aims to tackle the aforementioned challenges by addressing the issues of incompleteness and sparsity, hence improving the structure of the KG.

LINK PREDICTION FOR KNOWLEDGE GRAPH COMPLETION Due to the fact that the majority of KGs are created manually, automatically or semi-automatically, many implicit entities and relations have not been recognized, causing incompleteness a prominent challenge in almost all the KGs. As mentioned earlier, KGs store information in form of triples, hence the KGC problem can be looked upon as a problem of estimating missing parts of the triples. Therefore, KGC is achieved by *link prediction* that aims to estimate the probability of the existence of links between entities based

on the current observed information in the KG [163]. Link prediction can be further categorized into three different types of prediction problems depending on the nature of the missing links. The different types are as follows:

- **Head and Tail Prediction**: The head $<?, r, e_t>$ or tail entity in a triple $< e_h, r, ?>$ is predicted by defining a scoring function. For e.g., in reference to the KG shown in Figure 1.1, the prediction of the missing entity in the triple *(Cambridge, located_in, ?)* is denoted as the tail prediction as the head entity and relation information are provided. On the other hand, *(?, located_in, United_Kingdom)* is considered as the head prediction since the relation and tail entity are given. This is referred to as *General Link Prediction* in [54].

- **Triple Classification**: A binary classifier is trained to identify whether a given triple is false (0) or true (1). With reference to the triple in the illustration Figure 1.1 as an example, triple classification helps in identifying if *(Cambridge, located_in, United_Kingdom)* is a true triple for the KG. As it deals with common links in a KG, it can be also considered as *General Link Prediction*.

- **Entity Type Prediction**: It deals with predicting the special kind of links i.e., the semantic types of the entities in the KGs. The problem is transformed into a classification problem in order to predict the semantic type of each entity in the KG and is given by $< e, \text{type}, ? >$, where e is the entity.

By incorporating diverse features provided by the KGs, this dissertation addresses each of the aforementioned types of link prediction problems for KGC. To date, several KGC or link prediction models based on rule reasoning or statistical features have been proposed [122, 123]. As mentioned in [22], single-step reasoning methodologies based on rules rely on a large number of precise and accurate rules as well as statistical features. However, obtaining efficient and widely-applied rules and constraints is challenging, which results in a low rate of recall. The benefit of the rule-based reasoning KGC approach is that rules are either automatically created according to semantics or explicitly extracted, giving them high interpretability. The accuracy is high if complete and precise rules are achieved. As pointed out in literature [22], there are certain drawbacks in this approach as well: **(i)** this strategy heavily relies on rules that are created either manually or automatically and it is very challenging to find full coverage. As a result, it is impossible to actually accomplish the desired reasoning accuracy and completeness effect. **(ii)** Secondly, the rule-based reasoning KGC approach is computationally expensive, especially with the growing size of KGs and the traditional methods are unable to satisfy the demands of the application at hand. As a result, KG embeddings-based models started to evolve for KGC via link prediction [22].

In order to address the aforementioned challenges, KG embedding is proposed that transforms the entities and relations in a KG to a low-dimensional vector space while

preserving its underlying semantics. The entities which are similar to each other appear closer in the vector space. The last decade has witnessed extensive growth in the research of link prediction for KGC using embeddings.

Based on the techniques employed, which are comprehensively discussed in Chapter 3, KG embedding models are then classified into various categories. Most of the KG embedding models use triple information such as the translational model TransE [19] being the oldest one. It models relationships by interpreting them as translations operating on the low-dimensional embeddings of the entities. However, TransE is capable of dealing with only one-to-one relations, which is then overcome by the subsequent models TransH [141], TransR [76], etc which consider one-to-many, many-to-one, and many-to-many relations. Other models that use triples as features are the neural network-based models e.g., ConvE [34] etc., semantic matching models such as RESCAL [91] and its extensions, DistMult [157], etc. However, these models treat each triple separately and independently, hence the graph structure in the KGs is ignored in the modelling. To tackle this, a new set of path-based models namely GAKE [40], PTransE [75], and PConvKB [59] etc. are introduced to surpass this issue. These models only consider the relational paths between the head and tail entities of a triple in order to take into account the graph structure for embedding. The context of each entity in the KG, which is included in the neighbouring nodes, as well as its relationships with other entities, are not carefully considered. Furthermore, some other models such as DKRL [150], and Jointly (ALSTM) [153] exploit the textual entity descriptions together with the triples to generate the KG embeddings. Different static language models (SLMs) are used to generate the text embeddings and the triples are encoded using TransE in DKRL, whereas Jointly (ALSTM) extends DKRL using a gated mechanism and attentive LSTM to encode the text. Due to the usage of SLMs, the representation of the words remains unchanged irrespective of its context, hence the contextual information contained in the textual descriptions remains unexplored. KG-BERT [158] is another text-based embedding model which uses BERT [35] in which triples are considered as input sentences to the BERT model for triple classification. However, this model does not take into consideration the entities' structural information.

Similarly, for entity type prediction, different KG embedding models such as APE [60], HMGCN [61] are proposed in literature which considers the graph structure, anchored text, and Wikipedia categories in the form of adjacency matrices followed by a neural network to jointly learn the representation to a unified space. Next, this entity representation is used to predict their corresponding types in a KG. MuLR [155], and FIGMENT [154] are trained on large annotated corpora using SLMs to generate the entity embedding to predict the missing types. A comprehensive discussion of the existing models is given in Chapter 4. However, none of these models considers the contextual information of the entities captured in the graph walks of the entities. Also, similar to *Link Prediction* models, the contextual text descriptions are ignored. Furthermore, the information contained in the Wikipedia categories remains largely uncharted.

1.2 RESEARCH OBJECTIVES

This dissertation focuses on building models that exploit the contextual information of the entities in the KG together with the textual entity descriptions to address the three aforementioned link prediction tasks: **(i)** Head and Tail Prediction, **(ii)** Triple Classification, and **(iii)** Entity Type Prediction. In order to develop methods that address KGC, the thesis holds the following hypotheses:

- *Contextual information of the entities is crucial for better representation of the entities in a KG.* The role of contextual information is thoroughly investigated in the *Link Prediction* approach.
- *Textual entity descriptions add additional and pertinent details about the entities in KG embeddings.* The impact of including textual entity descriptions for head and tail prediction as well as for entity type prediction is carefully examined.
- *Neural Language Models (NLMs) play an essential role in identifying true triples in a KG.* A NLM-based triple classification approach is studied to understand the impact of NLMs in KGC.
- *Wikipedia Categories are beneficial features in predicting the missing entity types.* Both implicit and explicit features of under-used features in a KG i.e., Wikipedia Categories are exploited extensively for entity typing.
- *Strategic graph walks encapsulate relevant information for entity type inferencing*. Different strategic graph walk methods are employed to learn a better representation of entities to predict the missing entity types.
- *Entity types can be inferred merely from their names.* Different NLMs are leveraged to predict the missing semantic types from entity names.

This thesis focuses on addressing these two challenges of link prediction for KGC: **(i)** Link Prediction in KGs that include head and tail prediction, triple classification, and **(ii)** Entity Type Prediction.

- **Challenge 1 (C1): Link Prediction in KGs**: Recent research has focused on the relational paths in KGs [40, 75] to encapsulate the structural information of the KGs, while other models incorporate the textual entity descriptions [150, 153] into the latent representation of the entities. However, the impact of the structural contextual information in KGC is an unexplored problem. Additionally, the impact of contextual entity description embeddings in link prediction also remains uncharted. These give rise to the following research questions:
  - **C1-RQ1**: Does the contextual information of entities and relations in a KG help in the task of link prediction?

  - **C1-RQ2**: What is the impact of incorporating textual entity descriptions in a KG for the task of link prediction?

  Contextual NLMs have a significant influence on applications based on Natural Language Processing (NLP). However, KG-BERT [152] proposes that NLMs can also be used to predict the missing links in a KG. This provides the inspiration for the research question:

  - **C1-RQ3**: Can we identify correct triples leveraging contextual NLM?

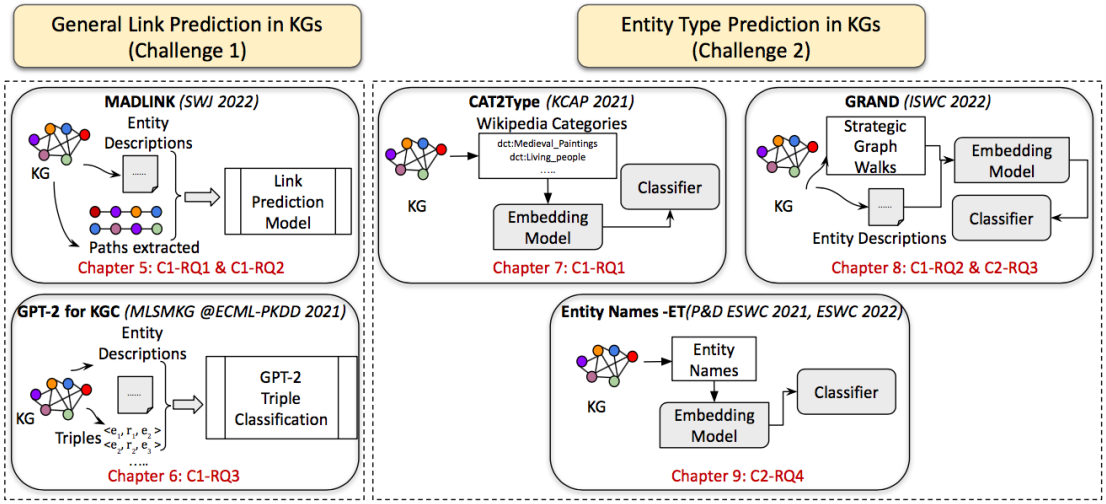
- **Challenge 2 (C2): Entity Type Prediction**: Recent research exploits different features from the KGs such as annotated anchored text, relations between the entities, Wikipedia categories [60, 61, 154], etc. to predict the missing types. However, none of these features is investigated thoroughly, including the utilization of the entire entity descriptions, the interconnections between the Wikipedia categories, random walks on the graphs, entity names, etc. All these open research gaps lead to the following research questions:
  - **C2-RQ1**: Do Wikipedia category labels and the connections between the categories have any impact on entity typing?
  - **C2-RQ2**: What is the impact of textual entity descriptions in predicting the corresponding missing types?
  - **C2-RQ3**: Are strategic graph walks beneficial for entity typing?
  - **C2-RQ4**: Can the types of entities be predicted merely from the entity names?

1.3 THESIS OUTLINE AND CONTRIBUTIONS

The rest of the thesis comprises foundational concepts, the state-of-the-art (SoTA) works concerning general link prediction as well as entity type prediction, followed by the proposed contributions and conclusion. Chapter 2 describes the several fundamental concepts and techniques required to understand the proposed methodologies. A comprehensive literature review is provided in Chapter 3 and Chapter 4 for general link prediction and entity typing respectively, along with a discussion on the shortcomings of the existing models. The above-mentioned challenges and their corresponding research questions are investigated and studied in different chapters of this thesis and an overview of that is provided in Figure 1.2. To this end, the contributions of this dissertation are as follows:

- **Link Prediction in KGs**:
  - In Chapter 5, the research question **C1-RQ1** is addressed in the proposed *MADLINK* model, by generating random walks (paths) from the head and

Figure 1.2: Overview of the Thesis Structure



tail entities in a KG. The random walk captures the contextual information of the entities. The selection of the paths also takes into account the significance of an entity with respect to a relation. For a certain relation, its contextual information is captured by considering all the triples containing that relation. Furthermore, a novel attentive encoder-decoder architecture is proposed to generate the embeddings containing the path information. Next, to address **C1-RQ2**, contextual NLM is leveraged to generate the embedding of the textual description. Finally, the link prediction task, in particular, head and tail prediction is achieved by learning a scoring function on the obtained representations of the structural information and the textual entity descriptions. Additionally, triple classification is also performed to identify correct triples. The model is evaluated on 5 benchmark datasets FB15k [19], FB15k-237 [34, 137], WN18 [19], WN18-RR [34, 137], and YAGO3-10 [34].

  - Chapter 6 answers the research question **C1-RQ3**, where GPT-2 [110] is used to encode the entities and relations into a low-dimensional vector space. The triples and the entity descriptions are separately provided as input sentences to the GPT-2 model followed by a classification layer on the last layer to identify correct triples. The proposed approach is evaluated on WN11, and FB13 datasets.

- **Entity Type Prediction**:
  - In Chapter 7, the different features of the Wikipedia Categories are explored to address the research question **C2-RQ1**. A novel category embed-

ding model *CAT2Type* is proposed, which leverages pre-trained NLMs for learning the representations of entities for entity typing in KGs, and its performance with different NLMs is analyzed. The results provide strong evidence that entity representations based on pre-trained language models exhibit strong generalization and are thus not limited to only NLP tasks. Additionally, a novel category-category network has been constructed to leverage the underlying structure of the categories w.r.t. the shared entities between them. The results strengthen the fact that the category-category network is beneficial to predict types of unseen entities from a different KG. The model has been evaluated on 2 benchmark datasets namely, DBpedia splits from DBpedia630k [162], and FIGER [154].

- Chapter 8 addresses the two research questions **C2-RQ2** and **C2-RQ3**. A novel combined embedding model *GRAND* which leverages different graph walk strategies based RDF2vec models and a contextual NLM for textual entity descriptions is proposed to predict the missing entity types. Additionally, it also provides a generalized classification framework consisting of three different modules namely multi-class, multi-label, and hierarchical classification to predict the missing entity types at different levels of granularity. The proposed model is evaluated on the two aforementioned datasets DBpedia splits from DBpedia630k, and FIGER.
- The Chapter 9 focuses on predicting the missing entity types from their corresponding entity names, addressing the research question **C2-RQ4**. Different NLMs are exploited to gather information about the semantic types from only the entity names in English, as well as, in multiple languages such as German, French, Dutch, and Spanish. Furthermore, we study the effectiveness of this method for long-tailed entities as well as unseen entities in KGs.

Lastly, Chapter 10 summarizes the findings and limitations of this dissertation followed by potential future work.

Part II

BACKGROUND AND LITERATURE REVIEW

2

FOUNDATIONS

This chapter includes a brief introduction to several background topics and notations that will be extensively used throughout this thesis. In what follows, an introduction to graphs is provided in Section 2.1, KGs in Section 2.2, Deep Neural Networks (DNN) in 2.3, Neural Language Models (NLMs) in Section 2.4, Network Embeddings (NE) in Section 2.5, Knowledge Graph Embeddings (KGE) in Section 2.6, and lastly the different evaluation metrics used in this thesis in Section 2.7.

2.1 GRAPHS

Graph theory was originally proposed by Leonhard Euler [39] in 1735 as a solution to the *Seven Bridges of Königsberg Problem*. In the last two decades, with the commencement of large-scale social network platforms, interconnected web-enabled devices, etc. a substantial increase in the research of graph-structured data is witnessed. Graphs are a mathematical representation of a network that is built upon to analyze, understand, and learn from real-world complex systems. A graph is a collection of objects, represented by *nodes or vertices*, along with a set of interactions between pairs of these objects which are depicted by *edges*. For example, to illustrate a social network as a graph, the nodes are the users and an edge exists between two nodes if the two users are friends. The same graph formalization can be used to encode interactions between drugs and proteins, between atoms in a molecule, etc.

Definition 1 (Graph)

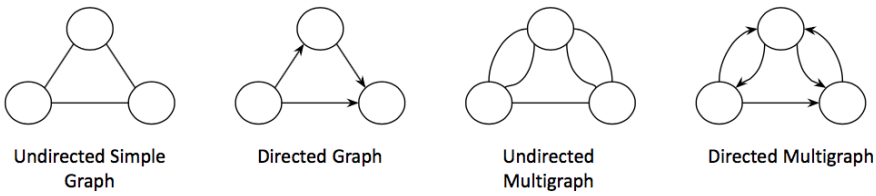
A graph $\mathcal{G}$, is an ordered pair and is given by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges between the nodes.

- *An edge from node $u \in \mathcal{V}$ to node $v \in \mathcal{V}$ is denoted by $(u, v) \in \mathcal{E}$.*
- *If $(u, v) \in \mathcal{E}$ is an edge in $\mathcal{G}$, then u is called adjacent to v.*
- *If $\mathcal{E}_1$ and $\mathcal{E}_2$ are two edges of $\mathcal{G}$, then $\mathcal{E}_1$ and $\mathcal{E}_2$ are called adjacent if $\mathcal{E}_1 \bigcap \mathcal{E}_2 \neq \phi$, i.e., the two edges are incident to the same vertex in $\mathcal{G}$.*

Definition 2 (Graph Variants)

- A ***Undirected Simple Graph*** is an ordered pair defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of at most one undirected edge between each pair of nodes in without any self-loops given by $(u, v) \in \mathcal{E} \leftrightarrow (v, u) \in \mathcal{E}$.

Figure 2.1: Different Graph Variants



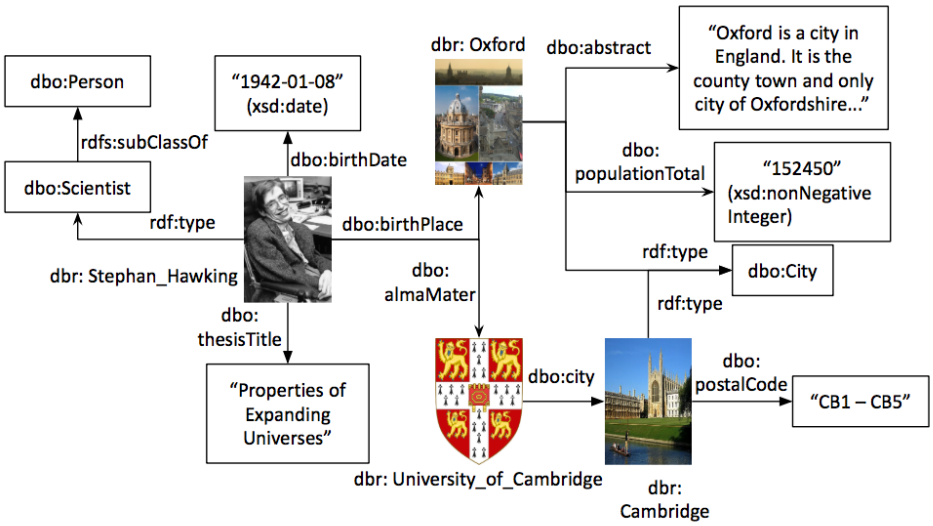
- A ***Directed Graph*** $\mathcal{G}$ is an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} = \{(u, v)|(u, v) \in \mathcal{V}^2\}$. The edges of a directed graph are also called *arcs*.
- An ***Undirected Multigraph*** $\mathcal{G}$ is an ordered triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ and $\mathcal{E}$ are the set of nodes and edges respectively, and $\mathcal{R}$ is an indicator function such that $\mathcal{R} : \mathcal{E} \to \{(u, v)|u, v \in \mathcal{V}\}$, assigning to each edge an unordered pair of endpoint nodes.
- A ***Directed MultiGraph*** $\mathcal{G}$ is an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of directed edges. It consists of multiple arcs i.e., arcs with the same source and target nodes, as well as self-loops.

Therefore, types of graphs vary depending upon connectivity between the nodes and the nature of the edges and the illustrations of aforementioned graphs are provided in Figure 2.1.

2.2 KNOWLEDGE GRAPHS

Semantic networks [124] built in the 1960s are the foundation of current Knowledge Graphs (KGs). The first mention of the term *"Knowledge Graph"* was found in literature in 1972 [120], which was then reincarnated by Google in 2012 during the announcement of *"Google Knowledge Graph"* [125]. As defined in [92], a KG **(i)** mainly describes real-world entities and their interrelations, organized in a graph, **(ii)** defines possible classes and relations of entities in a schema, **(iii)** allows for potentially interrelating arbitrary entities with each other and **(iv)** covers various topical domains. The KGs can be broadly categorized into *open KGs*, and *enterprise KGs* [54]. Open KGs are publicly available and can be accessed online. Some of the most prominent and widely used open KGs are DBpedia [4], Freebase [16], YAGO [131], Wikidata [140], etc. These KGs are either extracted from Wikipedia or are community created. Additionally, open KGs have been created in specific domains, such as media [111], life sciences [21], etc. On the other hand, enterprise KGs such as Google [125], eBay [101], Amazon [70], Uber [49], etc., are confidential to the respective companies and are used for commercial purposes.

Figure 2.2: Example of KG extracted from DBpedia



Definition 3 (Knowledge Graph)

A KG $\mathcal{G}$ is a directed labelled graph consisting of a set of triples $\mathcal{T}$, given by, $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{C})$, where $\mathcal{E}$ is the set of resources referred to as entities, $\mathcal{R}$ is the set of relations (or properties) of the entities, $\mathcal{L}$ is the set of literals, and $\mathcal{C}$ is the set of semantic types or classes of the entities. An entity represents a real-world object or an abstract concept and is identified by a URI. A literal can be text, a date, a number, images, etc.

Definition 4 (Triple)

A triple $< e_h, r, e_t > \in \mathcal{T}$ in a KG $\mathcal{G}$, is an ordered set, where $e_h \in \mathcal{E}$ is the subject, $r \in \mathcal{R}$ is the relation, and $e_t \in \mathcal{E} \cup \mathcal{L} \cup \mathcal{C}$ is the object. The subject is referred to as the head entity and the object when $e_t \in \mathcal{E}$ is referred to as the tail entity. The triples with literals as objects, i.e., $e_t \in \mathcal{L}$ are known as attributive triples. Lastly, the triples with $e_t \in \mathcal{C}$ represent the semantic types of the entities.

Relations (or Properties): Depending on the nature of the objects in a triple, the relations are classified into two main categories:

- Object Relation (or Property), in which an entity is linked to another entity. For instance, in the triple *<dbr: Albert_Einstein, dbo:birthPlace, dbr:Ulm>*, both the subject *dbr: Albert_Einstein*[1] and the object *dbr:Ulm* are entities, the relation *dbo:birthPlace*[2] is an Object Relation (or Property).

[1] prefix dbr: <http://dbpedia.org/resource/> [2] prefix dbo: <http://dbpedia.org/ontology/>

- Data Type Relation (or Property), in which the entity is linked to a literal. For instance, in the triple <*dbr: Albert_Einstein, dbo:birthDate, "1879-03-14"*>, where *"1879-03-14"* is a date and thereby, the relation *dbo:birthDate* is a Data Type Relation (or Property).

Additionally, as mentioned earlier, an entity is also linked to a class or a semantic type of entity using a special kind of relation or property. For example, in DBpedia, *rdf:type* and in Freebase *isA* relation, represent the relation that is used to state that an entity is an instance of a class in the respective KGs. A triple of the form: $< e_i, \text{rdf:type}, \mathcal{C}_k >$, states that $\mathcal{C}_k \in \mathcal{C}$ is a class and $e_i \in \mathcal{E}$ is an entity in $\mathcal{G}$ and is an instance of $\mathcal{C}_k$. The semantic types or the classes in a KG are organised in a hierarchical tree structure. An entity can belong to more than one class in a KG. In this thesis, the words *properties* and *relations* are used interchangeably.

Literals The literals in a KG encode that additional information of the entities which in general are not represented by the entities and their relations. The different types of literals present in a KG are:

- *Text literals*: Different information is stored in a KG in form of free natural language texts such as labels, entity descriptions, comments, titles, etc.
- *Numeric literals*: Date, population, size, and other data stored as integers, floats, and so on also provide important information about an entity in a KG.
- *Image literals*: Images also encode useful information about the entities. For instance, the gender of a person or the shape of an object can be determined by analysing the respective images of the entities.
- *Other Types of literals*: External URIs containing an image, text, audio, videos, etc. linked to the entities also contain beneficial information.

Figure 2.2 illustrates an example of a KG extracted from DBpedia [4] consisting of entities, relations, semantic types of the entities, and literals. Here, *dbr: Stephan_Hawking, dbr: Oxford, dbr: University_of_Cambridge, dbr: Cambridge* are the entities. The entity *dbr: Stephan_Hawking* is of type *dbo: Scientist*, which is a subclass of *dbo: Person* depicting the class hierarchy. The relations *dbo: birthPlace, dbo: almaMater, dbo: city* are object relations as they link two entities. The example also consists of triples with text and numeric literals. The relations in the attributive triples such as *dbo: thesisTitle, dbo: abstract, dbo: populationTotal*, etc. are the data type relations.

The datasets used in the proposed approaches of this thesis are extracted from the open KGs namely DBpedia, Freebase, YAGO, and WordNet. Details regarding these KGs are provided below.

DBPEDIA DBpedia [4] is the first publicly available KG published in 2007. It is generated by an automated framework that extracts information from the *Wikipedia infoboxes*.

It also includes the categorization information, thumbnail images, links to external Web pages from Wikipedia articles, and geo-coordinates. The English version of the DBpedia describes 4.58 million things, out of which 4.22 million are classified in a consistent ontology, which includes 1,445,000 persons, 735,000 places, 123,000 music albums, 87,000 films and 19,000 video games, 241,000 organizations (including 58,000 companies and 49,000 educational institutions), 251,000 species and 6,000 diseases, etc.[3]

FREEBASE Freebase [17] was a large collaborative KG consisting of more than 4000 semantic types of entities, more than 125 million triples, and more than 7000 properties. It was made publicly available in 2007 and the data contained in Freebase was harvested from various sources including user-submitted wiki contributions as well as its community members. It was developed by Metaweb, an American software company. Later, Metaweb was acquired by Google but the Freebase API was discontinued in 2016.

WORDNET WordNet [86] is a lexical database of semantic relations between words in more than 200 languages. WordNet was first created in English only in the Cognitive Science Laboratory of Princeton University and had its first release in the 1980s. Words from the same lexical category that are synonymous are grouped together and are called *synsets*. The synsets contain short definitions and usage examples. WordNet links words into semantic relations including synonyms, hyponyms, and meronyms. The database contains 155,327 words organized in 175,979 synsets for a total of 207,016 word-sense pairs. It includes nouns, verbs, adjectives and adverbs but ignores prepositions, determiners and other function words.

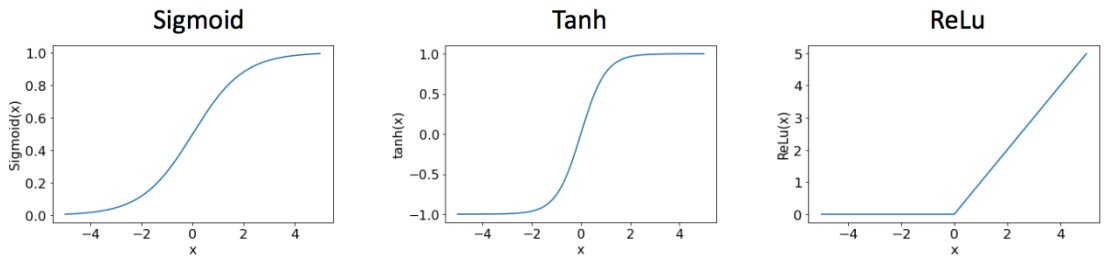
YAGO YAGO [129] is an open-source KG developed in 2007 at the Max Planck Institute for Computer Science in Saarland. As of 2019, YAGO3 has knowledge of more than 10 million entities and contains more than 120 million facts about these entities. The information in YAGO is extracted automatically from the categories, redirects, and infoboxes from Wikipedia, WordNet (e.g., synsets, hyponymy), and GeoNames. As described in [129, 131], the accuracy of YAGO was manually evaluated to be above 95% on a sample of facts. Furthermore, YAGO is linked to the DBpedia ontology and to the SUMO ontology.

2.3 NEURAL NETWORKS

Artificial Neural Networks (ANNs) comprise interconnected processing units called *Artificial Neurons*, the functionality of which is inspired by the biological neurons of the mammalian nervous system. McCulloch and Pitts [81] designed the first computa-

[3] http://wikidata.dbpedia.org/about

Figure 2.3: Illustration of different activation functions



tional model of neural networks in 1943. Later in 1958, Rosenblatt [115] proposed the perceptron model which is a neural network for pattern recognition. Neural networks offer an adaptive approach by learning patterns from the data to solve various problems. Activation function $g(\bullet)$ is used in the learning process of the neural network. It decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. There are different forms of activation functions for neural networks which are discussed below and the corresponding illustrations are shown in Figure 2.3.

THRESHOLD FUNCTION A threshold function generates the output 1, if the input exceeds a certain value n, otherwise it returns 0. The most commonly used threshold function is the Heaviside step function where $n = 0$ and is given by:

$$g(x) = \begin{cases} 1, & \text{if } x < 0 \\ 1 & \text{if } x \geqslant 0 \end{cases} \tag{2.1}$$

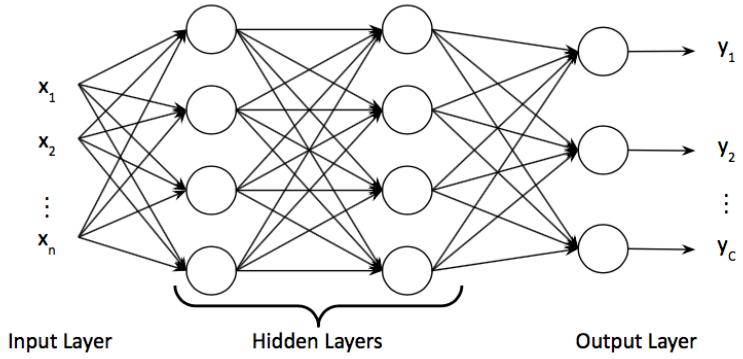
SIGMOID FUNCTION Sigmoid Function is a non-linear function that generates an S-shaped curve and maps its inputs into the interval of values of $]0, 1[$. It is effective for the backpropagation algorithm [144] as it has a non-zero gradient and is defined as:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

HYPERBOLIC TANGENT FUNCTION Similar to the sigmoid function, a Hyperbolic Tangent Function ($\tanh$) generates an S-shaped curve and it maps its inputs to values $]-1, 1[$. It is formulated as:

$$g(x) = \tanh(x) \tag{2.3}$$

Figure 2.4: An Artificial Neural Network with an input, two hidden and an output layers



RECTIFIED LINEAR UNITS Rectified Linear Unit (ReLU) uses a specialized ramp function for the activation, given by :

$$g(x) = max(0, x) \tag{2.4}$$

The computational complexity of ReLU is less than other activation functions such as sigmoid and $tanh$, hence widely used in complex neural networks with huge numbers of neurons. Over the years researchers developed different types of artificial neural networks [46]. However, in the following sections, the other neural network models are used in this dissertation such as Feed-Forward Networks, Convolutional Neural Networks (CNN), Long Short Term Memory (LSTM), and Gated Recurrent Unit (GRU).

2.3.1 *Feed-Forward Networks*

A Single Layer perceptron (SLP) is a feed-forward network based on a threshold transfer function. In a feed-forward neural network, the information only flows in one direction, from input to output. Each input is multiplied by a weight followed by a summation of the results together with a bias and passed on to an activation function. Formally, each unit is defined as,

$$f^{(1)}(x) = g(\sum_{n=1}^{N} w_n^{(1)} x_n + b_0^{(1)}), \tag{2.5}$$

where $g(\bullet)$ is the activation function and superscript of f and $w_1, ..., w_N$ represent a single-layer function and their internal weights respectively, and $b_0^{(1)}$ is the bias. However, SLPs are unable to distinguish non-linearly separable data. Therefore, Multi-

Layer Perceptron (MLP) is proposed which is also a feed-forward network consisting of multiple layers. These layers are fully connected to each other, i.e., every single neuron is linked to the neurons in the next layer. These neurons are generally arranged into three different layers: **(i)** Input layer, **(ii)** Hidden layer, and **(iii)** Output layer. The layers between the input and output layers are called hidden layers which process the information from the input layer, between the hidden layers and transmit it to the output layer. One such basic Feed-Forward Network is depicted in Figure 2.4 consisting of an input layer, two hidden layers, and one output layer. The input to the network is denoted by $x_1, x_2, ..., x_n$, where n is the total number of inputs and the output is given by $y_1, y_2, ..., y_C$, where C is the total number of classes.

2.3.2 *Convolutional Neural Network*

Neocognitron [41] is the first proposed Convolutional Neural Network (CNN) architecture used for the recognition of handwritten Japanese characters. Later, the training of CNN using the backpropagation algorithm by proposed in 1989 [72]. Since its commencement, CNN has gained immense popularity in image analysis as well as in various Natural Language Processing (NLP) based applications [74]. However, this section focuses on the basic working principle of CNN.

A CNN comprises multiple stages each of which takes a volume of feature maps as input and generates a new feature map. Each stage of CNN consists of three layers: **(i)** convolutional layer, **(ii)** A ReLU layer, and **(iii)** a pooling layer. Finally, the fully-connected layer maps the last layer onto a class of probability distribution in the output layer. The convolutional layer connects perceptrons locally preserving the information about the surrounding perceptrons and processing them depending on their corresponding weights. It detects local conjunctions of features from the previous layer and maps it to a feature map. Following the convolution layer in each stage, the data is processed using rectification and pooling subsamples from each layer. The most commonly used pooling techniques are MAX-pool and AVG-pool.

Each convolutional layer has m_1 filters and the number of filters applied in one stage is equivalent to the depth of the volume of the output feature maps. The filters help in detecting features on the input. The output $Y_i^{(l)}$ of layer l consists of $m_1^{(l)}$ feature maps of size $m_2^{(l)} \times m_3^{(l)}$. The i^{th} feature map $Y_i^{(l)}$ is computed as

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} K_{i,j}^{(l)} \star Y_j^{(l-1)}, \tag{2.6}$$

where $B_i^{(l)}$ is the bias matrix, $K_{i,j}^{(l)}$ is the filter connecting the j^{th} feature map in layer $(l-1)$ with i^{th} feature map in the layer. CNNs perform better for data with grid-like

topology as spatial relations between separate features are considered in the convolutional and pooling layers.

2.3.3 *Long Short Term Memory*

One of the first Recurrent Neural Networks (RNN), also known as the Hopfield Network, was proposed by J.J. Hopfield [55]. The major drawbacks of RNNs are that the range of contextual information is limited and the back-propagation through time does not perform well resulting in either vanishing or exploding the outputs of the network, which is popularly known as *Vanishing Gradient Problem* or *Exploding Gradient Problem*. To overcome this shortcoming, Long Short Term Memory (LSTM) network was introduced by Hochreiter & Schmidhuber in 1997 [52].

Each LSTM block consists of a forget gate, an input gate and an output gate. Applies a multi-layer long short-term memory (LSTM) RNN to an input sequence. For each element in the input sequence, each layer computes the following function:

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{ai}a_{t-1} + b_{ai}), \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{af}a_{t-1} + b_{af}), \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{ag}a_{t-1} + b_{ag}), \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ao}a_{t-1} + b_{ao}), \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t, \\ a_t &= o_t \odot \tanh(c_t), \end{aligned} \tag{2.7}$$

where i_t, f_t, g_t, o_t are the input, forget, cell, and output gates respectively, x_t is the input at time t, a_{t-1} is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0, a_t and c_t are the hidden state and the cell state at time t respectively. σ is sigmoid function and $\odot$ is the Hadamard Product. A Bidirectional LSTM (Bi-LSTM) models consist of two LSTMs to process input from both forward and backward direction. It improves the contextual information of input and is used in sequence processing.

2.3.4 *Gated Recurrent Unit*

Gated Recurrent Units (GRU) proposed by Kyunghyun Cho [23] in 2014 are similar to LSTM, which uses a gated mechanism to adaptively reset or update its memory content. It uses a reset gate and an update gate which are very similar to the forget gate and input gate of LSTM. However, it has lesser parameters than LSTM as it does not have the output gate. GRUs does not store the state of the cell and hence the whole memory is exposed at each time step. The LSTMs select the piece of information to be carried forward, whereas the GRU decides how much information needs to

be forgotten between two successive recurrent units. For each element in the input sequence, each layer computes the following function:

$$\begin{aligned} r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{ar}a_{t-1} + b_{ar}), \\ z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{az}a_{t-1} + b_{az}), \\ n_t &= \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{an}a_{t-1} + b_{an})), \\ a_t &= (1 - z_t) \odot n_t + z_t \odot h_{t-1}, \end{aligned} \tag{2.8}$$

where r_t, z_t, n_t are the reset, update, and new gates respectively, x_t and a_t are the input and hidden state respectively at time t, a_{t-1} is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0, σ is the sigmoid function, and $\odot$ is the Hadamard Product. Experimental results show that LSTM and GRU outperform the traditional tanh-unit [25]. Similar to Bi-LSTM, Bi-GRUs also process the input from the forward and backward directions.

2.4 LANGUAGE MODELS

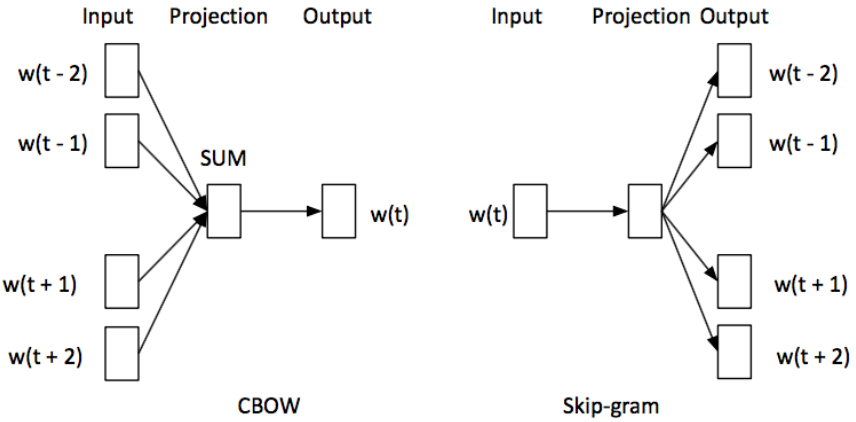
A Language Model (LM) learns the probability of word occurrences based on a text corpus which is used for various machine learning-based NLP applications such as Machine Translation [68], Speech Recognition [159], etc. It is the task of assigning a probability to each sequence of words or a probability for the likelihood of a given word based on a sequence of words [45]. *Statistical Language Models (SLMs)* are *n-gram* based approaches that assign probabilities to a sequence s of n words, and is given by

$$P(s) = P(w_1 w_2 ... w_n) = P(w_1)P(w_2|w_1)...P(w_n|w_1 w_2 ... w_{(n-1)}), \tag{2.9}$$

where w_i denotes i—th word in the sequence s. The probability of a word sequence is the product of the conditional probability of the next word given the previous words or the context [62]. The SLMs fail to assign probabilities to the n-grams that do not appear in the training corpus which is tackled using the smoothing techniques. However, the curse of dimensionality refrains the SLMs models to be trained on huge corpora.

Neural Language Models (NLMs), on the other hand, are neural network-based LMs that learn the distributed representation of words into a continuous low-dimensional vector space. The semantically similar words appear closer to each other in the embedding space. The contextual information is captured on all the different levels in the text corpus, such as sentences, sub-word, and characters, as well as the entire corpus. The NLMs marked a significant breakthrough for better performance for NLP-related problems. As explained in [108], the NLMs can be broadly classified into two categories: **(i)**Non-contextual Embeddings, and **(ii)** Contextual Embeddings. The detailed descriptions of different Non-contextual and Contextual Embeddings used in this thesis are provided in the sections below.

Figure 2.5: Illustration of the Word2vec model



2.4.1 *Non-contextual Embeddings*

The Non-contextual Embeddings map words or sub-words into distributed embeddings to implicitly represent the syntactic or semantic features of the language. Formally, a word w in a vocabulary $\mathcal{W}$ is mapped to a vector $\mathbf{w} \in \mathbb{R}^{\mathbf{D}}$, where D is the dimension of the embedding. These embeddings are trained on large-scale corpora and are static in nature. The following section comprises of detailed explanation of the different Non-contextual Embeddings used in this dissertation.

WORD2VEC Word2vec [85] is one of the pioneer models that uses a shallow neural network with two hidden layers for distributed representation of words into a low dimensional vector space. A sliding window of predefined length is moved across the text and in each step, the training is done with the words inside the window. The embeddings of the words sharing common contexts in the text appear closer to each other in the vector space. To encapsulate the semantic and syntactic information of the words, the word embeddings are generated using **(i)** Continuous Bag of Words (CBOW) and **(ii)** Skip-gram approach as depicted in Figure 2.5.

In *CBOW* approach, the model predicts the current word from a window of surrounding context words. However, the order of the context words in the text does not have any influence on the prediction process. On the other hand, the *Skip-gram* model uses the current word to predict the surrounding window of context words. More weight is applied to the nearby context words compared to the distant context words. Skip-gram is efficient with less training data and non-frequent words are well presented, whereas CBOW performs better with repeated words. The Word2vec model is trained with hierarchical softmax and negative sampling. The hierarchical softmax

model is based on the Huffman tree which is a binary tree. It returns all the words depending on their counts and is normalized at each step from the root node to the target node. The negative sampling method minimizes the log-likelihood of the sampled negative instances. The hierarchical softmax works better for infrequent words and negative sampling is efficient with low dimensional vectors and for repeated words.

GLOVE The Word2vec model captures the semantics of the words but the connectivity of the words is ignored in the model. To overcome this, Global Vectors for Word Representation (GloVe) [94] is an unsupervised learning algorithm proposed. The model is trained on aggregated global word-word co-occurrence statistics from a corpus resulting in word embeddings. An element x_{ij} in the co-occurrence matrix $\mathcal{X}$ denotes the co-occurrence of the words w_i and w_j in an appropriate context window. It extends the Word2vec models by predicting the current word from its neighbouring context words. It is a two-step model involving the generation of the co-occurrence matrix in the first step, followed by a factorization method to generate the word embeddings.

FASTTEXT The main limitation of both Word2vec and GloVe is that these models are unable to generate embeddings for Out-of-Vocabulary (OOV) words. FastText [15] overcomes this drawback by breaking a word into n-grams or sub-words which are then provided as input to the neural network. It learns the semantics of the words and the relationships between the characters in the words on the sub-word level. Therefore, it is capable of generating better representations of the rare words in the corpus along with the OOV words. The model is proposed by Facebook's AI Research lab and has been trained on Wikipedia for 294 languages using the default parameters from the Word2vec skip-gram model [63]. However, the original paper [15] mentions the usage of the CBOW approach of Word2vec.

WIKIPEDIA2VEC Wikipedia2vec [156] jointly maps words and entities into the same continuous vector space such that similar words and entities are closer to each other in the vector space. The model learns the embeddings of the words and entities from Wikipedia using three sub-models, namely **(i)** Wikipedia Link Graph Model, **(ii)** Word-based skip-gram model, and **(iii)** Anchor Context model. All these sub-models use the skip-gram architecture from Word2vec. The Wikipedia Link Graph is an undirected entity-entity graph in which each node is an entity in Wikipedia and there exists an edge between two nodes if the page of one entity has a link to the other entity. The entity embeddings are learned by predicting the neighbouring entities in the Wikipedia Link Graph. The Word-based skip-gram model learns the word embeddings by predicting the neighbouring words of a given word. The anchor context is built by obtaining referent entities and their neighbouring words from the links contained on the Wikipedia page. The model learns by predicting the context words given to each entity.

2.4.2 *Contextual Embeddings*

The Non-contextual Embeddings are static in nature and have the same embeddings of the words irrespective of the given context. Hence, it fails to encode the polysemous words. Contextual Embeddings are proposed to distinguish the semantics of the words in different contexts. Given a text $w_1, w_2, ..., w_n$, where each token $w_i \in \mathcal{W}$ is a word or sub-word, the contextual embeddings of the token w_i is $\mathbf{h_t}$ and is formulated as,

$$[\mathbf{h_1}, \mathbf{h_2}, ..., \mathbf{h_n}] = \mathbf{f_{encode}}(\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_n}), \tag{2.10}$$

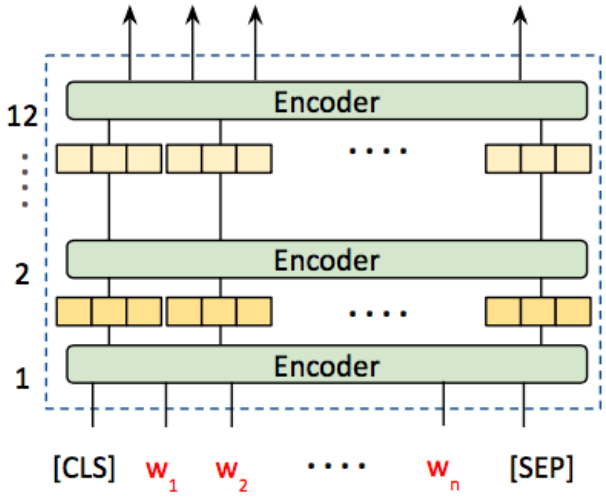
where $f(\bullet)$ is the encoder. The following sections focus on detailed descriptions of the Contextual Embedding models namely BERT, S-BERT, and GPT-2, that are used in the models discussed in the chapters later.

BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT) It is a transformer-based unsupervised model proposed by Google [35]. BERT has emerged from earlier pre-training contextual embedding models such as semi-supervised sequence learning [30], generative pre-training, ELMo [97] and ULMFit [56]. Unlike the static context-free embedding models such as Word2vec, GloVe, etc., BERT takes into account the context for each occurrence of a given word. For example, the embedding of the word *anchored* is the same for the static context-free embedding models for both of its occurrences in the following sentences:

*She **anchored** the television documentary series last year.*
*The boat is **anchored** in the lee of the island.*

BERT generates different contextual embeddings according to the sentences. The architecture builds upon the original transformer model proposed by Vaswani et al. [139] and it comprises a variable number of encoder layers and self-attention heads. The original BERT trained on the English language has two variants: **(i)** BERT-base model which has 12 encoders, 12 bidirectional self-attention heads, and 768 hidden units, and 110M parameters and **(ii)** BERT-large model containing 24 encoders, 16 bidirectional self-attention heads, 1024 hidden units, and 340M parameters. The input to the encoder in the BERT model is a sequence of tokens, which are first converted into vectors and then processed in the neural network. Each layer then applies self-attention, and passes its results through a feed-forward network, and then to the next encoder. The input embeddings are the sum of the *token embeddings, segmentation embeddings* and *position embeddings*. For *token embeddings*, a [CLS] token is added to the input word tokens at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence. In the case of *segment embeddings*, a marker indicating the sentences is added to each token which allows the encoder to identify different sentences. A *positional embedding* is added to each token to indicate its position in the sentence. The transformer then stacks a layer that maps sequence to sequence and the output has the same sequence of vectors corresponding to the input. BERT is pre-trained together

Figure 2.6: Illustration of the BERT model



on two tasks: **(i)** Masked Language Modelling (MLM), **(ii)** Next Sentence Prediction (NSP). In the *MLM* task 15% of the tokens are masked and the model is trained to predict the masked words from the context. During training, the prediction is done only on the masked tokens, while the non-masked ones are ignored. For *NSP*, given a sentence, the model is trained to predict if a chosen next sentence is random or not, with the assumption that the random sentence will be disconnected from the first sentence. During training, two sentences are provided as input to the BERT model and the output of the classification token [CLS] is transformed using a classification layer and the IsNext-Label is assigned using the softmax function. The BERT model can be easily fine-tuned and has been applied across a wide variety of tasks under general language understanding like natural language inference, sentiment analysis, question answering, paraphrase detection, linguistic acceptability, etc. Figure 2.6 illustrates the BERT-base model with 12 encoder layers, and the input to the model is a sequence of n words given by $w_1, w_2, ..., w_n$ together with the special tokens [CLS] and [SEP]. Furthermore, Chapter 7 of this dissertation shows that this contextual embedding model BERT plays a significant role in predicting missing semantic types of the entities in a KG.

SENTENCE-BERT (SBERT) The most common way of generating a sentence embedding in the BERT model is by averaging all the word-level embeddings or by using the output of the first token i.e., the [CLS] token. However, experimental results in [112] show that these sentence embeddings often perform worse than the sentence embeddings obtained from the GloVe embeddings for several downstream tasks for various

tasks such as textual similarity, Wikipedia Sections Distinction, etc. SBERT [112] model tackles all the above-mentioned problems.

SBERT fine-tunes the BERT model using the siamese and triplet networks to update the weights such that the resulting sentence embeddings are semantically meaningful and semantically similar sentences appear closer to each other in the embedding space. It is fine-tuned with a 3-way softmax classifier objective function for one epoch. The two input sentences (say u and v) to the SBERT model are passed through the BERT model followed by a pooling layer namely, CLS-token, MEAN-strategy, and MAX-strategy are appended on top of it. This pooling layer enables the generation of a fixed-size representation for the input sentences. It is then concatenated with the element-wise difference and multiplied with a trainable weight, W, and is optimized using cross-entropy loss. In order to encode the semantics, the twin network is fine-tuned on Semantic Textual Similarity data. The SBERT model is first trained on Wikipedia via BERT and then fine-tuned on Natural Language Inference (NLI) data. NLI is a collection of 1,000,000 sentence pairs created by combining The Stanford Natural Language Inference (SNLI)[4] and Multi-Genre NLI (MG.NLI) datasets. Later in Chapter 5 and Chapter 8 it is observed that SBERT embeddings generated for the textual entity descriptions help in KG completion.

GENERATIVE PRE-TRAINED TRANSFORMER 2 (GPT-2) GPT-2 [110] is a transformer-based language model which is trained with the objective of predicting the next word given all the previous words within some text. The model is trained with 1.5 billion parameters and a dataset of size 40 GB which consists of 8 million web pages. GPT-2 is a direct scale-up of GPT [109] with a ten-fold increase in the parameters and the size of the training data. It is a decoder-only transformer model comprising 12 layers of decoders using 12 masked attention heads with 64-dimensional states for each attention head. Unlike BERT, in GPT-2 the future tokens of masked, and hence, in the calculation of self-attention the tokens to the right of the current token are blocked. For instance, to make predictions for the i^{th} token w_i in the sequence $w_1, w_2, ..., w_n$, only input tokens from 1 to i $w_1, w_2, ..., w_i$ are considered while the tokens $w_{i+1}, ..., w_n$ are ignored in the mask mechanism for self attention. The word vectors used in the first layer of GPT-2 are generated using Byte Pair Encoding (BPE). BPE is a data compression technique in which the most common pair of consecutive bytes of data is replaced with a byte that does not occur within that data. The Adam optimization algorithm is used in GPT-2 and the learning rate is increased linearly from 0 to 2.5×10^{-4} over the first 2000 iterations using the cosine annealing[5]. GPT-2 is a self-supervised model i.e., it is pre-trained on raw texts with an automated process to generate inputs and labels from those texts. GPT-2 has been used in a wide range of tasks such as text generation, question answering, text summarization, translation, etc. Furthermore, GPT-2 is

[4] https://nlp.stanford.edu/projects/snli/ [5] Cosine Annealing is a type of learning rate schedule that has the effect of starting with a large learning rate that is relatively rapidly decreased to a minimum value before being increased rapidly again.

capable of generating full sentences as well as comprehensible and semantically meaningful paragraphs by continuing to predict tokens or words in the sequence [110]. It is to be noted that GPT-2 has a task-agnostic architecture and has not been trained specifically for any of the aforementioned downstream tasks. Despite this, fine-tuned GPT-2 outperforms task-oriented RNN, CNN, and LSTM-based models on different tasks [109]. In this dissertation, the general-purpose task-agnostic feature of GPT-2 is leveraged for KG completion, a detailed description of which is provided in Chapter 6.

2.5 NETWORK EMBEDDINGS

Network embedding aims to learn the latent representations of the nodes of a graph into a continuous low-dimensional vector space. In network embeddings, the intrinsic information of the network (or the graph) is to be preserved, noise and redundant information are to be removed and the embeddings of the similar nodes should be closer to each other in the vector space. A network embedding model roughly includes the following steps:

- decide on the dimension of the embedding vector,
- randomly initialize embeddings for each node, and
- learning the embeddings by repeatedly incrementally improving the embeddings preserving the similarity in the network by solving an optimization problem.

The learned node embeddings can effectively support in downstream tasks such as predicting unseen links, identifying important nodes, and inferring node labels. One of the advantages of node embedding is that feature engineering by domain experts is not required for downstream tasks. Further details on the different network embedding models can be found in [3, 28]. Chapter 7 of this dissertation shows how Node2vec [47] model can be leveraged to predict the missing entity types for Knowledge Graph Completion. Therefore, a detailed explanation of the model is provided below.

NODE2VEC Node2vec [47] learns the latent representation of the nodes in a graph by preserving neighbourhood information. Biased random walks based on an efficient network-aware search strategy are generated from target nodes. In the network, the selection of the next hop in the first-order random walk is done based on the transition probability calculated by normalizing the edge weights and is given by,

$$P(u|v) = \frac{w_{uv}}{\sum_{u' \in N_v} w(u'v)} = \frac{w_{uv}}{d(v)}, \tag{2.11}$$

where $u, v \in V$ are the nodes, N_v are the neighbouring nodes of v, $d(v)$ is the degree of node v, and $w(u, v)$ is the weight of the edge between the nodes u and v. The second

order transition applies a bias factor α to reweigh the edge weights depending on the previous state and the transition probability is given by,

$$P(u|v,t) = \frac{\alpha_{pq}(t,u)w(u,v)}{\sum_{u' \in N_v} \alpha_{pq}(t,u')w(u',v)}, \tag{2.12}$$

where $t, u, v \in V$, and N_v are the neighbouring nodes of v. The random walk has already traversed the edge (t,v) and the transition probability is calculated for the next node from starting from v. The bias factor α is given by,

$$\alpha_{pq}(t,u) = \begin{cases} \frac{1}{p} & d_{tu} = 0 \\ 1 & d_{tu} = 1 \\ \frac{1}{q} & d_{tu} = 2 \end{cases}, \tag{2.13}$$

where d_{tu} determines the shortest distance between the nodes t and u, parameter p controls the likelihood of immediately revisiting a node in the walk, and q allows the search to differentiate between inward and outward nodes. These random walks are treated as sentences in the skip-gram model to generate the node embeddings in the node2vec model. The main idea is to maximize the probability of predicting the correct context node given the centre node.

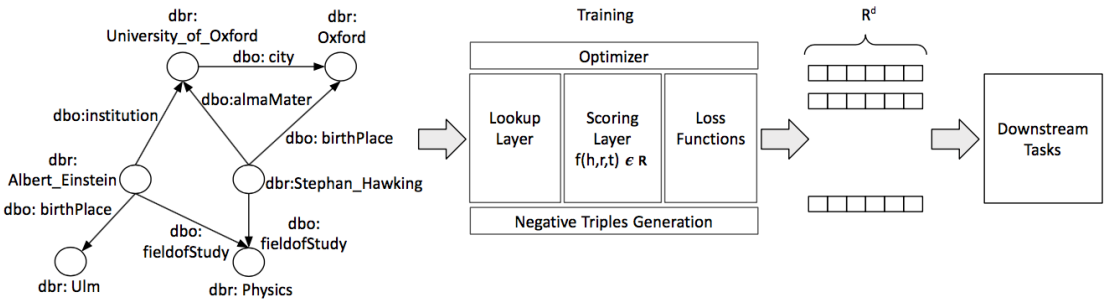
The second order transition in the random walks that stores the interconnections between the neighbours of every node [47]. Furthermore, the neighbourhoods N_v considered in the random walks are not only restricted to immediate neighbours but also extended to vastly different structures within the network depending on the sampling strategy.

2.6 KNOWLEDGE GRAPH EMBEDDINGS

The main goal of Knowledge Graph Embedding (KGE) models is to generate a latent representation of the entities and relations in a KG to a continuous low-dimensional vector space that can be used for different knowledge acquisition tasks and downstream applications. The dimension d of the entity and the relation vectors is fixed and often low ranging between $50 \leqslant d \leqslant 1000$ [54]. The entity embeddings (or vectors) and relation embeddings (or vectors) are usually denoted by $\mathbf{e}$ and $\mathbf{r}$ respectively. As discussed in [58], a typical KGE model is characterized by the following steps:

- *Representation Space*, the low-dimensional space in which the relations and entities are represented. Entities are represented as vectors or modelled as multivariate Gaussian distributions. Relations, on the other hand, can be represented as vectors, matrices, tensors, multivariate Gaussian distributions, or even mixtures of Gaussians.

Figure 2.7: Generalized framework of a KGE model



- *Scoring Function* given by $f_r(e_h, e_t)$ is defined on each triple $< e_h, r, e_t >$ to measure its plausibility. The triples observed (or true triples) in the KG tend to have a higher score and lower scores are assigned to false/negative/corrupted triples.
- *Encoding Models* for representing and learning relational interactions between the entities. The model learns the representations of the entities and relations by solving an optimization problem that maximizes the total plausibility of observed triples. Negative or corrupted triples are generated in this step and the method used to generate negative samples has an impact on learning the embeddings [69].
- *Auxiliary Information*: Any additional information available in the KG, such as literals, that can be leveraged to enrich the embeddings of the entities and the relations. In such a scenario, an ad-hoc scoring function is defined for the additional information and is integrated into the general scoring function.

All the aforementioned steps are depicted in Figure 2.7. An extensive study of the existing KGE models is provided in Chapter 3. Furthermore, Chapter 5 discusses the shortcomings in the baseline models and proposes a new KGE model for link prediction in KGs.

2.7 EVALUATION METRICS

As mentioned earlier in Chapter 1, the problem of KGC can be further subdivided into sub-problems, namely **(i)** link prediction in KGs, and **(ii)** entity type prediction. Furthermore, the task of link prediction in KGs includes two subtasks which are head/tail prediction and triple classification. The entity type prediction problem, on the other hand, is converted to a classification problem with semantic types of the entities as

classes in the classifier. Therefore, different evaluation metrics are used for the different subtasks of the KGC problem. This section comprises detailed explanations of the evaluation metrics used in different models presented in this thesis.

The most commonly used evaluation metrics for the Link Prediction task that includes the head or tail entity prediction of the triples are Mean Reciprocal Rank (MRR), and Hits@k. For triple classification, accuracy is used as an evaluation metric for the classifier. The evaluation metrics Accuracy, Macro-F_1, and Micro-F_1 are used for entity type prediction.

MEAN RECIPROCAL RANK Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks of the correct entities. It scores the predicted triples based on whether they are true or not. If the first predicted triple is true, its score is 1, and the second true score is $\frac{1}{2}$, and so on. When the n^{th} triple is established, it is scored $\frac{1}{n}$, and the final MRR value is given by,

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{q}, \tag{2.14}$$

where q is the prediction/recommendation item, and Q represents all the prediction/recommendation items given by the model. The larger the MRR value, the better the model effect. MRR is a commonly used index to measure the effect of search algorithms [22].

HITS@K Hits@k is the proportion of the correct entities in top-k predictions, mathematically given by,

$$\text{Hits@k} = \frac{|\{q \in Q : q < k\}|}{|Q|} \tag{2.15}$$

where q represents the prediction/recommendation item, and Q represents all the prediction/recommendation items given by the model. The value of Hits@k is between 0 and 1. The larger the value, the better the algorithm works. Hits@K reflects the accuracy of an embedding model to predict the relation between two given triples correctly. Hits@k is an indispensable evaluation metric for link prediction tasks in KGC and $k = 1, 3$, and 10 are usually chosen [22].

ACCURACY Accuracy is the proportion of correct predictions among the total number of cases examined in a classification problem. Mathematically it is given by,

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.16}$$

where TP, TN, FP, and FN are true positives, true negatives, false positives, and false negatives respectively. In multi-class classification problems such as in entity type prediction in KGs, accuracy is expressed in percentage and is calculated as,

$$Accuracy = \frac{\text{correct classifications}}{\text{all classifications}} \tag{2.17}$$

In a multiclass classification problem, a prediction is considered correct when the class with the highest score matches the class in the label.

F1-SCORE, MACRO-F1 AND MICRO-F1 F_1-score is defined as the harmonic mean of the precision and recall:

$$F_1 - score = 2 \star \frac{\text{Precision} \star \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.18}$$

where *Precision* is the fraction of retrieved items that are relevant to the query. In classification problems, Precision $= 1$ refers that all samples that are classified as the positive class are truly positive. *Recall* is the fraction of the relevant documents that are successfully retrieved. In classification, Recall $= 1$ means all the truly positive samples were predicted as the positive class. F_1-score is a measure used to assess the quality of binary classification problems as well as problems with multiple binary labels or multiple classes. The best F_1-score is 1 while the worst value is 0.

The Macro F_1-score (Ma-F_1) is defined as the mean of class-wise F_1-scores and is formally given by,

$$Ma - F_1 = \frac{1}{C} \sum_{i=0}^{C} F_1 - score_i \tag{2.19}$$

where i is the class index and C the number of classes. In Ma-F_1, the F_1 score is calculated for each class followed by their unweighted mean. It considers all the classes equally and thereby the class imbalance is not taken into account. It performs better in the popular classes (or frequent classes) whereas drops poorly in the rare classes.

Micro F1-score (Mi-F_1) is the harmonic mean of the precision and recall and is mathematically defined as,

$$Mi - F_1 = 2 \star \frac{\text{Mi-Precision} \star \text{Mi-Recall}}{\text{Mi-Precision} + \text{Mi-Recall}} \tag{2.20}$$

It measures the F1-score of the aggregated contributions of all classes by globally counting the total true positives, false negatives and false positives. The value of Mi-F_1 ranges between 0 and 1 with 1 being the best score and 0 being the worst. The micro-averaging allows more emphasis on the most frequent classes. The classes with lesser data do not have a significant influence on the overall F_1 score if the model

performs well for the popular classes. This metric is widely used to assess the quality of multi-label binary classification.

3

LINK PREDICTION - LITERATURE REVIEW

One of the important sub-problems of KGC is link prediction. In the past decade, there has been a significant rise in interest in the research on generating knowledge graph embeddings to predict the missing links in KGs. Based on the techniques used, this chapter gives a categorization of the most recent, state-of-the-art (SoTA) link prediction models, followed by comprehensive details of the models.

3.1 INTRODUCTION

Link Prediction is a fundamental task of KGC that aims to estimate the likelihood of the existence of links between entities based on the current observed structure of the KG. The typical graph topology, however, makes KGs difficult to manipulate for different tasks due to the huge amount of information available. The primary disadvantages of conventional KGs are computationally expensive, and data sparsity. As a result, the need to learn a KG's latent representation in a low-dimensional space arises. Knowledge graph embeddings can utilize self-supervision to develop a low-dimensional numerical model of a KG that (usually) translates input edges to an output plausibility score indicating the likelihood of the edge is true. KG embedding-based link prediction models focus on different features from the KGs. Inspired by [43], a categorization of the existing KG embeddings based on the embedding techniques is provided in Table 3.1.

Using the KG definition from the previous chapter (see Section 2.2 of Chapter 2), a KG $\mathcal{G}$ consists of a set of triples $\mathcal{T}$, given by, $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{C})$, where $\mathcal{E}$, $\mathcal{R}$, $\mathcal{L}$, $\mathcal{C}$ are the set of entities, relations, literals, and semantic types or classes respectively. A triple $< e_h, r, e_t > \in \mathcal{T}$ in a KG $\mathcal{G}$, is an ordered set, where $e_h \in \mathcal{E}$ is the head entity, $r \in \mathcal{R}$ is the relation, and $e_t \in \mathcal{E} \cup \mathcal{L} \cup \mathcal{C}$ is the tail entity.

As this dissertation focuses of leveraging the contextual information of the entities in a KG, the SoTA KG embedding models exploiting those features are discussed extensively along with an overview of the other models are provided.

3.2 TRANSLATION-BASED MODELS

In TransE, given a triple $< e_h, r, e_t >$ in a KG G, the relation r is considered as a translation operation between the head (e_h) and tail (e_t) entities on a low dimensional space defined by $\mathbf{e_h} + \mathbf{r} \approx \mathbf{e_t}$, where $\mathbf{e_h}, \mathbf{r}, \mathbf{e_t}$ are the embeddings of the head, relation and the tail entity respectively. TransH [141] extends TransE by projecting the entity

Table 3.1: Link Prediction Models and their categories

Categories		Models
Translational Models		TransE [19] and its variants, RotatE [132], etc.
Semantic Matching Models		RESCAL [91] and its extensions, DistMult [157], HoIE [90], SME [20], etc.
Path based Models		GAKE [40], PTransE [75], PConvKB [59], etc.
Logical Rule based Models		KALE [48], LS-KBC [143]
Neural Network Based Models		NTN [126], HypER [5], ConvE [34], ConvKB [29], R-GCN [118], etc.
Literal Based Models	Text-based	DKRL [150], Jointly(ALSTM) [153], SSP [148], KG-BERT [158],BLP [32]
	Numeric-based	KBLRN [42], LiteralE [71], TransEA [147], MT-KGNN [134]
	Multi-modal	MKBE [99], IKRL [149], etc.

vectors to relation-specific hyperplanes. TransR [76] models entities and relations into distinct semantic spaces and projects entities from the entity space to the relation spaces. The scoring function of RotatE models the relation as a rotation in a complex plane to preserve the symmetric/anti-symmetric, inverse, and composition relations in a KG. For a triple $< e_h, r, e_t >$, the relation among them can be represented as $e_t = e_h \circ r$, where $|r| = 1$, i.e., restricting it to the unit circle and $\circ$ represents element-wise product.

3.3 SEMANTIC MATCHING MODELS

Semantic Matching Model (SME) is based on semantic matching using neural network architectures. Given a triple $< e_h, r, e_t >$ in a KG G, it projects entities and relations to their vector embeddings in the input layer. The relation vector $\mathbf{r}$ is then combined with the head entity vector $\mathbf{e_h}$ and tail entity vector $\mathbf{e_t}$ to get $g_u(\mathbf{e_h}, \mathbf{r})$, and $g_v(\mathbf{e_t}, \mathbf{r})$ respectively in the hidden layer. The final score is given by matching g_u and g_v via their dot product. In DistMult, each entity is mapped to a $d-$dimensional dense vector and each relation is mapped to a diagonal matrix, and the score of a triple is computed with the help of matrix multiplication between the entity vectors and the relation matrix. RESCAL models the triple $< e_h, r, e_t >$ into a three-way tensor, X. In X, two modes hold the concatenated entities (e_h, e_t), and the third mode holds relation r.

The model explains triples via pairwise interaction of latent features. The score of the triple is calculated using the weighted sum of all the pairwise interactions between the latent features of the entities (e_h, e_t). ComplEx extends DistMult by introducing complex-valued embeddings so as to better model asymmetric relations. It allows joint learning of head and tail entities by using the Hermitian dot product. In this model, the entity and relation embeddings lie in a complex space. Holographic Embedding (HolE) intend to overcome the curse of dimensionality of tensor product used in RESCAL by using circular correlation. Furthermore, this circular correlation is not commutative allowing HolE to model asymmetric relations.

3.4 NEURAL NETWORK BASED MODELS

NTN represents an entity using the average of the word embeddings in the entity name. ConvE uses 2D convolutional layers to learn the embeddings of the entities and relations in which the head entity and the relation embeddings are reshaped and concatenated which serves as an input to the convolutional layer. The resulting feature map tensor is then vectorized and projected into a k-dimensional space and matched with the tail embeddings using the logistic sigmoid function minimizing the cross-entropy loss. In ConvKB, each triple $< e_h, r, e_t >$ is represented as a 3-column matrix which is then fed to a convolution layer. Multiple filters are operated on the matrix in the convolutional layer to generate different feature maps. Next, these feature maps are concatenated into a single feature vector representing the input triple. The feature vector is multiplied with a weight vector via a dot product to return a score which is used to predict whether the triple is valid or not. HypER model uses a hyper network approach to generate convolutional filter weights for each relation. It is a method by which one network generates weights for another network enabling weight-sharing across layers. A hyper network projects a relation embedding via a fully connected layer, the result of which is reshaped to give a set of convolutional filter weight vectors for each relation. Relational Graph Convolutional Network (R-GCN) extends Graph Convolutional Network (GCN) [67] to handle different relationships between entities in a KG. GCN is a semi-supervised learning approach to graph-structured data. It takes a feature matrix containing the feature description of each node as an input as well as an adjacency matrix. A CNN model is trained on the input and the model learns hidden layer representations that encode both local network structure and node attributes growing linearly with the number of graph edges. In R-GCN, different edge types use different weights and only edges of the same relation type r are associated with the same projection weight. For each node, an R-GCN layer operates in two steps. First, it computes the outgoing message using node representation and weight matrix associated with the edge type. Then the incoming messages are aggregated to generate new node representations. A 2-D matrix is used to define the initial node features and a 3-D tensor describes the node's hidden features. This tensor is able to encode different

relations by stacking r batches of matrices, where each of these batches encodes a single typed relation.

3.5 PATH BASED MODELS

PTRANSE The model extends TransE by introducing relation paths as features. Given a triple, $< e_h, r, e_t >$, it considers the path defined as $p = r_1 \rightarrow r_2 ... \rightarrow r_l$ between the head and tail entity. It considers the three types of composition operations i.e.,

$$\begin{aligned} \text{addition: } p &= r_1 + r_2 + ... + r_l \\ \text{multiplication: } p &= r_1 \circ r_2 \circ ... \circ r_l \\ \text{RNN: } p_i &= f(W[p_{i-1}; r_i]) \end{aligned} \tag{3.1}$$

Here, p_i is the accumulated path vector at the i^{th} relation, W is a composition matrix shared by all relations, f is the non-linear function, and $[p_{i-1}; r_i]$ denotes their concatenation. PTransE defines the loss for each positive triple w.r.t the paths as follows:

$$\mathcal{L}_{path} = \frac{1}{Z} \sum_{p in \mathcal{P}(\rceil_\langle, \rceil_\sqcup)} R(p|e_h, e_t).l(p, r) \tag{3.2}$$

in which $P(e_h, e_t)$ is the set of all paths connecting e_h and e_t, $R(p|e_h, e_t)$ indicates the reliability of a path p given the two entities, Z is the normalisation factor, and $l(p, r)$ is the loss specified on the path-relation pair. The loss function of TransE on aggregated $\mathcal{L}_{path}$ is computed over all the positive triples to learn the representations of the entities and relations.

PCONVKB The model [59] extends the ConvKB model by exploiting the path information of the KGs. Additionally, it uses an attention mechanism to measure the local importance in relation paths. Similar to the PTransE model, it also generates paths between the head and the tail entity. Similar to ConvKB, for each triple $< e_h, r, e_t >$ the score function of our model PConvKB is defined as follows:

$$f(e_h, r, e_t) = \sigma(\psi([\mathbf{e_h}, \sum_{i=1}^{N} \Phi_{G_i} \times \Phi_{L_i} \times \mathbf{p_i} + \mathbf{r}, \mathbf{e_t}] \times \Omega)).w \tag{3.3}$$

in which σ is the non-linear sigmoid function, ψ is the average pooling, Φ_{G_i} and Φ_{L_i} denote the global and local importance respectively of the i^{th} path, $\mathbf{p_i}$ is the embedding of the i^{th} path, $\mathbf{e_h}, \mathbf{r}, \mathbf{e_t}$ are the embeddings of the head, relation and tail entity respectively, and $\Omega$and$w$ are shared parameters. The path embedding is obtained by $\mathbf{p_i} = \sum \mathbf{r}$.

GAKE GAKE [40] is a graph-aware embedding model that takes into account three types of graph structure namely neighbour context, path context, and edge context. The neighbour context is the same as the triple information present in the KGs. Path context includes several hops from the entities. Lastly, the edge context is defined by all the incoming and outgoing links to and from an entity. It uses an attention mechanism to identify important entities in the context neighbourhood that have more representative power and hence impact the embeddings. The graph contexts are combined by jointly maximizing the objective functions:

$$O = \lambda_{\mathcal{N}} \mathcal{O}_{\mathcal{N}} + \lambda_{\mathcal{P}} \mathcal{O}_{\mathcal{P}} + \lambda_{\mathcal{E}} \mathcal{O}_{\mathcal{E}} \tag{3.4}$$

where $\lambda_{\mathcal{N}}, \lambda_{\mathcal{P}} and \lambda_{\mathcal{E}}$ represent the weightage from neighbour context, path context and edge context respectively. The stochastic gradient descent (SGD) algorithm is used to estimate model parameters by optimizing Equation 3.5. The derivatives are calculated using the back-propagation algorithm. Hierarchical Softmax is used to speed up the training process and hence to reduce the time complexity of normalization.

3.6 LITERAL BASED MODELS

Another set of algorithms improves KG embeddings by taking into account different kinds of literals such as numeric, text or image literals. Models such as TransEA [147] and KBLRN [42] incorporate numeric literals into their embedding spaces, whereas MKBE is a multi-modal KG embedding model which includes the numeric, text and image literals present in KGs into their embedding spaces. A detailed analysis of the literal-based KG embedding models is provided in [43]. Text-based models are extensively described below because this thesis focuses on analyzing the effects of textual entity descriptions on link prediction for KGC.

DKRL DKRL [150] extends TransE [19] by incorporating the textual entity descriptions in the model. Two types of vector representations, namely structure-based e_s and description-based e_d, are learned simultaneously into the same vector space for each entity e. Given a triple $<e_h, r, e_t>$, the energy function is defined as,

$$E = \|h_s + r - t_s\| + \|h_d + r - t_d\| + \|h_s + r - t_d\| + \|h_d + r - t_s\|, \tag{3.5}$$

in which h_s and t_s are the structured-based representations h_d and t_d are the description-based representations of the corresponding head and tail entities respectively. TransE is used to learn the structure-based representation, while the textual entity descriptions are encoded using a continuous bag-of-words (CBOW) approach as well as a CNN-based approach. The short text is generated from the entity description based on keywords and their corresponding pre-trained word embeddings are summed up to generate the entity embedding in the CBOW approach. The CNN model comprises

five layers with each convolutional layer followed by a pooling layer, with tanh or ReLU as the activation function. CNN preserves the sequence of words, and hence performs better than CBOW. During the training of the model, a margin-based score function is considered an objective function and minimized using Stochastic Gradient Descent (SGD).

JOINTLY (ALSTM) Jointly (ALSTM) [153] is another entity description-based embedding model which extends the DKRL model with a gate strategy and uses attentive LSTM to encode the textual entity descriptions. The appropriate balance between structure-based and description-based representations is accomplished by incorporating the gated mechanism. Unlike DKRL, Jointly (ALSTM) uses bag-of-words, LSTM, and Attentive LSTM to generate description-based representations. For the structure-based representations, existing embedding models such as TransE are used. The joint representation is a linear interpolation between these two types of representations and is mathematically given by,

$$\mathbf{e} = g_e \odot \mathbf{e_s} + (1 - g_e) \odot \mathbf{e_d}, \tag{3.6}$$

where $\odot$ is an element-wise multiplication and g_e is the gate mechanism which is a real-valued vector. Jointly (ALSTM) follows the scoring function of TransE and is defined as follows:

$$f(e_h, r, e_t; d_h, d_t) = \|(\mathbf{g_h} \odot \mathbf{h_s} + (1 - \mathbf{g_h}) \odot \mathbf{h_d}) + r - (\mathbf{g_t} \odot \mathbf{h_t} + (1 - \mathbf{g_t}) \odot \mathbf{t_d})\|_2^2 \tag{3.7}$$

where $\mathbf{h_s}, \mathbf{h_d}$, and$\mathbf{g_h}$ are the structured-based, description-based and gate respectively for the head entity and $\mathbf{t_s}, \mathbf{t_d}$, and$\mathbf{g_t}$ are the structured-based, description-based and gate respectively for the tail entity.

KG-BERT KG-BERT [158] is a contextual neural language model-based approach which fine tunes the BERT model [35] on the KGs. Each triple $< e_h, r, e_t >$ is considered as a sentence and is provided as an input sentence of the BERT model for fine-tuning. For the entities, KG-BERT has been trained with either the entity names or their textual entity descriptions and for relations, the relation names are used. The first token of every input sequence is always [CLS], whereas the separator token [SEP] separates the head entity, relation and tail entity. The elements separated by the [SEP] token have different segment embeddings. The tokens in the head and tail entity name or the sentences in their corresponding descriptions share the same segment embedding e_A whereas the for relation the segment embedding is different and is denoted by e_B. Different tokens in the same position $i \in \{1, 2, 3, ..., 512\}$ have the same position embedding. The token representations are fed into the BERT model architecture. The aggregate sequence representation for determining triple scores is the last hidden

state corresponding to [CLS]. Classification layer weights are the only new parameters added during triple classification fine-tuning and is given by $W \in \mathbb{R}^{2\times A}$, where A is the hidden state size of the pre-trained BERT. The sigmoid function is used for the triple classification on the last hidden state.

BLP FRAMEWORK BLP [32] proposes the use of a pre-trained language model for learning representations of entities via a link prediction objective. It introduces a holistic evaluation framework for entity representations, that comprises link prediction, entity classification, and information retrieval. It uses BERT-based entity representations from textual entity descriptions for link prediction. For relations, the initial vectors are randomly assigned which are then optimised using SGD. The framework uses different base KG embedding models namely TransE, DistMult, ComplEx, SimplE to predict the missing links. For an entity, $e_i \in \mathcal{E}$, the problem of optimizing the embeddings of the entities and the relations in the graph is done by minimising the loss function given by,

$$\sum_{(e_i,r_j,e_k)\in\mathcal{T}} max(0, 1 - s(e_i, r_j, e_k) + s(e'_i, r_j, e'_k)), \tag{3.8}$$

where e'_i, e'_j are the embeddings of the unobserved negative triples. One advantage of this model is that, since it uses pre-trained embeddings of the textual entity descriptions, the model is not computationally expensive. Also, entity embeddings can be obtained for long-tailed entities in a KG with only textual descriptions available. Furthermore, it is designed to predict missing links in unseen data.

3.7 DISCUSSION AND OUTLOOK

It is evident from the number of research papers discussed in this chapter that KG embeddings for link prediction have gained enormous popularity in recent years. However, certain interesting observations in the text-based and path-based models have been identified and are addressed below:

- *Path Based Models*: **(i)** Both PTransE and PConvKB model focuses on the relational paths between the head and tail entity in the triple, hence discarding the other neighbourhood information of the entities. Also, there is a possibility to have no paths between the head and tail entity of a triple (except for one hop which is the triple itself). Moreover, the paths can be longer containing irrelevant information for the head and tail entities as the contextual information about the entities decreases with increasing hops. **(ii)** As discussed in [73], GAKE is categorized as a homogeneous model as it generates three different graphs based on the different relations. The different kinds of relationships or node types in a KG are also not considered.

- *Text Based Models*: In DKRL, CBOW and CNN approaches are used to produce the embeddings from the entity descriptions. In CBOW approach static word embeddings are used, hence embedding of the word remains the same irrespective of its surrounding context. CNN works better than CBOW as it preserves the sequence of words. However, it is a unidirectional model, therefore encoding tokens only in one direction. Furthermore, CNNs do not learn long-range structure within a sequence like an LSTM. Jointly (ALSTM) extends DKRL by using an attention-based Bi-LSTM model to encode the text. KG-BERT on the other hand treats the triples as sentences and provides them as input to the BERT model.

It follows that investigating the structural contextual information of the entities for the task of link prediction in KGC remains uncharted territory. Additionally although, the contextual NLMs are explored by the KG-BERT model and the BLP framework, their full potential in combination with structural contextual data has yet to be realised.

4

ENTITY TYPE PREDICTION - LITERATURE REVIEW

This chapter offers a categorization of current, cutting-edge state-of-the-art (SoTA) entity typing models based on the methodologies employed, followed by a comprehensive explanation of each model and its shortcomings.

4.1 INTRODUCTION

Information on entity types, which groups together entities with similar attributes (or properties), is one of the fundamental elements of KGs. The types of entities in a KG are usually organised as a hierarchical structure commonly known as *class hierarchy* or *type hierarchy*. As mentioned earlier (see Chapter 1), most KGs are incomplete and lack the type information of the entities. In a KG, type linkages have a special significance and it can take advantage of specialised techniques that take into consideration the unique semantics of these relationships. Therefore, the entity type prediction can be converted to a classification problem [82] with semantic types of the entities as classes. The classification model can be trained to identify the semantic types (or classes) based on other features of the entities in the KG. Entity Typing can be further subdivided into **(i)** *coarse-grained*, and **(ii)***fine-grained*, depending on the level of hierarchy considered in the classification model. In the coarse-grained model, the entity types with a large degree of dissimilarity such as *Person, Location, Organisation* are considered as classes in the classification model. The traditional entity typing models concentrate on a restricted number of coarse-grained types. However, the fine-grained entity typing models assign more specific types of an entity i.e., aims to distinguish subclasses within entry-level classes. The coarse-grained classes are the parent classes of the fine-grained classes in the hierarchy tree. For instance, given a hierarchy tree such that, $Physicist \subset Scientist \subset Person$, for coarse-grained classification the entity *Albert Einstein* will be assigned to the class *Person* and as *Physicist* in fine-grained classification.

Recent years have witnessed intense research on entity type prediction for KGC by leveraging different features of the entities in a KG. Depending on the techniques used, the entity typing models can be broadly classified into **(i)** Heuristic Models, **(ii)** Classical Machine Learning based models and **(iii)** Neural Network based models. The Non-Neural Network-based models can be further divided into heuristic models, and classical Machine Learning (ML) based models, whereas the Neural Network-based models are subdivided into Neural Language Modelling based models, graph structure-based models, and knowledge graph embedding-based models. A brief overview of the aforementioned categories and the corresponding models are provided in Table 4.1. The

Table 4.1: Entity Type Prediction Models and their categories

Heuristic Models	Classical ML based models	Neural Network Based Models	
		Neural Language Model based	Graph structure based models
SDType [93], CUTE [151]	CUTE [151], SLCN [82]	MuLR [155], FIGMENT [154]	APE [60], HMGCN [61], ConnectE [164]

following sections of this chapter include detailed descriptions of each of the SoTA entity typing models mentioned in Table 4.1.

Following the KG definition provided in Section 2.2 of Chapter 2, the same notations are used in this chapter as well. A KG $\mathcal{G}$ consists of a set of triples $\mathcal{T}$, given by, $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{C})$, where $\mathcal{E}$, $\mathcal{R}$, $\mathcal{L}$, $\mathcal{C}$ are the set of entities, relations, literals, and semantic types or classes respectively. In this dissertation, the terms *entity types* and the *classes* in a KG are used interchangeably.

4.2 HEURISTIC BASED ENTITY TYPING MODELS

SDTYPE SDType [93] uses links between the entities as indicators for their corresponding types. The model uses each incoming and outgoing link to and from an entity as a feature to identify the type. For each link, the statistical distribution of types in the subject and object position of the relation for predicting the type of entity is used. Given an entity e with a certain relation r, the conditional property to determine how likely a type c is expressed as $P(c(e)|(\exists r.\top)(e))$, where r is an incoming or an outgoing relation. It forms the building block of SDType and the confidence of an entity e having a type c is computed as:

$$conf(c(e)) = \frac{1}{N} \sum_{\text{all relations } r \text{ of } e} P(c(e)|(\exists r.\top)(e)) \tag{4.1}$$

where N is the number of relations that connects an entity to another one and $c \in \mathcal{C}$ is a type and $\mathcal{C}$ is the set of all types or classes. The problem with the faulty links is addressed as the average probabilities of each type do not contribute much to the overall probability. In the case of a heavily skewed KG containing many general purpose relations such as *owl:sameAs* or *rdfs:label*, there is a huge possibility of overrating the more frequent types. Therefore, a certain weight w_r is assigned to each relation r that determines its capability to predict the type and is given by

$$w_r = \sum_{\text{all types in } \mathcal{C}} (P(c) - P(c|\exists r.\top))^2 \tag{4.2}$$

Therefore, Equation 4.2 can be redefined using the weight value as,

$$conf(c(e)) = \nu.\frac{1}{N} \sum_{\text{all relations r of e}} w_r.P(c(e)|(\exists r.\top)(e)) \tag{4.3}$$

where ν is the normalised factor given by $\nu = \frac{1}{\sum_{\text{all relations r of e}} w_r}$. The SDType model works on the basis of common relations between the entities. Therefore, the model often fails to distinguish between types with a large number of common properties, such as *dbo: BasketballPlayer* and *dbo: VolleyballPlayer* in DBpedia.

CROSS-LINGUAL TYPE INFERENCE (CUTE) CUTE [151] proposes a framework to type Chinese entities with DBpedia types. It is a two-step process in which the cross-lingual entity linking between Chinese and English entities to construct the training data. Then a classification model is proposed to assign the types of Chinese entities. Wikipedia Categories, entity attributes (or properties) of the entities and entity attribute-value pairs are used as features for entity typing. Firstly, it follows the same technique as described in YAGO [129] for determining the *rdfs: subclassOf* relations. Then the probabilistic subClassOf relations are calculated using PARIS [128]. The probability is proportional to the number of entities of category M that belong to type c and is given by,

$$P(M \subseteq c) = \frac{\#(M \cap c)}{\#M} \tag{4.4}$$

where #M is the number of entities belonging to category M and $\#(M \cap c)$ is the number of entities of category M that belong to class c. Finally, the maximum of the probabilities obtained from YAGO and PARIS method is chosen as the final probability. In the second step, an entity e is assigned to a type c using a Noisy-or model [127] and is given by,

$$Pr(e \in c) = 1 - \prod_{m \in M(e)} (1 - Pr(m \subseteq c)) \tag{4.5}$$

where $M(e)$ is the categories that entity e belongs to. If the probability of $Pr(e \in c)$ is greater than or equal to a threshold θ ($0 \leqslant \theta \leqslant 1$), we assign the type c to entity e. A top-down hierarchical classification approach is used to assign the types of entities. Several binary classification models such as Logistic Regression, Random Forest, and SVM are deployed on each node in the hierarchy to find the best-fitting fine-grained type of an entity.

4.3 CLASSICAL MACHINE LEARNING BASED MODEL

SCALABLE LOCAL CLASSIFIER PER NODE (SLCN) SLCN [82] is a modification of the local classifier per node approach used for the task of entity typing. The framework includes a multi-label classifier together with feature selection, instance sampling, and class balancing for each local classifier. SLCN is based on the local classifier per node (LCN) with a top-down prediction approach and siblings' negative examples selection policy. A binary classifier is trained on each type in the hierarchy. For all the types to which the instance is predicted to belong to, the local classifiers of its sub-type predict if the instance belongs to any of its children, and so forth. Whenever the instance is predicted not to belong to a given type, then it is assumed that it does not belong to any of its subtypes either. Feature selection is conducted by computing information gain and top-k relevant features are considered. For the popular classes i.e., the classes with the higher number of entities, the local classifier is trained using a smaller sample size reducing the training time. During sampling, the potential class imbalance is addressed individually for each class. The model uses different local classifiers namely Naive Bayes, J48, Adaboost, and SVM. As mentioned by the authors, one limitation of SLCN is that it assumes independence between sibling nodes, thereby does not support disjoint classes.

4.4 NEURAL NETWORK-BASED MODELS

Following major breakthroughs in deep learning, the use of neural network-based models for entity typing in KGC has significantly increased in recent years. These models are described at great length in this section.

4.4.1 *Models using Neural Language Models*

MULTI-LEVEL REPRESENTATIONS (MULR) MuLR [155] learns multi-level representations of entities via character, word, and entity embeddings followed by the hierarchical multi-label classification. It uses a large corpus in which mentions of the entity e are linked. A set of training examples are used to learn the probability $P(c|e)$ such that entity e has type c. These probabilities are used to assign new types to the entities in the KG as well as to determine types of unknown entities. $P(c|e)$ is modelled as a multi-label classification by training a multilayer perceptron (MLP) with one hidden layer. The output layer has size $|\mathcal{C}|$, i.e., the size equal to the number of types in the KG and it outputs the probability for type $c \in \mathcal{C}$. Mathematically, the probability of type c is given by,

$$[P(c_1|e)...P(c_k|e)] = \sigma(W_{out} f(W_{in}\mathbf{e})) \tag{4.6}$$

where $W_{in} \in \mathbb{R}^{a \times d}$ is the weight matrix from $\mathbf{e} \in \mathbb{R}^{\mathbf{d}}$ to the hidden layer with size a, f is the rectifier function, $W_{out} \in \mathbb{R}^{|\mathcal{C}| \times a}$ is the weight matrix from hidden layer to the output layer of size $|\mathcal{C}|$, and σ is the sigmoid function. A binary cross-entropy loss function is used. The contexts of the entity e as well as its name are used to represent its feature vector on the three levels of entity, word and character. Entities with similar meanings tend to have similar contexts. Thus, the $d-$ dimensional embedding of entity e is learned from a corpus in which all the mentions of e are replaced with a unique identifier. The authors refer to this as Entity Level Representation (ELR). An order-aware embedding model Structured SKIP (SSKIP), which is an extended version of the Skip-gram model is used to learn ELR. The Word Level Representation (WLR) is computed by taking the average of the embeddings of the words that the entity name such that

$$\mathbf{e} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{w_i}, \tag{4.7}$$

where $\mathbf{w_i}$ is the embedding of the i^{th} word of an entity name of length n. Embedding models that consider words as atomic units in the corpus, e.g., SKIP and SSKIP, are word-level. Furthermore, it also exploits the word embedding model FastText [14]. For Character Level Representation (CLR), four models namely Feed-forward network, CNN, LSTM, and BiLSTM are trained on the corpus.

FIGMENT FIGMENT [154] is a text embedding-based entity typing model which is trained on a large annotated corpus in which mentions of the entities are linked to one another. It constitutes of three scoring models namely, *Global* Model (GM), *Context* Model (CM), and *Joint* Model (JM). GM returns a score $S(e, c)$ for an entity-type pair (e, c), where $e \in E$ and $c \in \mathcal{C}$ are the entity and the class respectively. The scoring function $S(e, c)$, is learned using an MLP with one hidden layer and an output layer that contains, for each type $c \in \mathcal{C}$, a logistic regression classifier that predicts the probability of c:

$$S(e, c) = G_c(tanh(W_{input}\mathbf{e})), \tag{4.8}$$

where $W_{input} \in \mathbb{R}^{a \times d}$ is the weight matrix from $\mathbf{e} \in \mathbb{R}^{\mathbf{d}}$ to the hidden layer with size a, G is the logistic regression for type c that is applied on the hidden layer.

GM scores are based on the aggregated contextual information of the entity based on the entire entity-annotated corpus. CM on the other hand, first the individual occurrences of an entity in different contexts are scored separately followed by an aggregation of these scores from each context. Each score based on each context is computed using the MLP as described in Equation 4.8. This score for individual context in the

corpus gives an assessment of how likely it is that an entity e occurring in a certain context has type c. The overall scoring function for the context model is given by,

$$S_{CM}(e, c) = g(U_{e,c}) \tag{4.9}$$

where $U_{e,c} = \{S_{t_1}, S_{t_2}, ..., S_{t_n}\}$, is the set of scores of the type c based on the n contexts $\{t_1, t_2, ..., t_n\}$ of e in the corpus. The function g is a summary function of the distribution of scores, e.g., the mean, median or maximum. The joint model adds the scores of the individual models and is formally defined as

$$S_{JM}(e, c) = S_{GM}(e, c) + S_{CM}(e, c) \tag{4.10}$$

4.4.2 *Models using Graph Structures*

ATTRIBUTED AND PREDICTIVE ENTITY EMBEDDING (APE) APE [60] generates a partially-labelled attributed entity network, including link structure, entity attributes, and type information. Given a KG, $\mathcal{E}^u$ denotes the set of entities without type information, whereas $\mathcal{E}^l$ is the set of entities with type information. The authors define an entity network $\mathcal{G} = (\mathcal{E}, \mathcal{L})$, where $\mathcal{E} = \mathcal{E}^l \cup \mathcal{E}^u$ is the entity set, $|\mathcal{E}| = \mathcal{N}$, $|\mathcal{E}^l| = \mathcal{N}^l$, and $\mathcal{L} = \{(e_i, e_j)|e_i, e_j \in E, i \neq j\}$ is the link relation set. This entity network is defined in form of a matrix $\mathcal{G} = (\mathcal{A}, \mathcal{X}, \mathcal{Y})$, where $\mathcal{A}$ is the adjacency matrix $\mathcal{A} \in \mathbb{R}^{N \times N}$ is equivalent to the link set $\mathcal{L}$ and $\mathcal{Y}$ refers to the entity type vector. If there is link between entities e_i and e_j, $a_{ij} = 1$, otherwise 0. Each row in $\mathcal{A}$ is the *entity link vector* $\mathcal{A}_i$. $\mathcal{X} \in \mathbb{R}^{N \times M}$ is the attribute matrix where M is the number of attributes. It collects attributes of all entities and each row in $\mathcal{X}$ is the attribute vector of $\mathcal{X}_i$. For an entity, $e_i \in \mathcal{E}$, the link vector $\mathcal{A}_i$, and attribute vector $\mathcal{X}_i$ are trained jointly for prediction. An MLP is used to integrate structural and attribute information. The entity type prediction function $P(c|e)$ denoting the probability that entity e belongs to type c, uses $\mathcal{E}^l$ as training data. The predictive type probability is defined through a softmax function in the last layer of the neural network. Formally, the probability that an entity e_i belongs to class C_j is defined as,

$$P(C_j|e_i) = \frac{exp(\mathbf{C_j}.\mathbf{e_i})}{\sum_{j=1}^{k} exp(\mathbf{C_j}.\mathbf{e_i})}, \tag{4.11}$$

where k is the total number of classes. The entity attributes used by APE are words from the entity descriptions, properties, and Wikipedia categories, in the form of an adjacency matrix.

HIERARCHICAL MULTI GRAPH CONVOLUTIONAL NETWORKS (HMGCN) As the name suggests, HMGCN [61] leverages the Graph Convolutional Network (GCN) to predict the missing types of the entities in a KG. The model uses the anchor texts

in textual entity descriptions, Wikipedia Categories, and properties in the KGs. Inspired by the APE model, three undirected entity graphs and their corresponding adjacency matrices are generated to capture the different kinds of semantic correlations between the entities, i.e., entity co-occurrence based $\mathcal{A}_{co}$, category-based graph $\mathcal{A}_{cat}$, and property-based graph $\mathcal{A}_{prop}$. $\mathcal{A}_{co}$ is derived from textual anchor texts to encode the topical relevance between entities. An element $\mathcal{A}_{co}[i,j]$ in the adjacency matrix $\mathcal{A}_{co}$ is set to 1, if an entity e_i occurs in the textual description of another entity e_j and vice-versa. $\mathcal{A}_{cat}$ is constructed through similarity computation based on category information, based on the assumption that entities with similar categories tend to have the same type. Jaccard similarity coefficient is calculated for each element in the matrix $\mathcal{A}_{cat}$ and is given by,

$$\mathcal{A}_{cat}[i,j] = \frac{|Cat(e_i) \cap |Cat(e_j)|}{|Cat(e_i) \cup |Cat(e_j)|}, \tag{4.12}$$

where $Cat(e)$ is the category set of the entity e. Similarly, to determine the correlation between the entities via the properties, Jaccard similarity is also used to generate $\mathcal{A}_{prop}$. These three matrices are then provided into three GCN models. These models use shared parameters in order to leverage the three semantic perspectives. A hierarchical regularization is used to incorporate the *subClassOf* relations between types.

CONNECTE ConnectE [164] is an entity typing approach which includes two embedding-based models, one utilizes the local typing knowledge from existing entity type assertions and the other uses the global triple knowledge from KGs. For the first model *Mapping Entities to Types (E2T)*, a scoring function is designed to measure the similarity between an entity and its type. Inspired by TransSparse [57] model, the entities and entity types are first projected into two different spaces namely entity space and entity type space with an operation matrix $\mathcal{M}$. Next, a similarity measure is computed between this projection and an entity type embedding. The scoring function of E2T given (e, c) is:

$$S = \|\mathcal{M}.\mathbf{e} - \mathbf{c}\|_{l_2}^2, \tag{4.13}$$

where e, and c are the entity and the type respectively, $\mathcal{M} \in \mathbb{R}^{d \times k}$ is the transfer matrix mapping the entities to their corresponding types, d is the dimension of the entity embedding space and k is the dimension of the entity type space. The score is expected to be lower for true entity types and higher for incorrect types. Furthermore, it uses the relational knowledge from the triples in the second model, *TRT*. Based on the fact that the entities cluster effectively according to their types, one of the fundamental assumptions made in this paper is for a true triple <*h*, *r*, *t*>, the corresponding entity types should initially adhere to this connection. Therefore, a new entity type triple is proposed by replacing both head entity and tail entity with their corresponding types

i.e., <*head type, relation, tail type*>. The translational scoring function of TransE [19] is used for the embedding of the types and relations in the latent space. The final ConnectE model is trained by combining the ERT and TRT models. The negative samples are generated by randomly switching type from the entity type pairs in validation and the testing set is generated with the equal number of positive and negative samples. ConnectE exploits entity types to predict the missing types in a KG. Therefore, this model is a supervised approach in which the types of entities have an influence on their corresponding latent representations, as it uses the entity types in both the embedding models.

4.5 DISCUSSION AND OUTLOOK

The literature review conducted in this chapter explicitly states that the research on entity type prediction has gained a boost in recent years. The following are some shortcomings of the existing models that have been identified:

- *Heuristic based Models*: In the CUTE model, **(i)** any misclassification in one of the parent nodes will be carried on to all the consecutive classifiers down the hierarchy tree. **(ii)** The model is computationally expensive for KGs with a large number of classes as a binary classifier is trained on each node of the hierarchy.
- *Models with Graph Structures*: **(i)** Both APE and HMGCN models only take into account the instances of an entity in a particular text, thereby the information in the free text of the textual entity descriptions is disregarded. **(ii)** Based on the co-occurrence of properties between two entities, undirected entity graphs are created in both models, followed by a property matrix. This might result in huge sparse matrices in the case of a skewed KG with billions of entities. **(iii)** The significance of incoming or outgoing edges relative to entities is not considered.
- *Models using NLMs*: **(i)** Both MuLR and FIGMENT models rely on large annotated text corpus containing entity mentions. **(ii)** Despite the fact that both models make use of context information, the representations of the entities they create are static in nature since they use non-contextual word embeddings. **(iii)** These models overlook the enormous amount of information about the entities and their relationships that are accessible in the KGs as well as the textual information such as entity descriptions, etc.

It is to be mentioned that each of the aforementioned models predicts the missing entity types using triples, anchored text, entity labels, Wikipedia Categories, and entity types from the KGs. However, their capabilities are not fully utilized. There is a huge research gap in adequately using the textual data included in the KGs. Also, contextual information related to the entities has been excluded. Additionally, little research has

been done on the underutilized KG features like Wikipedia Categories. In order to fill in the gaps in the literature, this dissertation takes into account the shortcomings of the baseline models in use and suggests several entity type prediction models.

Part III

LINK PREDICTION IN KNOWLEDGE GRAPHS

5

ATTENTIVE MULTIHOP AND ENTITY DESCRIPTIONS FOR LINK PREDICTION

Link prediction is a fundamental task of KGC that aims to estimate the likelihood of the existence of edges (links) based on the current observed structure of a KG [163]. To date many algorithms for generating KG embeddings have been proposed for the task of link prediction and a comprehensive study of the SoTA models and their shortcomings are discussed earlier in Chapter 4. Most of the existing models learn the vectors of the entities and relations from the triple information i.e., only consider one-hop information, whereas very few of them explicitly take into account the relational paths between the head and tail entity of a triple. Additionally, only a handful of them incorporates the textual entity descriptions that are contained in the KGs. It is identified that the structural contextual information of the KGs and contextual NLMs for textual entity descriptions has still not been thoroughly exploited in the General Link Prediction task for KGC in the existing literature. Therefore, this chapter bridges the research gap by proposing a novel KG embedding model [7] for general Link Prediction.

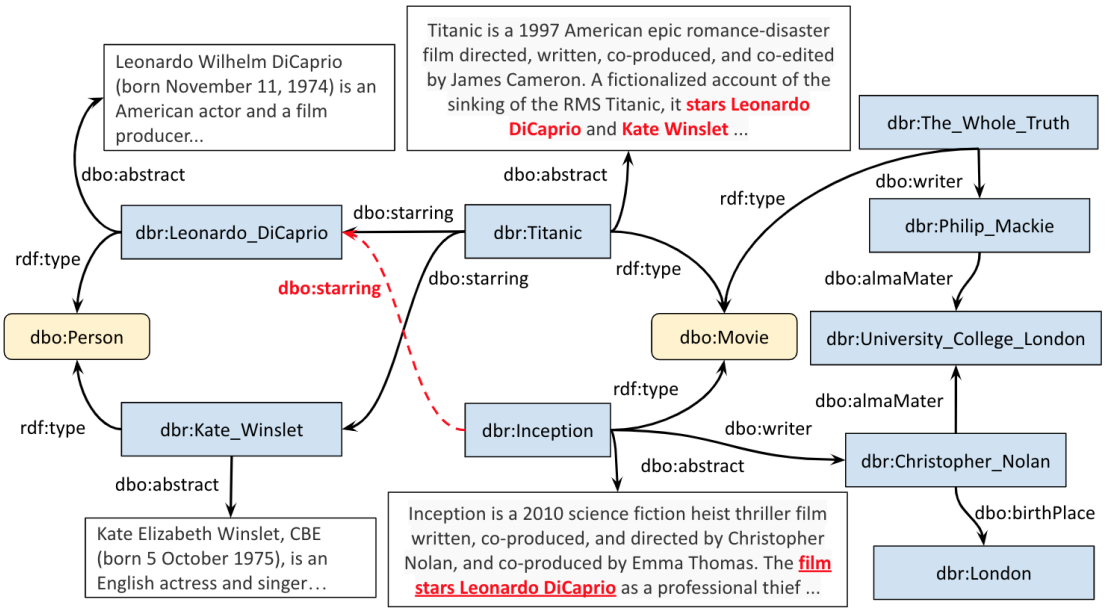
The rest of the chapter is organised as follows. To begin with, a motivation of the work is provided in the Introduction Section 5.1, followed by the problem formulation in Section 5.2. Section 5.3 accommodates the outline of the proposed approach followed by experiments in Section 5.4 and continued by link prediction results in Section 5.5 and that of triple classification in Section 5.6. Finally, Section 5.7 concludes the chapter with a brief discussion on future directions.

5.1 INTRODUCTION

Knowledge Graphs (KGs) have recently gained attention for representing structured knowledge about a particular domain. However, as discussed in Chapter 1 one of the major challenges is that KGs are sparse and often incomplete as the links between the entities are missing. Therefore, predicting the missing links is a vital task of KGC.

The textual description of the entities in the KGs contains rich semantic information and the graph structure provides the contextual information of the entity from the neighbouring nodes. The graph given in Figure 5.1 contains information about some entities from DBpedia [4] and the relations between them. The textual entity description is given by the property *dbo:abstract*. In this graph, *dbr:Leonardo_DiCaprio* and *dbr:Kate_Winslet acted* in the *movie dbr: Titanic* is given by the property *dbo:starring*. However, the *dbo:starring* information is missing for the *movie dbr:Inception*. But this information about *dbo:starring* is available in the textual entity description of the movie

Figure 5.1: An excerpt of KG from DBpedia



dbr:Inception given by the relation *dbo:abstract* which states *'The film stars Leonardo DiCaprio as a professional thief.'* Therefore, the information present in the textual entity description might play a vital role in predicting the missing links.

On the other hand, the movies *dbr:Inception* and *dbr:The_Whole_Truth*, as shown in Figure 5.1, are written by *dbr:Christopher_Nolan* and *dbr:Philip_Mackie* respectively, as given by the relation *dbo:writer*. Also, both these writers have attended the same university *dbr:University _College_London*. Therefore, *dbr: Inception* and *dbr:The_Whole_Truth* are similar in terms of being written by writers who went to the same university. This information from the graph is obtained by 2-hops starting from both source entities *dbr:Inception* and *dbr:The_Whole_Truth* and is given by

$$\textit{dbr:Inception} \xrightarrow{\texttt{dbo:writer}} \textit{dbr:Christopher\_Nolan} \xrightarrow{\texttt{dbo:almaMater}} \textit{dbr:University\_College\_London},$$

and,

$$\textit{dbr:The\_Whole\_Truth} \xrightarrow{\texttt{dbo:writer}} \textit{dbr:Philip\_Mackie} \xrightarrow{\texttt{dbo:almaMater}} \textit{dbr:University\_College\_London}$$

Considering the similarity in the given contextual information, the two movies *dbr:The_Whole_Truth* and *dbr:Inception* should be closer to each other in the vector space. Therefore, graph walks are beneficial for modelling the latent representation of the entities and the relations. However, incorporating the contextual information of an entity from the graph is non-trivial as not all relations are equally important to an entity.

Primarily, link prediction is the task of predicting the head or tail entities in a triple in a KG. However, triple classification, i.e., the task of finding if a given triple is valid

or not in a KG is also considered link prediction as it determines the validity of links between two entities. This work proposes a novel method, MADLINK, which improves the task of link prediction by combining the graph walks and textual entity descriptions to better capture the semantics of entities and relations. The model also incorporates contextual information about the relations in the triples. MADLINK adapts the seq2seq [133] encoder-decoder architecture with an attention layer to obtain a cumulative representation of the paths extracted for each entity from the KG. On the other hand, SBERT [112] has been used to extract the latent representations of the entity descriptions provided as natural language text. DistMult [157] is used as a base model to calculate the score of a triple for head or tail prediction.

The effectiveness of the model is evaluated with the benchmark datasets FB15K [19], FB15K-237 [137], WN18 [19], WN18-RR [34], and YAGO3-10 [34] against different SoTA models with and without entity descriptions. The results show that MADLINK outperforms most of the existing models and achieves comparable results with the rest. The main contributions of this chapter are:

- A path selection approach is introduced by exploiting the importance of a relation w.r.t. an entity in the KG.
- The textual entity descriptions are combined with the contextual information of the entities extracted from the paths for a better representation of entities and relations in KGs.
- An end-to-end attention-based encoder-decoder framework is proposed to generate a better representation of the paths of entities, relations as well as entity descriptions for the link prediction task.

5.2 PROBLEM FORMULATION

Following the definition of KG 3 provided in Chapter 2, a KG $\mathcal{G}$ consists of a set of triples $\mathcal{T}$, given by, $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{C})$, where $\mathcal{E}$, $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{C}$ are the set of entities, relations between the entities, literals, and semantic types of the entities respectively. $< e_h, r, e_t > \in \mathcal{T}$, represents a triple belonging to the set of triples $\mathcal{T}$ in the KG, where $(e_h, e_t) \in \mathcal{E}$ are the head and tail entities, and $r \in \mathcal{R}$ represents relation between them. In this work, we focus on predicting the missing links between entities. Furthermore, most of the KGs comprise textual descriptions for each entity providing semantic information about it. MADLINK aims to learn the latent representation of the entities and relations to a lower dimensional embedding space, $\mathbb{R}^d$, where d is the dimension of the embedding space for the task of link prediction. This section discusses the research questions to address the challenges.

- *RQ1: Does the contextual information of entities and relations in a KG help in the task of link prediction?*

- *RQ2: What is the impact of incorporating textual entity descriptions in a KG for the task of link prediction?*

5.3 MADLINK MODEL

This section comprises a detailed step-wise description of the proposed model and the training approach. The model consists of two parts: **(i)** structural and **(ii)** textual representation. Path selection forms the primary step of the structural representation whereas textual representation is the encoding of the textual entity descriptions.

5.3.1 *Path Selection*

A directed path in a directed labelled graph is a sequence of edges connecting a sequence of distinct vertices. Given a KG $\mathcal{G}$, a path is given by $\left\{e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} ... \xrightarrow{r_m} e_n\right\}$, where $(e_i, r_j), i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., m\}$ are the entities and relations, respectively. Starting from a certain entity, the paths capture the contextual information of an entity in a KG. However, a huge amount of information is stored in the KG and not all triples are equally important for an entity. Some of the triples explain the characteristics of an entity better than others. For example, in Figure 5.1, for the entity *dbr:Christopher_Nolan* the relation *dbo:almaMater* provides more specific information as compared to *dbr:birthPlace* as most of the persons in DBpedia certainly have a birthplace. Also as explained in Section 5.1, the path containing the relation *dbo:almaMater* provides more contextual information for the entities *dbr:Inception* and *dbr:The_Whole_Truth*. Therefore, it is essential to know the general importance of the relations for each entity. Eventually, the paths containing these relations would provide more valuable information compared to the paths without these relations. To tackle this challenge, Predicate Frequency Inverse Triple Frequency (PF-ITF) is used to identify the important relations for each entity [100].

PREDICATE FREQUENCY - INVERSE TRIPLE FREQUENCY (PF-ITF) In order to extract the contextual information related to an entity, paths consisting of l—hops are generated for each node. The properties are selected at each hop of the path using PF-ITF. Also, the cycles present in the KGs are straightened and considered as flat path. Given a KG $\mathcal{G}$, the predicate frequency of outgoing edges is given by $pf_o^e(r, \mathcal{G})$, the inverse triple frequency is given by $itf(r, \mathcal{G})$ and PF-ITF $pf - itf_e(r, \mathcal{G})$ is computed based on Equation 5.1.

$$\begin{aligned} pf_o^e(r, \mathcal{G}) &= \frac{|\varepsilon_o(e)|_{\pi(r)}}{|\varepsilon_o(e)|}, \\ itf(r, \mathcal{G}) &= \log \frac{|\varepsilon|}{|\varepsilon|_{\pi(r)}}, \\ pf - itf_e(r, \mathcal{G}) &= pf_e \times itf, \end{aligned} \tag{5.1}$$

where $\pi(r)$ is the set of relations, $|\varepsilon_o(e)|_{\pi(r)}$ represents the number of outgoing edges from the entity e w.r.t. to the relation r, $|\varepsilon_o(e)|$ is the total number of outgoing edges for the entity e, $|\varepsilon|$ is the total number of triples, and $|\varepsilon|_{\pi(r)}$ is the total number of triples containing the relation r. In this work, the paths are generated starting from a certain entity, so PF-ITF is calculated using the outgoing edges, i.e., Eq. 5.1.

Next, the relations per entity are ranked based on the PF-ITF score. The PF-ITF value increases proportionally with the number of outgoing edges of an entity w.r.t. a relation and is offset by the total number of triples containing the relation which helps to adjust the relations which appear more frequently in general. The highest PF-ITF score of a relation w.r.t. an entity indicates that the triples containing this relation have more information content compared to the rest. Based on the ranks, top-n relations are selected for each entity. Thereafter, paths are generated for the entities in the KG and crawled iteratively until $l - hops$. For computational simplicity, top-m important properties are considered for each entity based on the PF-ITF score.

5.3.2 *Textual Representation*

The textual descriptions of an entity provide semantic information. These descriptions are of variable length and are often short, i.e., less than or equal to 3 words. The textual entity descriptions are encoded into a vector representation. Also, an enormous amount of text data is available outside the KGs which can be leveraged for a better representation of the entities. The static pre-trained language models such as Word2Vec [83], GloVe [94], etc., as well as the contextual embedding model such as BERT [36], have been used in the literature [150, 152, 153] to generate latent representations of Natural Language text. BERT applies transformers which is an attention-based mechanism to learn contextual relations between the words and/or sub-words in a text. The transformer encoder reads the entire sequence of words at once which allows the model to learn the context of a word based on its surroundings.

Sentence-BERT (SBERT) [112] is a modification of BERT which provides more semantically meaningful sentence embeddings using Siamese and triplet networks. The details of the SBERT architecture are provided under Section 2.4.2 in Chapter 2.

Also, [112] shows that the sentence embeddings generated by SBERT outperform BERT for the SentEval toolkit, which is popularly used to evaluate the quality of sentence embeddings. In this work, the sentence embeddings from the pre-trained SBERT model which are fine-tuned with SNLI and STS datasets, are extracted. It follows the

same approach as followed in [112] for SentEval. Therefore, two sentences are not required as input to obtain the sentence embeddings. In this work, the input to the SBERT model is only the entity descriptions. The similar entities in the KG should have similar textual entity descriptions and hence the embedding obtained for the entity descriptions should exhibit similar characteristics. SBERT is designed to minimize the distance between two semantically similar sentences in the embedding space. Therefore, SBERT is leveraged in this work, to obtain similar encoding of the entity descriptions for similar entities. Also, the authors of [112] fine-tune Roberta with the same approach as SBERT and the result shows that the performance of SRoberta and SBERT are almost similar for different tasks and they outperform their respective base models. Furthermore, SBERT outperforms SRoberta in some of the tasks [112]. Also, the model used in this work is the SBERT-SNLI-STS-base model which outperforms the SRoberta-SNLI-STS-base model as shown in [112].

The sentence embeddings obtained from the SBERT model lose the domain-specific knowledge as it is trained and fine-tuned with two different datasets. Therefore, these sentence embeddings generated by SBERT perform better for a wide variety of tasks. In this work, to encode the textual description of the entities in a KG, the default pooling method of the SBERT model, i.e., the MEAN pooling has been used and the entire entity description is considered as one sentence. The fine-tuning of the original BERT model with the textual entity descriptions from both datasets have not been performed because the original BERT model is trained with Wikipedia and these textual entity descriptions are the abstracts of the Wikipedia articles. Therefore, further fine-tuning would have resulted in overfitting. Since SBERT is already fine-tuned with SNLI data, we opted for this model.

5.3.3 *Encoder - Decoder Framework*

In this work, a sequence-to-sequence (seq2seq) learning-based encoder-decoder model [133] is adapted to learn the representation of the path vectors in the KGs, the description vectors as well as the relation vectors. Figure 5.2 depicts the encoder-decoder architecture to generate the path embeddings.

ENCODER The encoder aims at encoding the entire input sequence into a fixed-length vector called a context vector. A path $p_i \in P$, where p_i is a path which is a sequence of entities and the relations between them and is given by, $\left\{ e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} ... \xrightarrow{r_m} e_n \right\}$ is considered as a sentence and the entities e_i and relations r_j are the words. The input to the encoder is the randomly initialized vectors of entities and relations that appear in the paths. These embeddings are passed through a Bi-directional GRU [24] which encapsulates the information for all input elements and compresses them into a con-

Figure 5.2: Encoder - Decoder Framework for paths

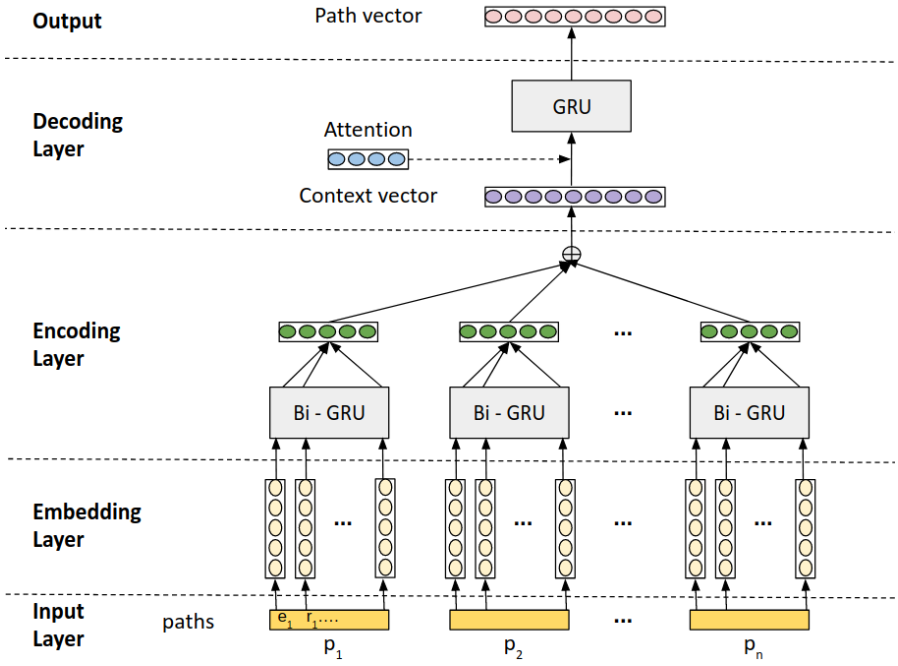
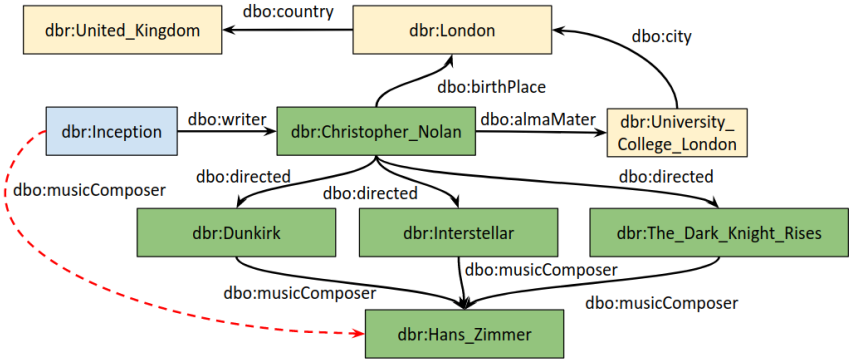


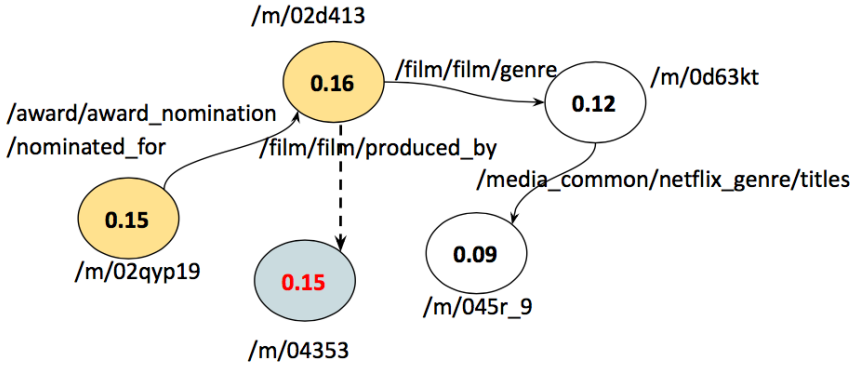
Figure 5.3: Attention for a path in predicting the 'dbo:musicComposer' for the movie Inception



text vector along with the representation of the final hidden states $\mathbf{A} = \mathbf{a_1 a_2 ... a_n}$ where $\mathbf{a_t}$ is given by,

$$\mathbf{a_t} = \mathbf{GRU}(\mathbf{a_{(t-1)}}, \mathbf{embed}(\mathbf{x_t})), \tag{5.2}$$

Figure 5.4: Attention weights for an excerpt from FB15k



where $\mathrm{embed}(x_t)$ is the embedding of entities and relations. In a multi-gated GRU, for each element in the input sequence, the following equations are calculated for each layer.

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{ar}a_{(t-1)} + b_{ar}), \tag{5.3}$$
$$z_t = \sigma(W_{ia}x_t + b_{ia} + W_{ha}a_{(t-1)} + b_{ha}), \tag{5.4}$$
$$n_t = \tanh\left(W_{in}x_t + b_{in} + r_t * (W_{an}a_{(t-1)} + b_{an})\right), \tag{5.5}$$
$$a_t = (1 - z_t) * n_t + z_t * a_{(t-1)}, \tag{5.6}$$

where W_{ir}, W_{ar}, W_{in}, and W_{an} are the weight matrices, x_t is the input at time t, r_t, z_t, n_t are the reset, update and new gates, respectively, and $\sigma, *$ are the sigmoid and Hadamard product, respectively. The context vector and the final hidden state of the encoder model is then passed through an attention layer to learn the weights. Similarly, for relation encoding, instead of the paths the triples containing the relation are considered.

SELF-ATTENTION An attention mechanism allows a neural network to focus on a subset of its inputs or features and is given by,

$$\begin{aligned} \mathbf{attn} &= \mathbf{f_{ffi}}(\mathbf{x}), \\ g &= \mathbf{attn} \odot \mathbf{z}, \end{aligned} \tag{5.7}$$

where $\mathbf{x} \in \mathbb{R}^{\mathbf{d}}$ is an input vector, $\mathbf{z} \in \mathbb{R}^{\mathbf{k}}$ is a feature vector, $\mathbf{attn} \in [\mathbf{0}, \mathbf{1}]^{\mathbf{k}}$ is an attention vector, and $f_\phi(x)$ is an attention network with parameters ϕ[1]. As explained in [40], given an input path sequence $\left\{e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} \ldots \xrightarrow{r_m} e_n\right\}$, not all the relations are

[1] http://akosiorek.github.io/ml/2017/10/14/visual-attention.html

equally important to model a specific fact. Some of them might be important for a certain entity but not for others and vice-versa. PF-ITF helps in identifying the important relations with respect to an entity in the KG. Now, the attention mechanism allows for identifying the important relations and other entities in the paths w.r.t. a certain entity e_h or e_t. Therefore, it is used to generate contextual path encoding. For example, in Figure 5.3, to predict the tail entity of the triple <*dbr:Inception, dbo:musicComposer, t*>, the nodes marked in green would have greater attention than the ones marked in yellow. Therefore, the paths starting from the node *dbr:Inception* which contain the nodes marked in green are impactful in predicting the *dbo: musicComposer* for Inception. Also, for relation encoding, not all triples are equally important for a certain relation. Similarly, for textual encoding not all the words and phrases in the entity descriptions are equally important to represent a certain entity. Hence, an attention mechanism is also used here to generate the contextual description encoding depending on the different words in the text.

As the attention mechanism, the scaled dot product self attention [138] is used because it is much faster and is more space-efficient. Queries and keys of dimension d_k, and values of dimension d_v are given as input. Then the dot product of the query is computed with all keys. Each of them is then divided by $\sqrt{d_k}$. A softmax function gives the weights on the values as an output. Practically, the attention function is computed on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. In this work, the final hidden layer of the encoder is taken as the query as well as the key while the value is the output, i.e., the context vector. The scaled dot product self-attention is given as

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \tag{5.8}$$

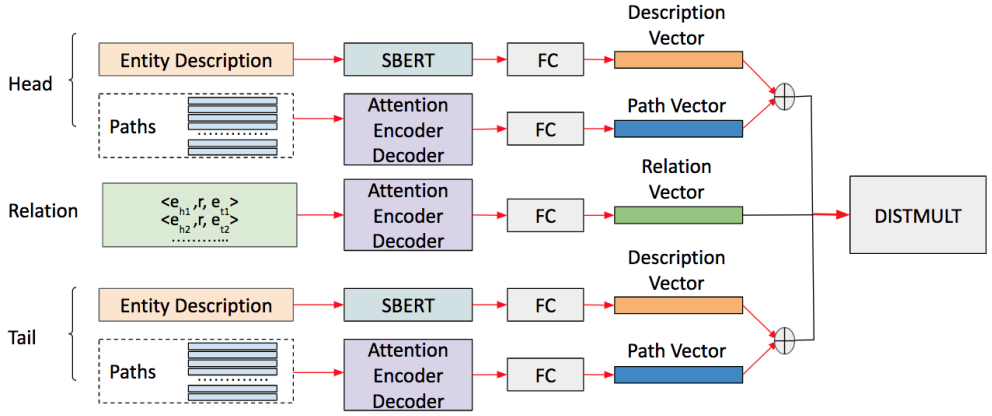
In terms of the MADLINK model, for a given word or relation x the above equation can be rewritten as,

$$\alpha(x) = softmax(\frac{a_t a_t^T}{\sqrt{dim(h_t)}})X, \tag{5.9}$$

where a_t is the hidden layer, $dim(a_t)$ is the dimension of the hidden layer, and X is the context vector. The attention weights of an excerpt from FB15k are illustrated in Figure 5.4. The nodes in yellow have the maximum attention which is required to predict the entity /m/04353.

DECODER The attention layer forms a bridge between the path embeddings and the input path sequences. The decoder network is initialized with the attention weights and the context vector which is then fed to a layer of GRU to obtain the final *Path*

Figure 5.5: Overall Architecture of the MADLINK model



Vector for each entity. Therefore, this Path Vector gives a representation of the entity. Figure 5.2 illustrates the encoder-decoder architecture used in this work.

The main advantage of using this seq2seq-based encoder-decoder in the MADLINK architecture is that it can generate output sequences after seeing the entire input. The attention mechanism allows focusing on specific parts of the input automatically to help generate a useful encoding, even for longer input. Therefore, the proposed MADLINK model looks into all the input paths for a certain entity, focuses on the specific parts of the input, and then generates an encoding for the entity.

5.3.4 *Overall Training*

Given a triple $< e_h, r, e_t >$, the encoding of the head and tail entities are generated by the respective path vectors as discussed in Section 5.3.3. Similarly, for a relation r, all the triples in the KG containing that relation are considered to generate the relation encoding. The textual representation of the entities is obtained from the embeddings of the entity descriptions. The overall architecture of the MADLINK model is depicted in Figure 5.5. Therefore, the parameters of the model are as follows: $\theta = \{\mathbf{D_h}, \mathbf{D_t}, \mathbf{P_h}, \mathbf{P_t}, \mathbf{R}, \mathbf{GRU_1}, \mathbf{GRU_2}\}$, where $\mathbf{D_h}, \mathbf{D_t}$ are the description embedding of the head and tail entities, respectively obtained from SBERT model, $\mathbf{P_h}, \mathbf{P_t}$ are the path embeddings of the head and tail entities, respectively, $\mathbf{R}$ is the relation embeddings, $\mathbf{GRU_1}$ and $\mathbf{GRU_2}$ represent the parameters from the Bi-directional GRU and the decoder GRU, respectively. Finally, the path $(\mathbf{P_h}, \mathbf{P_t})$, relation $(\mathbf{R})$ and the description embeddings $(\mathbf{D_h}, \mathbf{D_t})$ are passed through one fully connected layer with the same weights. The dimension of the SBERT embeddings of textual entity descriptions is 1024 which is reduced to a dimension of 100 or 150 based on the size of the input embedding vector to the DistMult model for different datasets.

Table 5.1: Statistics of the benchmark datasets

Datasets	FB15K	FB15K-237	WN18	WN18RR	YAGO3-10
#Entities	14,951	14,541	40,943	40,943	123,182
#Relations	1,345	237	18	11	37
#Entities with Description	14,515	14,541	40,943	40,943	107,326
#Triples Train set	483,142	272,115	141,441	86,834	1,079,040
#Triples Test Set	59,071	20,466	5,000	3,134	5,000
#Triples Validation Set	50,000	17,535	5,000	3,034	5,000

In this work, DistMult is used as the final scoring function of the model. The model uses a simplification of bilinear interaction between the entities and the relations. DistMult model uses the trilinear dot product as a scoring function

$$f_{DistMult} =< r_p, e_h, e_t >, \tag{5.10}$$

where e_h, e_t, r_p are the embeddings of the head, tail, and relation, respectively. In MADLINK, the D_h and P_h are concatenated and initialized as the head embedding to the scoring function. A similar operation is done with the tail entity as well.

5.4 EXPERIMENTS

This section discusses the benchmark datasets and the experiments conducted for showing the feasibility of MADLINK and its empirical evaluation on two KGC tasks, namely, link prediction, i.e., head or tail prediction and triple classification.

5.4.1 *Datasets*

The statistics of the datasets FB15K, FB15K-237, WN18, WN18RR, and YAGO3-10 used for the purpose of the evaluation are provided in Table 9.2. FB15K is a dataset extracted from large-scale cross-domain KG, Freebase [16]. As mentioned in [34, 137], FB15K has 80.9% test leakage, i.e., a large number of test triples are obtained by inverting the triples of the test set. For example, <*Republic, /government/form_of_government/countries, Paraguay*> is a triple from the training set of FB15K and its inverse <*Paraguay, /location/country/form_of_government, Republic*> is a triple in the test set, where *Republic* and *Paraguay* are the entities in Freebase and */government/form_of_government/countries* is the relation between them. The triple from the test set is the inverse of the triple from the training set. As mentioned by the authors, this might lead to the models for learning relations and their corresponding inverse relations for the link prediction task

instead of modelling the actual KG. Therefore, the FB15K-237 dataset has been introduced by [137], which is a subset of FB15K without the inverse relations. Similarly, WN18 is extracted from WordNet [87] which contains word concepts and lexical relations between the concepts. WN18RR is a subset of WN18 without inverse relations.

YAGO3-10 [34] is extracted from the large-scale cross-domain KG, YAGO [129]. It consists of those entities having at least 10 relations in YAGO. The authors state that the majority of triples in YAGO are the properties of people such as citizenship or gender. The poor performance of the inverse model ConvE [34] on YAGO3-10 implies that it is free from test leakage [116]. Therefore, it is observed, that most of the recent KG embedding models designed for the task of link prediction are evaluated on the FB15k-237 and WN18RR instead of FB15k and WN18. However, the proposed model MADLINK has been evaluated on all the aforementioned 5 benchmark datasets. Besides, the MADLINK model uses textual entity descriptions along with the structural information of entities.

5.4.2 *Experimental Setup*

In the path selection process, the following parameters are used: the number of hops 4, and the number of paths per entity 1000. The hyper-parameters used in the MADLINK model are as follows: the dimension of SBERT vectors 1024, a learning rate of the encoder-decoder framework 0.001, batch size 100, loss margin 1, dropout 0.5. In the pre-processing step of the textual entity description, only punctuation removal is done. The experiments with MADLINK have been performed on an Ubuntu 16.04.5 LTS system with 503GiB RAM. The training with DistMult, SBERT, and the encoder-decoder framework is performed with TITAN X (Pascal) GPU.

5.4.3 *Hyper-parameter Optimization*

The hyper-parameter optimization is performed using grid search as provided in [26] and the hyper-parameters are selected with the best performance on the validation dataset. For all the benchmark datasets, the search space provided in Table 5.2 and the hyperparameters used are provided in Table 5.3. Adam optimizer is used for the base model.

5.4.4 *Link Prediction*

Formally, link Prediction is a sub-task of KGC which aims at predicting the missing head (e_h) or tail entity (e_t) given (e_h, r) or (r, e_t) respectively [20]. Given a KG $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ and $\mathcal{R}$ are the set of entities and relations, link prediction can be defined by a mapping function that assigns a score to every possible triple $(e_i, r, e_j) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$.

Table 5.2: Hyper-parameter Search Space for MADLINK

Hyper-parameters	Range
Batches	{32, 64, 100}
Epochs	{500, 1000}
Embedding Size	{50, 100, 150, 200}
Eta (η)	{1, 5, 10}
Loss	Multiclass NLL
Regularizer Type	{L1, L2, L3}
Regularizer (λ_r)	{1e - 3, 1e - 4}
Optimizer param (lr)	{0.1, 0.01, 0.001}
Optimizer param (λ_o)	{1e - 5, 1e - 4, 1e - 3, 1e -2}

Table 5.3: Optimized hyper-parameters used in the training of MADLINK

Parameters	FB15K	FB15K-237	WN18	WN18RR	YAGO3-10
Batches	64	64	64	64	100
Embedding Size	150	150	150	150	150
Epochs	1000	1000	1000	1000	1000
Learning Rate (lr)	0.001	0.001	0.001	0.001	0.001
Regularizer	L3	L3	L3	L3	L3
Regularizer (λ_r)	1e - 4	1e - 4	1e - 5	1e - 5	1e - 4
Optimizer param (lr)	0.001	0.001	0.01	0.01	0.001
Optimizer param (λ_o)	1e - 3	1e - 3	1e - 3	1e - 3	1e - 3
Loss	Multiclass NLL	Multiclass NLL	Multiclass NLL	Multiclass NLL	Multiclass NLL

A high score of a triple indicates it to be true [43]. However, instead of considering the best score triple, a set of candidates is considered based on the ranking of the scores.

EVALUATION METRICS Following the model in [19] the evaluation metrics used are as follows: (**1**) Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks of the correct entities, and (**2**) Hits@k is the proportion of the correct entities in top-k predictions. Further details on the evaluation metrics are provided in Section 2.7 of Chapter 2. To evaluate most of the embedding models, negative sampling is used to generate corrupted triples by removing either the head or the tail entity. In doing so, some of the generated corrupted triples might actually occur in the KG and should be considered a valid triple. Therefore, all the triples which are true and are present in the training, test, and validation set are removed from the corrupted triples set and are termed as the 'filtered' setting in the evaluation. Also, the triples containing unseen entities are removed from the test and the validation sets.

BASELINES The effectiveness of the proposed model, MADLINK, is illustrated by comparing it with the following baseline models. These baselines are selected based on the diversity of the nature of the embedding models such as translation-based, neural network-based, textual entity description-based, rotational-based, etc.

- TransE [19] is a translation-based embedding model.
- DistMult [157] is bilinear diagonal model.
- ConvE [34] is a CNN-based embedding model.
- ConvKB [88] is also a CNN-based embedding model in which each triple is represented by a 3-column matrix which is then fed to a convolution layer to generate different feature maps. These feature maps are then concatenated into a single feature vector representing the input triple.
- DKRL [150] and Jointly (ALSTM) [153] are textual entity description-based embedding models. The former uses a CNN approach, whereas the latter uses an LSTM.
- RotatE [132] is a model which defines each relation as a rotation from the source to the target entity in the complex vector space. The authors propose a self adversarial negative sampling technique for the model.
- HypER [5] uses a hyper network to generate 1D relation specific vectors convolutional filter weights for each relation which is then used by another network to enable weight sharing across layers.
- R-GCN [118] is a relation aware Graph Convolutional Network, in which the encoder learns the latent representation of the entities and the decoder is a tensor factorization model exploiting these representations for the task of link prediction.
- QuatE [160]. In Quaternion embeddings, hyper complex-valued embeddings with three imaginary components, are utilized to represent entities. Relations are modelled as rotations in the quaternion space.
- MDE [117] (Multiple Distance Embedding model) is a framework to collaboratively combine variant latent distance-based terms by using a limit-based loss and by learning independent embedding vectors. It uses a neural network model that allows the mapping of nonlinear relations between the embedding vectors and the expected output of the score function.
- TuckER [6] model is based on Tucker decomposition of the third-order binary tensor of triples. Tucker decomposition factorizes a tensor into a core tensor multiplied by a matrix along with each mode.

- KG-BERT [158] and Multi-task BERT [66] are the two models which exploit the working principle of the contextual language model BERT to predict missing links in a KG.
- BLP-TransE is one of the models proposed in Inductive Entity Representations from Text via link prediction [32], in which entity representations are generated from their textual entity descriptions using BERT and different KG embedding models such as TransE, DistMult, etc., are used on top of it for the task of link prediction.
- LiteralE [71] is a literal embedding model which uses DistMult, ConvE, and ComplEx as the base model. The main model is based on numeric literal which is easily extendable with text and image literal.
- SSP [148], the Semantic Space The projection (SSP) model jointly learns from the symbolic triples and textual descriptions which uses TransE as the base model. It follows the principle that triple embedding is considered always the main procedure and textual descriptions must interact with triples in order to learn better representation. Therefore, triple embedding is projected onto a semantic subspace such as a hyperplane to allow strong correlation by adopting quadratic constraint.

5.5 LINK PREDICTION - RESULTS

The proposed model MADLINK is compared against the aforementioned baseline models on the 5 benchmark datasets FB15k, FB15K-237, WN18, WN18RR, and YAGO3-10 as depicted in Tables 5.4, 5.5, and 5.6. In these tables, MADLINK^{1} represents the results of the experiments in which the entities without textual entity descriptions are removed, whereas MADLINK^{2} represents the results of the experiments containing all the entities in the datasets. In Tables 5.5, and 5.6, the results marked in bold are the best results and the underlined ones are the second best. '—' is provided in all the 3 tables, if the corresponding results are not available in the respective papers.

5.5.1 *Comparison with textual entity description-based baseline models*

It is to be noted that, out of the above-mentioned baselines, DKRL, Jointly(ALSTM), KG-BERT, Multitask-BERT, and BLP-TransE use textual entity descriptions as to their features for link prediction. Therefore, these models form the primary baseline for our proposed model MADLINK as shown in Table 5.4. For the FB15K-237 dataset, MADLINK^{1} outperforms the SOTA models for all the metrics with an improvement of 8% for MRR and Hits@1, 8.2% for Hits@3, and 7.1% for Hits@10 better than the best baseline model Multitask-BERT. Both DKRL and Jointly(ALSTM) models have the

Table 5.4: Comparison of MADLINK results with the textual entity description-based baseline models on the 5 benchmark datasets

FB15K-237				
Models	MRR	Hits@1	Hits@3	Hits@10
DKRL	0.19	0.11	0.167	0.215
Jointly(ALSTM)	0.21	0.19	0.21	0.258
KG-BERT	0.237	0.144	0.26	0.427
Multitask-BERT	0.267	0.172	0.298	0.458
BLP-TransE	0.195	0.113	0.213	0.363
MADLINK[1]	**0.347**	**0.252**	**0.38**	**0.529**
MADLINK[2]	0.341	0.249	0.377	0.52
WN18RR				
Models	MRR	Hits@1	Hits@3	Hits@10
DKRL	0.112	0.05	0.146	0.288
Jointly(ALSTM)	0.21	0.112	0.156	0.31
KG-BERT	0.219	0.095	0.243	0.497
Multitask-BERT	0.331	0.203	0.383	**0.597**
BLP-TransE	0.285	0.135	0.361	0.580
MADLINK[1]	**0.477**	**0.438**	**0.479**	0.549
MADLINK[2]	0.471	0.43	0.469	0.535
FB15K				
Models	MRR	Hits@1	Hits@3	Hits@10
DKRL	0.311	0.192	0.359	0.548
Jointly(ALSTM)	0.345	0.21	0.412	0.65
SSP	-	-	-	0.771
MADLINK[1]	**0.712**	**0.722**	**0.788**	**0.81**
MADLINK[2]	0.69	0.714	0.78	0.798
WN18				
Models	MRR	Hits@1	Hits@3	Hits@10
DKRL	0.51	0.31	0.542	0.61
Jointly(ALSTM)	0.588	0.388	0.596	0.77
SSP	-	-	-	0.932
MADLINK[1]	**0.95**	**0.898**	**0.911**	**0.96**
MADLINK[2]	0.944	0.88	0.9	0.9
YAGO3-10				
Models	MRR	Hits@1	Hits@3	Hits@10
DKRL	0.19	0.119	0.234	0.321
Jointly(ALSTM)	0.22	0.296	0.331	0.41
MADLINK[1]	**0.538**	**0.457**	**0.580**	**0.68**
MADLINK[2]	0.528	0.447	0.573	0.67

same experimental setup as the MADLINK^2 variant, in which the models are trained on datasets that contain entities without text descriptions. MADLINK^2 variant outperforms DKRL with an increase of 15.1% on MRR, 13.1% on Hits@1, 21% on Hits@3, and 30.5% on Hits@10. Also, the proposed model outperforms the Jointly(ALSTM) model by a 13.1% increase on MRR, 5.9%, 16.7%, and 26.2% on Hits@1, Hits@3, and Hits@10 respectively. For WN18RR, MADLINK outperforms the baseline models with considerable improvement with all the metrics except for Hits@10. Multi-task BERT model performs best for the WN18RR dataset for Hits@10, whereas for other metrics MADLINK outperforms the same model with an improvement of 14.6% on MRR, 23.5% on Hits@1, and 9.6% on Hits@3. Furthermore, both the variants of MADLINK considerably outperform all the baseline models DKRL, Jointly(ALSTM), KG-BERT, and BLP-TransE.

The models KG-BERT, Multitask-BERT, and BLP-TransE have not been evaluated on FB15k, and WN18 due to test leakage problems in the original papers [158]. Therefore, MADLINK is compared against the DKRL and Jointly(ALSTM) models. It is noted that both the MADLINK variants significantly outperform both the text-based baseline models for all the metrics. Similarly, for the YAGO3-10 dataset, both the MADLINK variants show major improvement from both baselines.

It is to be noted that all the aforementioned text-based KG embedding models, such as DKRL, and Jointly (ALSTM), exploit the structural information of the KG in form of triples explicitly together with the textual entity descriptions. However, KG-BERT and Multitask-BERT use the triple information implicitly as the triples are considered as input sentences to the BERT model. On the other hand, in BLP-TransE, the textual entity and relation representations are provided as input to the TransE model in form of triple inputs. Therefore, the results infer that the textual entity descriptions together with the structural information captured via the paths in the MADLINK model capture better semantics of the KG for the task of link prediction.

5.5.2 *Comparison with structure-based baseline models*

The proposed model MADLINK is compared with the baseline KG embedding models such as TransE, DistMult, ConvE, ConvKB, RotatE, HypER, R-GCN, QuatE, MDE, and TuckER, which consider the triple information to generate the latent representation of the entities for the task of link prediction in KGs. The results are shown in Table 5.5 on FB15k-237 and WN18RR and Table 5.6 on FB15k, WN18, and YAGO3-10 datasets.

MADLINK achieves the second-best results after TuckER for FB15k-237 on MRR, Hits@1, and Hits@3 metrics, whereas for Hits@10, TuckER performs slightly better with an improvement of 1.5% over MADLINK. On the other hand, for WN18RR, MADLINK achieves the second-best result for MRR, Hits@1, and Hits@3 after TuckER. The latter performs marginally better than MADLINK with an improvement of 0.7% on MRR, 0.5% on Hits@1, and 0.3% on Hits@3. However, MADLINK performs better

Table 5.5: Comparison of MADLINK results with the structure-based baseline models on FB15k-237 and WN18RR datasets.

FB15K-237				
Models	MRR	Hits@1	Hits@3	Hits@10
TransE	0.31	0.22	0.35	0.5
DistMult	0.247	0.161	0.27	0.426
ConvE	0.26	0.19	0.28	0.38
ConvKB	0.23	0.15	0.25	0.40
RotatE	0.298	0.205	0.328	0.480
HypER	0.341	0.252	0.376	0.520
R-GCN	0.228	0.128	0.25	0.419
QuatE	0.311	0.221	0.342	0.495
MDE	0.344	-	-	0.531
TuckER	**0.358**	**0.266**	**0.394**	**0.544**
MADLINK[1]	0.347	0.252	0.38	0.529
MADLINK[2]	0.341	0.249	0.377	0.52
WN18RR				
Models	MRR	Hits@1	Hits@3	Hits@10
TransE	0.22	0.03	0.37	0.54
DistMult	0.438	0.424	0.442	0.478
ConvE	0.45	0.42	0.47	0.520
ConvKB	0.39	0.33	0.42	0.48
RotatE	0.476	-	-	**0.571**
HypER	0.465	0.436	0.477	0.465
R-GCN	0.39	0.338	0.431	0.49
QuatE	**0.481**	0.436	0.5	0.564
MDE	0.458	-	-	0.56
TuckER	0.47	**0.443**	**0.482**	0.526
MADLINK[1]	0.477	0.438	0.479	0.549
MADLINK[2]	0.471	0.43	0.469	0.535

Table 5.6: Comparison of MADLINK results with the structure-based baseline models on FB15k, WN18, and YAGO3-10 datasets.

FB15K				
Models	MRR	Hits@1	Hits@3	Hits@10
TransE	0.63	0.5	0.73	0.85
DistMult	0.432	0.302	0.498	0.68
ConvE	0.5	0.42	0.52	0.66
ConvKB	0.65	0.55	0.71	0.82
RotatE	**0.797**	-	-	0.884
HypER	0.790	0.734	0.829	0.885
R-GCN	0.69	0.6	0.72	0.8
QuatE	0.77	0.7	0.821	0.878
MDE	0.652	-	-	0.857
TuckER	0.795	**0.741**	**0.833**	**0.892**
MADLINK[1]	0.712	0.722	0.788	0.81
MADLINK[2]	0.69	0.714	0.78	0.798
WN18				
Models	MRR	Hits@1	Hits@3	Hits@10
TransE	0.66	0.44	0.88	0.95
DistMult	0.755	0.615	0.891	0.94
ConvE	0.93	0.91	0.94	0.95
RotatE	0.949	-	-	0.959
HypER	0.951	0.947	**0.955**	0.958
R-GCN	0.71	0.61	0.88	0.932
QuatE	0.949	0.941	0.954	0.96
MDE	0.871	-	-	0.956
TuckER	**0.953**	**0.949**	**0.955**	0.958
MADLINK[1]	0.95	0.898	0.911	**0.96**
MADLINK[2]	0.944	0.88	0.9	0.9
YAGO3-10				
Models	MRR	Hits@1	Hits@3	Hits@10
TransE	0.51	0.41	0.57	0.67
DistMult	0.354	0.262	0.4	0.537
ConvE	0.4	0.33	0.42	0.53
ConvKB	0.3	0.21	0.34	0.5
HypER	0.533	0.455	**0.58**	0.678
R-GCN	0.12	0.06	0.113	0.211
TuckER	0.427	0.331	0.476	**0.609**
MADLINK[1]	**0.538**	**0.457**	**0.58**	**0.68**
MADLINK[2]	0.528	0.447	0.573	0.67

than TuckER for Hits@10 by 2.3%. RotatE performs the best for Hits@10 amongst the mentioned baseline models and its performance is 2.2% better than MADLINK.

TuckER model outperforms all the baseline models as well as MADLINK for FB15k. However, for the YAGO3-10 dataset, MADLINK outperforms the TuckER model with an improvement of 11.1% on MRR, 12.6% on Hits@1, 10.4% on Hits@3, and 7.1% on Hits@10. Additionally, MADLINK outperforms all the other baseline models and achieves SOTA results for the YAGO3-10 dataset over all the metrics. For WN18, TuckER performs better than all the baseline models and MADLINK for all the metrics except for Hits@10.

Additionally, MADLINK outperforms its base model DistMult for all the metrics in all the datasets. For FB15K-237, there is an improvement of 10% in MRR, 9.1%, 11%, and 10.3% in Hits@1, Hit@3, and Hits@10 respectively. Similarly, for FB15K, a considerable increase of 28% in MRR, 42% in Hits@1, 29% in Hits@3, and 13% in Hits@10 have been achieved. On the other hand, for WN18, MADLINK shows a rise of 19.5% in MRR, 28.3% in Hits@1, and 2% for both Hits@3 and Hits@10. Also, for WN18RR similar increment of the results has been achieved with an increment of 3.9% in MRR, 1.4% in Hits@1, 3.7% in Hits@3, and 4.5% in Hits@10. Identical improvement has also been obtained for the YAGO3-10 dataset with an improvement of an average of 17.55% overall the evaluation metrics with an increase of 18.4% in MRR, 19.5% in Hits@1, 18%, and 14.3% in Hits@3, and Hits@10 respectively.

Also, from the results of Hits@k, for all the datasets it can be inferred that MADLINK correctly ranks many true triples in top-k as it achieves SOTA results for FB15K-237, WN18RR, WN18, and YAGO3-10 whereas comparable results for FB15K. Furthermore, MADLINK works better for the datasets without the reverse relations such as FB15K-237 and WN18RR as compared to FB15K and WN18 because in this work directed paths are considered and not undirected edges. However, in this work, the evaluation metric Mean Rank (MR) has not been used because it is sensitive to outliers as mentioned in other related work [89].

MADLINK has also been compared with LiteralE [71], which uses numerical literal to predict the missing links. The results shown in Table 5.7 illustrate that MADLINK performs better than LiteralE. Additionally for FB15k-237, MADLINK performs better than the LiteralE variant with both numeric and text data. It is to be mentioned that, the results of the DistMult variant of LiteralE are considered in Table 5.7 for a fair comparison with MADLINK as both of them use DistMult. The main advantage of MADLINK over the structure-based baseline models is that link prediction can be performed for unpopular entities in a KG, i.e., the entities without any relations or with less number of relations associated with them. This is because MADLINK considers the textual entity descriptions of the entities apart from the structural information. Similarly, since it considers the structural information of the entities in the forms of paths, therefore, missing links can be predicted for entities having triple information but without textual entity descriptions.

Table 5.7: Comparison of MADLINK with LiteralE on FB15k-237, FB15k, and YAGO3-10

FB15K-237				
Models	MRR	Hits@1	Hits@3	Hits@10
LiteralE (Numeric+Text)	0.32	0.234	-	0.488
LiteralE (Numeric)	0.317	0.232	0.348	0.483
MADLINK	**0.347**	**0.252**	**0.38**	**0.529**
FB15k				
Models	MRR	Hits@1	Hits@3	Hits@10
LiteralE (Numeric)	0.676	0.589	0.733	**0.825**
MADLINK	**0.712**	**0.722**	**0.788**	0.81
YAGO3-10				
Models	MRR	Hits@1	Hits@3	Hits@10
LiteralE (Numeric)	0.479	0.4	0.525	0.627
MADLINK	**0.538**	**0.457**	**0.580**	**0.68**

Therefore, it can be concluded that the path information of the entities when coupled with the textual entity descriptions in KGs provide better results in link prediction which is further analysed in Section 5.5.3.

5.5.3 *Ablation Studies*

This section discusses the analysis of different features considered in the MADLINK model for the task of link prediction.

IMPACT OF ONLY TEXTUAL ENTITY DESCRIPTION. The impact of only the textual entity description without the structural information for the task of link prediction has been evaluated along with the triples. The latent representation of the textual entity descriptions is obtained using SBERT vectors. The results as depicted in Table 5.8 show that it outperforms the base model DistMult for all the datasets. It is observed that the improvement for WN18 and WN18RR is very small compared to the other datasets. This is due to the fact that both the datasets contain 5780 entities for which the length of the textual entity description is less than or equal to five providing much

less information. However, Table 5.4 shows textual entity descriptions together with path information exhibit considerable improvement in link prediction.

IMPACT OF ONLY STRUCTURAL INFORMATION. The results depicted in Table 5.9 illustrate the impact of using only the structural information of the KGs in form of paths. The number of paths increases exponentially with the number of hops, for e.g., in FB15k, the average neighbour for each node is 30, therefore, the total number of possible paths of 4 hops would be 810,000. PF-ITF is used to filter out the uncommon relations which in turn reduces the number of paths. As mentioned earlier, 1000 paths with 4 hops are selected for each entity because the relevant contextual information w.r.t. the starting node decreases with more hops. Also, when the walk reaches a dead end, i.e., a node without any outgoing edges, the walk ends in that dead-end node, even if the maximum hops are not reached. However, there could be more than 1000 paths starting from a certain node in which the first 3 hops consist of the same entities and relations. But this does not provide any meaningful insight into the source code. Therefore, amongst the paths, we restrict the paths with the same sequence to a maximum of 30. For example, any path starting with this $e_1 \xrightarrow{r_1} e2 \xrightarrow{r_2} e_3 \xrightarrow{r_2} e_4$ can occur a maximum of 30 times amongst the 1000 paths generated from node e_1. If the number of paths is less than 1000 for any entity, then all paths for that entity are considered.

The results in Table 5.9 show that the MADLINK model with only the structural information outperforms the base model DistMult for all the metrics across all 5 benchmark datasets. Additionally, MADLINK with only structural information works slightly better than MADLINK with only textual entity descriptions for WN18 and WN18RR datasets. Therefore, it can be inferred that the neighbourhood information is well captured for these two datasets in the paths.

INFLUENCE OF ATTENTION IN THE NETWORK To analyse the impact of the attention mechanism in encoding the path vectors, experiments have been conducted without the attention layer as depicted in Table 5.10. The result depicts that there is an improvement in all the evaluation metrics for all the datasets if MADLINK is used with the attention mechanism. Therefore, with an improvement of an average of 5% over Hits@10 for FB15K, WN18, and YAGO3-10 as well as 3.1% for FB15K-237 and 1.4% for WN18RR, it can be seen that the attention mechanism helps in identifying the important entities and relations in a path for the task of link prediction.

5.6 TRIPLE CLASSIFICATION

Triple Classification is the task of determining if a given triple is correct or not. It is a binary classification task, where a given triple (e_h, r, e_t) is to be classified into either 0 (false) or 1 (true) as proposed by [126]. Since all the triples in the training set are true,

Table 5.8: Impact of Textual Entity Descriptions in MADLINK (without path information)

Datasets	Models	MRR	Hits@1	Hits@3	Hits@10
FB15K	DistMult	0.432	0.302	0.498	0.68
	MADLINK	**0.481**	**0.348**	**0.512**	**0.692**
FB15K-237	DistMult	0.2471	0.161	0.271	0.426
	MADLINK	**0.249**	**0.179**	**0.279**	**0.431**
WN18	DistMult	0.755	0.615	0.891	0.94
	MADLINK	**0.758**	**0.618**	**0.895**	**0.943**
WN18RR	DistMult	0.438	0.424	0.442	0.478
	MADLINK	**0.439**	**0.425**	**0.448**	**0.48**
YAGO3-10	DistMult	0.354	0.262	0.4	0.537
	MADLINK	**0.361**	**0.267**	**0.411**	**0.54**

Table 5.9: Impact of Structural Information in MADLINK (without textual entity description)

Datasets	Models	MRR	Hits@1	Hits@3	Hits@10
FB15K	DistMult	0.432	0.302	0.498	0.68
	MADLINK	**0.477**	**0.328**	**0.498**	**0.682**
FB15K-237	DistMult	0.2471	0.161	0.271	0.426
	MADLINK	**0.249**	**0.169**	**0.273**	**0.426**
WN18	DistMult	0.755	0.615	0.891	0.94
	MADLINK	**0.758**	**0.63**	**0.898**	**0.947**
WN18RR	DistMult	0.438	0.424	0.442	0.478
	MADLINK	**0.44**	**0.426**	**0.45**	**0.482**
YAGO3-10	DistMult	0.354	0.262	0.4	0.537
	MADLINK	**0.365**	**0.262**	**0.42**	**0.542**

negative triples are generated for this task, by replacing the head and the tail entities. Also, using this negative sampling method, some of the triples would be generated which would be true. Therefore, all the generated negative triples which are present in the training, test, and validation set are removed. As mentioned by the authors, a threshold ρ_r is set for triple classification maximizing the classification accuracy on the validation set. A triple is considered as positive if the conditional probability, $P(e_t|e_h, r) \geqslant \rho_r$ [126] holds.

However, for MADLINK a Convolutional Neural Network (CNN) binary classifier has been used on top of the embeddings of entities and relations obtained from the model. The classifier is trained with positive triples from the training set and negative

Table 5.10: Impact of Attention Mechanism in MADLINK

Datasets	Models	MRR	Hits@1	Hits@3	Hits@10
FB15K	MADLINK (w/o Attn.)	0.48	0.388	0.502	0.701
	MADLINK (with Attn.)	**0.51**	**0.412**	**0.591**	**0.758**
FB15K-237	MADLINK (w/o Attn.)	0.331	0.211	0.35	0.498
	MADLINK (with Attn.)	**0.347**	**0.252**	**0.38**	**0.529**
WN18	MADLINK (w/o Attn.)	0.92	0.822	0.85	0.91
	MADLINK (with Attn.)	**0.95**	**0.898**	**0.911**	**0.96**
WN18RR	MADLINK (w/o Attn.)	0.412	0.401	0.411	0.509
	MADLINK (with Attn.)	**0.477**	**0.438**	**0.479**	**0.549**
YAGO3-10	MADLINK (w/o Attn.)	0.411	0.331	0.524	0.623
	MADLINK (with Attn.)	**0.461**	**0.372**	**0.580**	**0.68**

Table 5.11: Triple Classification (Accuracy in %)

Models	FB15K	FB15K-237	WN18	WN18RR
TransE	82.9	75.6	87.6	74
DistMult	-	73.9	-	80.4
ConvE	87.3	78.2	95.4	78.3
ConvKB	87.9	80.1	96.4	79.1
Jointly(ALSTM)	91.5	-	97.8	-
PConvKB	89.5	82.1	97.6	80.3
MADLINK	**92.1**	**82.8**	**98**	**81.2**

triples obtained from the negative sampling model. The test set is also complemented with negative examples for proper evaluation. Triple Classification for all 4 datasets has also been compared against all the above-mentioned SOTA models along with PConvKB [59]. PConvKB is an embedding model that incorporates relation paths locally and globally which are then passed through a convolutional neural model. The results are depicted in Table 5.11. The proposed model achieves the SOTA results with

an improvement of 0.2% to 0.8% over the SOTA models for all the benchmark datasets. Furthermore, the accuracy of triple classification for the YAGO3-10 dataset is 80.1% but it has not been provided in Table 5.11 because of the lack of results from the SOTA models.

5.7 CONCLUSION AND OUTLOOK

In this work, a novel approach has been proposed for combining the contextual structural information of an entity from the KGs as well as textual entity descriptions in the embedding space to address the problem of KG completion using link prediction and triple classification. Moreover, an attention-based encoder-decoder approach is introduced to measure the importance of paths. Experimental results show that MADLINK achieves the SOTA results for the textual entity description-based embedding models for the link prediction task on all 5 benchmark datasets. Furthermore, MADLINK outperforms most of the baseline models whereas it achieves comparable results with the rest. The two major research questions are formulated and presented in Section 5.2. The answers to these questions are given as follows:

- *RQ1: Does the contextual information of entities and relations in a KG help in the task of link prediction?*
  - The contextual information of the entities and the relations in a KG are captured by generating paths using random walks. Also, the attention mechanism on the encoder-decoder model helps in identifying the important entities within a path. Handling the path information separately (as shown in Table 5.9) in the MADLINK model yields better results than the base model DistMult which uses only the triple information.
- *RQ2: What is the impact of incorporating textual entity descriptions in a KG for the task of link prediction?*
  - The latent representations of the textual entity descriptions are generated using the SBERT model. The impact of the textual entity descriptions in the link prediction task is dependent on the length of textual information available for the corresponding entities. It can be observed from the results of MADLINK as depicted in Table 5.8 that link prediction works better for FB15k and FB15k-237 compared to WN18 and WN18RR. This is because Freebase entities have detailed and longer text descriptions than WordNet entities. Also, only the textual description-based variant of MADLINK outperforms the base model DistMult for all 5 benchmark datasets. Therefore, the textual entity descriptions play an important role in the task of link prediction in KGs.

The obtained results suggest that the impact of the textual entity description and the contextual structural information is different for different KGs. However, the combination of contextual structural information together with the textual entity descriptions in the MADLINK model outperforms all the text-based KG embedding models.

In future work the following research directions will be considered to further improve the model:

- Explore the translational embedding models such as TransR to learn the initial embeddings of the entities and relations.
- Explore the different scoring functions such as ConvE, translational models, etc. for the base model to analyze the embeddings for the link prediction task.
- Use different multi-hop strategies to generate the context information.
- Multiple text literals available for the entities in the KGs as labels, summaries, comments, etc. can also be incorporated into the model. Also for relations, relation name labels can be considered as the textual description.
- Include explicit external text information such as from Wikipedia into the model.

This chapter consists of a brief analysis of the impact of the text embeddings generated from contextual NLMs for link prediction. However, the effect of NLMs in Natural Language Processing (NLP) based applications is undeniable. KG-BERT [152] paved the way for fine-tuning the contextual NLMs BERT on KGs, which opens up a new research paradigm in link prediction for KGC. This raises the intriguing research question of whether other large-scale contextual NLMs' performance has a comparable influence on predicting the correct triple in a KG. The next chapter of this thesis focuses on addressing the *C1-RQ3* (refer to Section 1.2 of Chapter 1).

6 GPT-2 FOR KNOWLEDGE GRAPH COMPLETION

Neural Language Models (NLMs) have become the backbone of Natural Language Processing (NLP) based applications. Its influence on KGC has been studied in the literature by exploiting the models to generate embeddings for the textual entity descriptions. Static Neural Language models such as Word2vec have been used as a base model for a few node embedding [47] and KG embedding models [113]. However, the usage of contextual NLMs to embed the entities and relations in a KG remains unexplored. One of the pioneers in this area of research is KG-BERT [152], which uses BERT as the base model for KG embedding. Inspired by KG-BERT, this chapter proposes a novel KG embedding model that uses GPT-2 [11] as the base model to predict true triples in a KG.

The remaining sections are outlined as follows: Section 6.1 provides the motivation of this work followed by problem formulation in Section 6.2. Section 6.3 describes the approach, subsequently, experiments are described in Section 6.4. The results are discussed in Section 6.5 and finally the conclusion and future work in Section 6.6.

6.1 INTRODUCTION

Recent years have witnessed extensive research on KGC with a focus on representation learning. Most of these models use structural information i.e., the triple information such as TransE [19], ConvE [33] whereas a few others include textual entity descriptions such as TEKE [142], DKRL [150], etc. However, the models considering the textual information leverage only static word embedding approaches, such as word2vec, GloVe etc. to generate the latent representation of the textual entity descriptions. Consequently, the semantic information encoded in the contextual entity embeddings is not exploited for KGC. A detailed discussion of the literature review is provided in Chapter 3.

On the other hand, pre-trained contextualized Neural Language Models (NLMs) such as BERT [35], and GPT-2 [110], have gained huge momentum in applications of NLP. These models are trained on huge amounts of free text resulting in the encoding of the semantic information leading to a better linguistic representation of the words. GPT-2 is one of the distinguished models which has achieved state-of-the-art results for various language understanding-based tasks. It operates on a transformer decoder architecture with attention masks to predict the next word of a sequence.

However, a combination of contextualized NLMs for the task of KGC is an open research problem. KG-BERT [158] is one of the pioneers in this research in which the

BERT model is fine-tuned on KG data and has been used for link prediction and triple classification as sub-tasks of KGC. The results presented in [158] depict that the information contained in pre-trained NLM plays an important role in predicting the missing links in a KG. Inspired by KG-BERT, a novel GPT-2 based KGC model is explored in this work for the triple classification sub-task. The triples in a KG are considered sentences and the triple classification is considered a sequence classification problem. Furthermore, an analysis of the contextualised NLMs for KGC is also provided.

6.2 PROBLEM FORMULATION

Referring to the KG definition 3 provided in Chapter 2, a KG $\mathcal{G}$ consists of a set of triples $\mathcal{T}$, given by, $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{C})$, where $\mathcal{E}$, $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{C}$ are the set of entities, relations between the entities, literals, and semantic types of the entities respectively. $< e_h, r, e_t > \in \mathcal{T}$, represents a triple belonging to the set of triples $\mathcal{T}$ in the KG, where $(e_h, e_t) \in \mathcal{E}$ are the head and tail entities, and $r \in \mathcal{R}$ represents relation between them. The proposed model aims to learn the latent representation of the entities and relations to a lower dimensional embedding space, $\mathbb{R}^d$, where d is the dimension of the embedding space for the task of triple classification. This section discusses the research question to address the challenges in reference to *C2-RQ1* from Section 1.2 of Chapter 1.

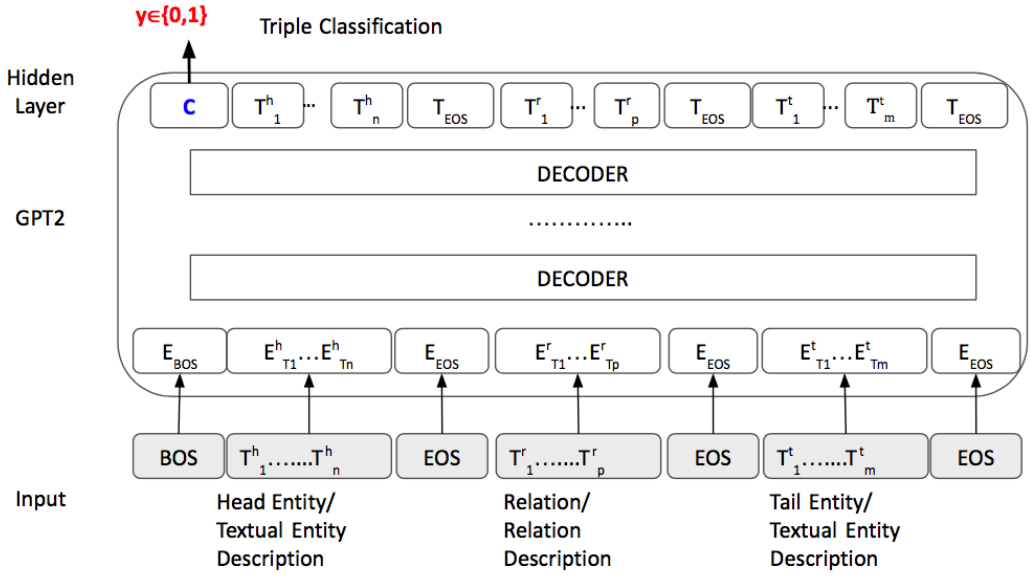
- *C1-RQ3: Can we identify correct triples leveraging contextual NLM?*

6.3 LANGUAGE MODELS FOR KNOWLEDGE GRAPH COMPLETION

This section comprises an analysis of NLMs on KGs followed by a detailed description of the GPT-2 based KGC task. The basic idea of the approach lies in the fact that the contextual NLMs trained on huge corpora also capture relational information present in the training data [98]. Consequently, NLM models can be exploited further to predict the missing links in a KG. However, the impact of the pre-trained contextual NLMs for KGC is still open research.

BERT FOR KNOWLEDGE GRAPH COMPLETION One of the pioneers in this domain is the KG-BERT [158] model in which the pre-trained BERT model is fine-tuned on KGs for KGC. Each triple $< h, r, t >$ is considered a sentence and is provided as an input sentence of the BERT model for fine-tuning. For the entities, KG-BERT has been trained with either the entity names or their textual entity descriptions. The first token of every input sequence is always [CLS], whereas the separator token [SEP] separates the head entity, relation and tail entity. Therefore, each input sequence for the BERT model is given by

Figure 6.1: GPT-2 Architecture for Triple Clasification



([CLS] head entity/description [SEP] relation [SEP] tail entity/description [SEP]). A sigmoid scoring function is introduced on the top of the final layer for the triple classification.

GPT-2 FOR KNOWLEDGE GRAPH COMPLETION Inspired by KG-BERT, GPT-2 [110] is exploited in this work for KGC. GPT-2 is a large transformer-based language model trained on 8 million web pages with 1.5 billion parameters. The model predicts the next word based on all the previous words in the text corpus. An attention mechanism is used to selectively focus on the segments of the input text. The architecture comprises a 12-layer decoder-only transformer, using 12 masked self-attention heads, with 64 dimensional states each. The Adam optimization is used and the learning rate was increased linearly from zero to a maximum of 2.5×10^{-4}. The model was able to outperform the previous NLMs on language tasks like question answering, reading comprehension, summarization, translation, etc. However, the basic difference between BERT and GPT-2 is that BERT uses transformer encoder blocks whereas GPT-2 uses transformer decoder blocks.

Similar to KG-BERT, GPT-2 is also fine-tuned with KG triples where each triple is considered as an input sequence. In this model, two variants have been used to model the input sequence for the fine-tuning task. Given a triple *Albert Einstein, bornIn, Germany*, the input sequence is modelled as

Albert Einstein bornIn Germany [EOS],

Table 6.1: Dataset Statistics

Dataset	#Ent.	#Rel.	#Train	#Val.	#Test
WN11	38,696	11	112,581	2,609	10,544
FB13	75,043	13	316,232	5,908	23,733

[BOS] Albert Einstein [EOS] bornIn [EOS] Germany [EOS],

where [BOS] and [EOS] are the beginning of sequence and end of the sequence respectively. Both entity names and descriptions are considered for the head and tail entities. The input sequences are fed into the GPT-2 model architecture which is a transformer decoder based on the original implementation [110]. It consists of stacked decoder blocks of the transformer architecture and the context vector is initialised with zero for the first word embedding. The masked self-attention is used to extract information from the prior words in the sentence as well as the context word. The word vectors in the first layer of GPT-2 follow byte pair encoding i.e., tokens are parts of words. Furthermore, it compresses the tokenized words list into a set of vocabulary items by considering the most common word components. The GPT-2 sequence classification module is leveraged to determine the plausibility of the triples. Since GPT-2 outputs one token at a time, the classifier is built on the last token. A 2-dimensional vector $\in [0, 1]$ sigmoid scoring function is introduced for triple classification. The architecture of GPT-2 for triple classification is illustrated in Figure 6.1

6.4 EXPERIMENTS

This section comprises an analysis of the initial results obtained on deploying the GPT-2 model on the triple classification task for KGC. The model has been evaluated on two benchmark datasets WN11 and FB13.

6.4.1 *Datasets*

The two benchmark datasets WN11 and FB13 are subsets of WordNet and Freebase KGs respectively and are introduced in [126]. WordNet [86] is a large lexical KG of English comprising nouns, verbs, adjectives and adverbs. They are grouped into sets of cognitive synonyms known as synsets. Each synset expresses a distinct concept. They are interlinked by means of conceptual-semantic and lexical relations. Freebase [17] is a large collaborative KG consisting of structured data captured from various sources including individual, user-submitted wiki contributions. The statistics of the KGs used to fine-tune with GPT-2 followed by the triple classification are provided in Table 6.1.

Table 6.2: Results of Language Models on Triple Classification (accuracy in %)

Model Types	Models	WN11	FB13
KG embeddings with Textual	TEKE	86.1	84.2
Contextual LMs	KG-BERT (labels)	93.5	79.2
	KG-BERT (description)	-	90.4
	Ours with GPT2 (labels)	83	73
	Ours with GPT2 (description)	85	89

6.4.2 *Experimental Setup*

The pre-trained GPT-2 base model with 12 decoder layers, 768 hidden layers, 12 attention heads and 117M parameters is used for fine-tuning. The set of hyperparameters chosen are as follows: batch sizes = $\{256, 128, 32, 8, 1\}$, epochs = $\{5, 3\}$, and learning rate = $\{2e-5, 5e-5\}$. The experiments with GPT-2 have been performed on an Ubuntu 16.04.5 LTS system with 503GB RAM and Tesla V100S GPU.

6.5 RESULTS

The results depicted in Table 6.2 represent some initial results on the triple classification task using the pre-trained GPT-2 model on KGs. Since all the triples in the training set are true, a negative sampling method is used to generate synthetic negative triples for the training of the classifier. The negative triples are generated for this task, by replacing the head and the tail entities with arbitrary entities based on a locally closed world assumption. In this work, filtered settings are used, i.e., if by chance true triples are generated using negative sampling methods, then they are removed. Therefore, the set of triples in the train, test, and validation sets are disjoint.

TEKE [142] and KG-BERT are considered as baseline models as they consider NLMs to model the KGs for KGC. TEKE exploits structural information of the KGs using an embedding layer, a BiLSTM layer followed by mutual attention layer. The results of the baselines are taken from the KG-BERT [158] except for KG-BERT (labels) variant for FB13. The experiment for this variant is performed with the same settings as mentioned in [158]. It is observed from the results that with GPT-2, the model achieves comparable results with the previous models. Also, the results are better for GPT-2 with descriptions variant, this is because the textual entity descriptions have more contextual information resulting in the generation of better representation of triples. The same behaviour has been observed for KG-BERT. Since the NLMs are trained on large corpora, the model parameters contain a huge amount of linguistic knowledge which

Table 6.3: Results with the pre-trained GPT2 model for Triple Classification with different parameter settings

Dataset	Feature	Model details	Precision	Recall	F_1-score
WN11	Labels	batch=128, epoch=10, lr=2e-5	0.76	0.76	0.76
		batch=32, epoch=3, lr=5e-5	0.74	0.74	0.74
		batch=1, epoch=3, lr=5e -5	0.83	0.83	0.83
	Description	batch=8, epoch=5, lr=2e - 5	0.79	0.79	0.79
		batch=1, epoch=3, lr=5e -5	0.85	0.85	0.85
FB13	Labels	batch=32, epoch=10, lr=2e -5	0.69	0.64	0.61
		batch=256, epoch=5, lr=2e -5	0.68	0.68	0.68
	Description	batch=1, epoch=3, lr=5e-5	0.90	0.89	0.89

helps in overcoming the data sparsity problem in KGs. Furthermore, the main advantage of contextual NLM-based KGC methods is that they do not consider the structural information of the entities in a KG. Hence it is independent of any underlying structure in a KG. Furthermore, these models are also applicable to the less popular entities in KGs with a lesser number of triples compared to the others. The task of triple classification in KGC with GPT-2 is similar to the sequence classification task in text and the self-attention mask helps in identifying the important words in the sequences. The variants with labels i.e., the entity names for both KG-BERT and the proposed GPT-2 based model work better for WN11 as compared to FB13. This is because WordNet is a linguistic KG and the NLMs are able to capture more information on the entity names as compared to FB13.

Table 6.3 depicts the precision, recall, and F_1 score of the model with different hyperparameter settings. It is observed that the best results are obtained with batch=1, epoch=3, and lr=5e-5. The changing of epochs does not have much variation in the model whereas batch size has. The lower the batch size, the better the performance of the model.

6.6 CONCLUSION AND OUTLOOK

This work presents an analysis of the effect of exploiting NLMs for KGC. A novel GPT-2 based KGC model has also been proposed. The initial results from the triple classification sub-task show that the semantic information stored in the NLMs can provide vital information for the KGC task. The research question mentioned in Section 6.2 is addressed in this chapter and is given by,

- *C1-RQ3: Can we identify correct triples leveraging contextual NLM?*

– Even though GPT-2 does not outperform KG-BERT, the results obtained provide interesting insights about the model. GPT-2 is a unidirectional decoder model, unlike BERT, hence during training and fine-tuning it only scans for tokens in one direction. However, it is observed that GPT-2 performs well for longer text compared to shorter ones as seen in the triple classification results that were obtained. The suggested GPT-2 variation outperforms the variant using just triples by using textual entity descriptions as input. In BERT and GPT-2, context-specificity presents very differently. It is observed in the literature that in BERT, two words in the same sentence are more dissimilar to each other in the upper layers but are still similar compared to two randomly sampled words. For GPT-2, the words in the same sentence are as dissimilar as randomly chosen words [38]. Therefore, this has an impact on the training of the triples. Furthermore, since it has been trained across millions of websites, there is a potential that it will produce incorrect information [121]

In future, further hyper-parameter tuning to improve model performance and additional experiments on link prediction sub-tasks will be conducted. Also, other contextual existing NLMs are to be explored and analyze the difference in the performances of the models.

Part IV

ENTITY TYPE PREDICTION IN KNOWLEDGE GRAPHS

7

WIKIPEDIA CATEGORY EMBEDDINGS FOR ENTITY TYPING IN KNOWLEDGE GRAPHS

Entity Typing is the task of assigning or inferring the semantic type of an entity in a KG and is an important step of KG construction and completion. Previously, many entity typing approaches in KGs have been proposed which use different features of a KG such as the annotated, anchor text mentions, relations between the entities, Wikipedia categories, etc. Chapter 4 offers a thorough overview of the SoTA entity typing models along with the research gaps. It is observed in the literature that there is still little research done on the distinctive features of Wikipedia categories. This chapter exploits the Wikipedia categories from different aspects and proposes a novel category embedding framework CAT2Type [12] which in return can be leveraged for entity typing.

The remaining chapter is structured as follows: Section 7.1 provides a motivation behind exploiting Wikipedia categories as features. The problem is formulated in Section 7.2. Next, Section 7.3 describes the proposed methodology, followed by experiments and results in Section 7.4 and Section 7.5 respectively. Finally, Section 7.6 provides the conclusion and an outlook of future work.

7.1 INTRODUCTION

Wikipedia categories provide a taxonomic organization of the knowledge underlying DBpedia, YAGO, etc. Due to their rich taxonomic structure as well as the textual information in their labels, Wikipedia categories have been successfully used in tasks such as entity disambiguation [27], Named Entity Recognition in low-resource language [31], etc. Categories are used in Wikipedia to link articles under a common topic. Figure 7.1 shows an excerpt extracted from DBpedia including entities, their types, and Wikipedia Categories. In this example, *dbc:Musicians_from_ New_Jersey*[1] has type as well as the place of birth of the entities contained in that category, i.e., *dbr:Alan_Silvestri*. On the other hand, the entity *dbr:Robert_Zemeckis* has the *rdf:type dbo:Person* which is a coarse-grained type. The fine-grained type of the entity *film_director* is present in its Wikipedia Category *dbc:Science_fiction_film_directors*.

Various studies have been proposed that use the semantics underlying the Wikipedia categories for completing RDF data sources such as DBpedia. Existing approaches use graph-based algorithms [2] for deriving “is a" taxonomy from Wikipedia categories or rule-based algorithms [50, 102, 103] or association rule based approaches [1]. As men-

[1] prefix dbc: <http://dbpedia.org/resource/Category>

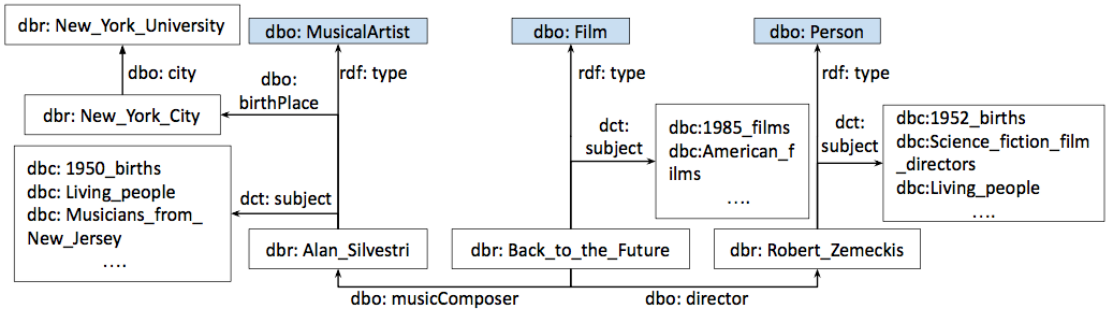


Figure 7.1: Excerpt from DBpedia

tioned earlier in Chapter 1, 307,164 entities in the same DBpedia version are assigned to owl:Thing, which is the most generalised class in the type hierarchy. It is observed that 89.2% of 307,164 untyped entities in DBpedia have at least one Wikipedia Category associated with it. On the other hand, only 2797 entities out of the untyped entities have properties associated with them. Hence, Wikipedia categories provide more information about the long-tailed entities than the properties in DBpedia. Therefore, in contrast to existing approaches, the proposed model *CAT2Type* exploits different characteristic features of the Wikipedia categories, namely **(i)** the textual content in the Wikipedia category labels by leveraging different Neural Language Model (NLMs) and **(ii)** the structural information exploiting the connectivity of the Wikipedia categories based on shared entities between them using network embedding model. There are 1,475,015 categories out of which 1,299,665 categories under the *Main topic classifications* are considered for entities similar to [50]. The evaluation shows that the usage of Wikipedia categories to predict the missing types of entities is not only restricted to DBpedia but can be used for other open KGs as well.

NLMs such as Word2vec [85], BERT [36], etc. have gained huge momentum in different NLP-based applications. These pre-trained NLMs generate generalized representations of textual information without being trained on task-specific corpus and these representations can be used in various downstream tasks while retaining the performance [36, 108]. Therefore, this motivates us in exploiting the NLMs on the Wikipedia category labels for predicting types in the KGs. Furthermore, the reusability of the models also reduces the computational complexity of the proposed CAT2Type model. Moreover, this study includes the construction of the Wikipedia Category network in a novel way for capturing contextual information about the categories. Vector representations from these categories are then generated using a random walk-based node embedding algorithm. Finally, classification is performed using a fully connected neural network for entity-type prediction on top of the feature vectors generated from the aforementioned representations. Several experiments were performed to show the feasibility of the proposed approach on two benchmark datasets DBpedia630k and FIGER [154]. CAT2Type outperforms the state-of-the-art (SoTA) model HMGCN with

an average improvement of 19.3% and 16.7% on $Ma - F_1$ and $Mi - F_1$ respectively on DBpedia630k dataset and 5.4% for $Mi - F_1$ on FIGER. Further experiments show that the proposed model also performs considerably well for unseen data. The main contributions of the chapter are:

- A framework which leverages pre-trained NLMs for learning the representations of entities for entity typing in KGs, and studying its performance with different NLMs. The results provide strong evidence that entity representations based on pre-trained language models exhibit strong generalization and are thus not limited to only NLP tasks.
- A novel category-category network has been constructed to leverage the underlying structure of the categories w.r.t. the shared entities between them. The results strengthen the fact that the category-category network is beneficial to predict types of unseen entities from a different KG.
- A generalized classification framework for both multi-label and multi-class classification is introduced, which can be easily deployed on any entity representations for entity type prediction for any KGs.

7.2 PROBLEM FORMULATION

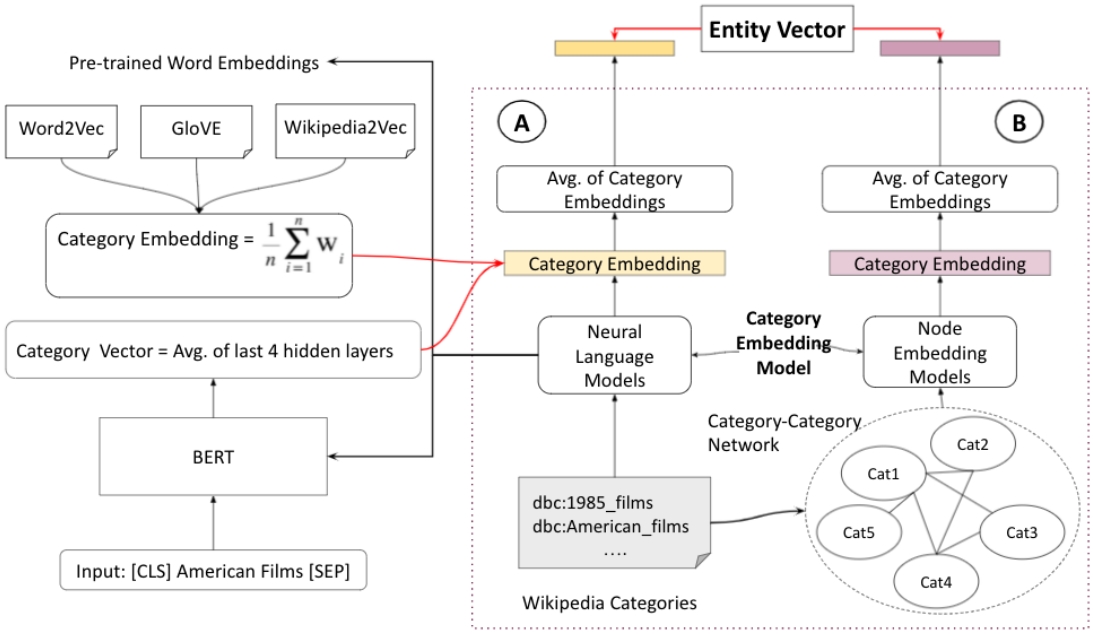
Using the KG definition from Chapter 2 as a reference, a KG $\mathcal{G}$ consists of a set of triples $\mathcal{T}$, given by, $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{C})$, where $\mathcal{E}$, $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{C}$ are the set of entities, relations between the entities, literals, and semantic types or classes of the entities respectively. $< e_h, r, e_t > \in \mathcal{T}$, represents a triple belonging to the set of triples $\mathcal{T}$ in the KG, where $(e_h, e_t) \in \mathcal{E}$ are the head and tail entities, and $r \in \mathcal{R}$ represents relation between them. `rdf:type` is an instance of `rdf:Property` that is used to state that a resource is an instance of a class. A triple of the form: $< e_i$, `rdf:type`, $C_k >$, states that $C_k \in C$, is an instance of `rdfs:Class` and $e_i \in E$ is an entity in G and is an instance of C_k. The proposed model intends to predict the missing types of the entities in a KG addressing the research question mentioned in Section 1.2 of Chapter 1 and is given by,

- *C1-RQ1: Do Wikipedia category labels and the connections between the categories have any impact on entity typing?*

7.3 ENTITY TYPE PREDICTION: CAT2TYPE FRAMEWORK

This section discusses the overall architecture of CAT2Type in detail which exploits the textual information in the Wikipedia Category labels as well as the structural information of Wikipedia categories as illustrated by components Ⓐ and Ⓑ respectively in Figure 7.2.

Figure 7.2: Overall Architecture of the CAT2Type model



7.3.1 *Textual Information in Wikipedia Category Labels*

Wikipedia category labels are comprised of rich semantic information about the characteristic features of the entities which have been used for KG completion tasks [50, 79]. In contrast to the previous work (cf. Chapter 4), in CAT2Type the textual information available in the Wikipedia category labels are leveraged to predict the entity type information. CAT2Type focuses on uncovering the rich semantic information encoded in the Wikipedia category labels (see Section 7.1 for the running example) using different NLM.

NLMs are neural network-based models that learn the distributed representation of words into a continuous low-dimensional vector space. The semantically similar words appear closer to each other in the embedding space. A detailed description of the NLMs is provided in Section 2.4 of Chapter 2.

WORD2VEC & GLOVE Both the models aim at learning the distributed representation of words in a large corpus to a low dimensional vector space. On the other hand, Glove exploits the global word-word co-occurrence statistics in the corpus with the underlying intuition that the ratios of word-word co-occurrence probabilities encode the meaning of the words. In CAT2Type, the Wikipedia category representations of the entities are generated from the Word2Vec model pre-trained on the Google News dataset.

Given a category C of the i^{th} entity C_{e_i} $(w_1, w_2, ..., w_n)$ represent the sequence of n words in the category name, the category representation is given by

$$C_{e_i}^{W2V} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{W}_j, \tag{7.1}$$

where $\mathbf{W}_j$ is the word embedding of the j^{th} word in the category name extracted from the pre-trained Word2Vec model. For Glove, the Wikipedia category representations of the entities are generated from the GloVe model pre-trained on Wikipedia 2014 version and Gigaword 5[2]. Given a category $C_{e_i} = (w_1, w_2, ..., w_n)$, where e_i is the i^{th} entity and $w_1, w_2, .., w_n$ are the n words in the category name, the category representation is given by

$$C_{e_i}^{GloVe} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{W}_j, \tag{7.2}$$

where $\mathbf{W}_j$ is the word embedding of the j^{th} word in the category name extracted from the pre-trained GloVe model.

WIKIPEDIA2VEC It is a skip-gram based LM in which the entities and words from Wikipedia are jointly learned and optimized using three sub-models namely, the Wikipedia link graph model, the Word-based skip-gram model, and the Anchor context model. In CAT2Type, Wikipedia2Vec model pre-trained on English Wikipedia 2018 version[3] is used. Given a category $C_{e_i} = (w_1, w_2, ..., w_n)$, where e_i is the i^{th} entity and $w_1, w_2, .., w_n$ are the n words in the Category name, the category representation is given by

$$C_{e_i}^{Wiki2Vec} = \frac{1}{n}\sum_{j=1}^{n} W_j, \tag{7.3}$$

where W_j is the word embedding of the j^{th} word in the category name extracted from the pre-trained Wikipedia2Vec model.

BERT It is a multi-layer bidirectional Transformer encoder for word representations which takes a sequence of words as input and generates their vector representations. The word representations generated by BERT are sensitive to their respective context in which they appear in the natural language text. The pre-trained BERT model optimized on a huge amount of text allows the model to learn transferable and task-agnostic properties of language [38].

In this work, the entire category name has been used as an input which allows the BERT model to capture the semantics in the category labels based on the sequence

[2] http://nlp.stanford.edu/data/glove.6B.zip [3] https://wikipedia2vec.github.io/wikipedia2vec

of words contained in those labels. Given a category $C_{e_i} = (w_1, w_2, ..., w_n)$, where e_i is the i^{th} entity and $w_1, w_2, .., w_n$ are the n words in the category name, special tokens [CLS] and [SEP] are added by the BERT encoder at the beginning and at the end of the category name respectively. [CLS] is a special classification token that marks the beginning of the input sequence and SEP is used to mark the end of the sequence. In BERT, the decision is that the hidden state of the first token is taken to represent the whole sentence. The input sequence to the BERT model is given by, $([CLS], w_1, w_2, ..., w_n, [SEP])$. The output of the model is a sequence of contextualized embeddings of the tokens in the input sequence together with the added special tokens and is given by, $g(\hat{C_{e_i}}) = (h_{[CLS]}, h_1, h_2, ..., h_n, h_{[SEP]})$, where h_i is the hidden representation of the i^{th} token of the input sequence. For instance, the input sequence to the BERT model for the category *dbc: Science_fiction_film_directors* is *[CLS] Science fiction film directors [SEP]*. The internal representations of words in BERT are a function of the entire input sentence and hence are called contextualized word representations [38]. Therefore, unlike the static word embedding models, the words in the category *dbc: Science_fiction_film_directors* are represented w.r.t to its context in BERT. CAT2Type exploits the feature-based approach of the BERT model in which fixed features are extracted from the BERT model, similar to [36, 80]. In this work, the MEAN pooling is considered as the representation of the input sequence which is the average of the k hidden layers. It is observed to have outperformed the other pooling methods over the hidden layers for several tasks [80]. Therefore, the final category representation generated from the category labels is given by,

$$C_{e_i}^{(BERT)} = \frac{1}{k} \sum_{j=1}^{k} (h_{[CLS]}^j, h_1^j, ...h_n^j, h_{[SEP]}^j), \tag{7.4}$$

where $(h_{[CLS]}^j, h_1^j, ...h_n^j, h_{[SEP]}^j)$ is the representation of the j^{th} hidden state. CAT2Type considers the average of the last 4 hidden layers.

Other pre-trained transformer-based contextual text embedding models [78] as well as static embedding models, can easily be used to generate the representation of the categories. The main advantages of using the pre-trained language models for entity typing are:

- they are computationally inexpensive as the model is pre-computed on huge training data. However, if required the training of the NLMs can be done once and reused for entity typing and other downstream tasks.
- Task-specific classification architecture can be easily deployed on top of this representation.
- The types of entities can be predicted for the entities only with the Wikipedia Category information available for long-tailed or less popular entities in a KG, i.e., the entities with less or no properties in a KG.

- Category Representations can be generated for the new Wikipedia categories without any training of the model.

7.3.2 *Structural Features of Wikipedia Categories*

Besides the Category Name, the connection between the Wikipedia categories w.r.t an entity in a KG provides meaningful semantic information for the task of entity typing. This section provides a description of the proposed category-category network followed by embedding models which learn the representations of categories by preserving the neighbourhood information of the nodes.

CATEGORY - CATEGORY NETWORK CONSTRUCTION To capture the semantic relatedness between the categories, a latent representation of the categories is learned. For this reason, an undirected homogeneous category-category network is constructed in this work, as shown in component (B) in Figure 7.2. Since Wikipedia categories club together similar entities, therefore, categories sharing the same entities are similar to each other. The category-category network is constructed to exploit this connection between the categories based on common entities between them. It is given by $G_{cat} = (V, R)$, where V is the set of nodes and R is the set of edges. The nodes in the network are the Wikipedia categories and there exists an edge R_k between two nodes V_i and V_j, if these two categories share common entities. The categories which do not share any common entities with other categories are not considered in the graph. The weights of the edges between different nodes are crucial due to their significant impact on the embedding model. Therefore, the number of common entities between two categories is the weight of the edge between them.

The advantages of constructing the category-category network instead of using the available category taxonomy are as follows:

- It captures the connection between two categories with a large number of shared entities between them but belonging to different branches in the taxonomy. However, if only taxonomy is considered, there exists no connection between these two categories. Therefore, random walk-based network embedding algorithms such as node2vec fail to capture the similarity between these two categories when trained on the category taxonomy tree, projecting these two categories far apart in their vector space.
- The semantic relatedness between the categories with shared entities is exploited. It helps in capturing the diversity of connectivity patterns observed in the category-category network.
- The problems with numeric values with NLMs [135] can be avoided with the category-category network as the Wikipedia Category labels might contain numerical values.

NODE2VEC The node2vec framework optimizes a neighbourhood preservation objective to learn low-dimensional representations for nodes in a network. The method adapts to different definitions of network neighbourhoods by simulating biased random walks. A detailed description of the model is provided in Section 2.5 of Chapter 2. For the category-category network G_{cat}, firstly the second-order biased random walks are generated given by $\{Cat_1 \rightarrow Cat_2 ... \rightarrow Cat_n\}$. These random walks are then treated as "sentences" for the skip-gram model to generate the embeddings of the categories. Furthermore, the neighbourhoods considered in the random walks are not only restricted to immediate neighbours but also extended to vastly different structures within the network depending on the sampling strategy. Therefore, the connectivity between the categories in the category-category network is exploited to its fullest in learning the latent representation of the categories.

7.3.3 *Entity Type Prediction*

In this work, entity typing is considered a classification problem that takes the entity representation as an input to the classifier and predicts the corresponding type of entity.

ENTITY REPRESENTATION Previously explained approach generates the category representations from the category labels and nodes in the category-category network. The entity representation from the category labels feature is the average of the category representations generated by the respective language models LMs corresponding to that entity. Formally, it is given by,

$$E_i^{Model_{LM}} = \frac{1}{m} \sum_{j=1}^{m} C_j^{LM}, \tag{7.5}$$

where $E_i^{Model_{LM}}$ is the entity representation of the i^{th} entity generated by $Model_{LM}$ and C_j^{LM} represents the Category embedding of the j^{th} category generated by the LMs in Equations 9.1, 7.2, 7.3 and, 7.4, and m is the total number of categories associated with the entity.

Similarly, the average of all category representations generated by the node embedding models is taken as the latent representation of the entities consisting of multiple categories and is given by,

$$E_i^{Model_{NE}} = \frac{1}{m} \sum_{j=1}^{m} C_j^{NE}. \tag{7.6}$$

Here, $E_i^{Model_{NE}}$ is the entity representation of the i^{th} entity generated from the node embedding model Node2Vec, C_j^{NE} represents the Category embedding of the j^{th} cat-

egory of the corresponding entity, and the total number of categories associated with the entity is m. For each entity, separate entity representations are generated from each of the embedding models used in this work.

CLASSIFIERS In CAT2Type, both multi-class and multi-label classifications are used. For multi-class classification, a Fully Connected Neural Network (FCNN) consisting of two dense layers with ReLU as an activation function is deployed on the top of the entity representation. A softmax classifier with a cross-entropy loss function is used in the last layer to calculate the probability of the entities belonging to different classes. Formally it is given by,

$$f(s)_i = \frac{e^{s_i}}{\sum_j^{C_T} e^{s_j}}, \tag{7.7}$$

$$CE_{loss} = -\sum_i^{C_T} t_i \log(f(s)_i), \tag{7.8}$$

where s_j are the scores inferred for each class in C_T given in Equation 8.6. t_i and s_i in Equation 7.8 are the ground truth and the score for each class in C, respectively.

On the other hand, for multi-label classification, a similar FCNN with RELU as an activation function is used for the two dense layers. In multi-label classification, an element can belong to more than one class. Hence, an entity belonging to one class has no impact on the decision of its belonging to another class. Therefore, a sigmoid function with binary cross-entropy loss is used in the last layer which sets up a binary classification problem for each class in C_T. Therefore, the binary cross-entropy loss is,

$$CE_{loss} = -t_i \log(f(s_i)) - (1 - t_i)\log(1 - f(s_i)), \tag{7.9}$$

where s_i and t_i are the score and ground truth for i^{th} class in C_T.

7.4 EXPERIMENTS

This section gives details about the benchmark datasets, experimental setup, and analysis of the results obtained.

7.4.1 *Datasets*

The two benchmark datasets, i.e., FIGER [154] and DBpedia630k [162] are used to evaluate the proposed model. FIGER dataset consists of 201,933 entities with 102 classes from Freebase and DBpedia630k which was originally constructed for text classification consisting of 630,000 entities and 14 non-overlapping classes. Both FIGER and

Table 7.1: Statistics of the datasets

Parameters	DB-1	DB-2	DB-3	FIGER
#Entities	210,000	210,000	210,000	201,933
#Categories	232,112	231,979	231,580	322,654
#Entities train	105,000	105,000	105,000	101,266
#Entities test	63,000	63,000	63,000	60,447
#Entities validation	42,000	42,000	42,000	40,220
#Classes	14	14	14	102

DBpedia630k datasets have been expanded in [61]. However, the extended datasets are not available publicly. Therefore, in this chapter, these datasets are rebuilt by following the description provided in [61]. Furthermore, it is expanded with Wikipedia categories. One of the contributions of this work is to rebuild and extend both datasets and make them publicly available for reusability purposes. In FIGER, out of 201,933 entities 199,111 have a corresponding DBpedia entity via the `owl:sameAs` relation. The entities of DBpedia630k are split equally into three parts DB-1, DB-2, and DB-3, each containing 210,000 entities as in [61]. Furthermore, each split is divided into a training set with 50%, a test set with 30%, and a validation set with 20% of the total entities in each split. It is to be noted that there are no shared entities between the train, test, and validation sets for all the DBpedia split as well as for the FIGER dataset. The statistics of the extended versions of both datasets are provided in Table 7.1 and are available in Zenodo [4] for further research purposes.

7.4.2 *Experimental Setup*

Following the explanation of the language models described in Section 7.3.1, the dimensions of the word vectors generated from the pre-trained Word2Vec and GloVe models are 300. The Wikipedia2vec model used in CAT2Type is pre-trained on Wikipedia 2018 version with window size 10, epochs 10, negative sampling 15, and dimension 300 is used. BERT base figuration *bert-base-uncased*[5] which comprises of 12-layers, 768 hidden layers, 12 attention heads, and 110M parameters is used. Since the average of the hidden layers in BERT is considered as the output of the model in CAT2Type, therefore the dimension of the output vector is also 768. The node2vec model is trained on the category-category network with window size 10, length of the biased walk 10, number of walks per node 100, and 100 epochs. The dimension of the output entity vectors is 300. The FCNN classifier used in CAT2Type has batch size $\{32, 64\}$ and 100

[4] https://zenodo.org/record/7688590 [5] https://huggingface.co/models

epochs. The experiments with CAT2Type are performed on an Ubuntu 16.04.5 LTS system with 503GiB RAM with TITAN X (Pascal) GPU.

7.5 RESULTS

To evaluate the performance of *CAT2Type*, similar to the baseline models [60, 154], Micro-averaged $F_1(Mi-F_1)$ and Macro-averaged $F_1(Ma-F_1)$ metrics are used, details of which are provided in Section 2.7 of Chapter 2. Different variants of *CAT2Type* have been evaluated which serve as an ablation study.

7.5.1 *Results on DBpedia splits*

IMPACT OF NLMS The CAT2Type-BERT variant of the proposed model outperforms all the baseline models with an average improvement of 19.3% and 16.7% on $Ma-F_1$ and $Mi-F_1$ respectively for all the DBpedia splits as illustrated in Table 7.2. The contextual embedding model BERT encodes the semantics of the words differently based on different contexts. Therefore, the input to the model i.e., the entire Wikipedia category name, as discussed in Section 7.3.1 allowed the encoder to better capture the underlying semantics needed to learn more informative category representations. For DB1, the variations of CAT2Type with the other pre-trained NLMs, namely Word2Vec, GloVe, and Wikipedia outperform all the baseline models for the DB1 split and achieve comparable results with the baseline models for the other two splits. The intuitive reasoning behind this difference in the performance of the non-contextual NLMs and BERT are: **(i)** the embeddings generated from non-contextual NLMs are static. The vector representation of the work is always the same regardless of its context. Therefore, these NLMs fail to model polysemous words, i.e., words with multiple meanings. **(ii)** These static NLMs suffer from out-of vocabulary problem [108]. The NLM-based CAT2Type variations are similar to MuLR and FIGMENT which consider word embeddings and entity embeddings as features. However, unlike FIGMENT, no annotated corpus is required in CAT2Type. Furthermore, CAT2Type does not require any additional information about the entities in the KG apart from the Wikipedia category labels.

IMPACT OF NODE EMBEDDINGS *CAT2Type-node2vec* in Table 7.2, which is the variation with the Node2Vec model trained on the category-category network considerably outperforms all the baseline models. It shows an average improvement of 16% on $Ma-F_1$ and 13% on $Mi-F_1$ for all the splits as depicted in Table 7.2. Furthermore, node2vec achieves the second-best result after BERT variants amongst the CAT2Type variants. It can be inferred that in Node2Vec the biased random walks efficiently explore diverse neighbourhoods of a given node.

Table 7.2: Results on DBpedia splits and FIGER

Models	DB1		DB2		DB3		FIGER	
	$Ma-F_1$	$Mi-F_1$	$Ma-F_1$	$Mi-F_1$	$Ma-F_1$	$Mi-F_1$	$Ma-F_1$	$Mi-F_1$
CUTE [151]	0.679	0.702	0.681	0.713	0.685	0.717	0.743	0.782
MuLR [155]	0.748	0.771	0.757	0.784	0.752	0.775	0.776	0.812
FIGMENT [154]	0.740	0.766	0.738	0.765	0.745	0.769	0.785	0.819
APE [60]	0.758	0.784	0.761	0.785	0.760	0.782	0.722	0.756
HMGCN [61]	0.785	0.812	0.794	0.820	0.791	0.817	**0.789**	0.827
CAT2Type-node2vec	0.950	0.948	0.948	0.946	0.948	0.946	0.683	0.84
CAT2Type-word2vec	0.876	0.876	0.723	0.738	0.723	0.742	0.502	0.726
CAT2Type-GloVE	0.883	0.884	0.728	0.742	0.731	0.746	0.501	0.726
CAT2Type-Wikipedia2Vec	0.897	0.897	0.733	0.749	0.739	0.754	0.522	0.737
CAT2Type-BERT	**0.983**	**0.984**	**0.983**	**0.983**	**0.985**	**0.985**	0.764	**0.881**

Table 7.3: Results on DBpedia splits on 7 classes

Models	DB1		DB2		DB3	
	$Ma-F_1$	$Mi-F_1$	$Ma-F_1$	$Mi-F_1$	$Ma-F_1$	$Mi-F_1$
CAT2Type-node2vec	0.972	0.973	0.971	0.972	0.969	0.971
CAT2Type-word2vec	0.917	0.931	0.797	0.812	0.797	0.813
CAT2Type-GloVE	0.938	0.942	0.803	0.816	0.801	0.814
CAT2Type-Wikipedia2Vec	0.955	0.953	0.813	0.822	0.812	0.822
CAT2Type-BERT	**0.98**	**0.99**	**0.98**	**0.99**	**0.989**	**0.99**

IMPACT ON COARSE-GRAINED DBPEDIA TYPES To analyze the impact of the model on coarse-grained entity types in a KG, the types in the DBpedia630k dataset are substituted with 7 coarse-grained types from the DBpedia hierarchy. The types are *dbo: Organisation*, *dbo: Person*, *dbo: MeanOfTransportation*, *dbo: Place*, *dbo: Animal*, *dbo: Plant*, and *dbo: Work*. Here also the BERT variant yielded the best result and node2vec is the second best as depicted in Table 7.3. However, it is interesting to observe that the static NLMs show considerable improvement in their performances. This is due to the fact, the Wikipedia Category labels often have meaning in a broader sense such as *dbc:Swiss_Jews*, which makes it difficult for the static embeddings to obtain fine-grained types of the entities. The performance of the node2vec model strengthens the fact that the underlying structural information in the G_{cat} network provides rich semantic information for better entity representations.

Also to analyze the results on fine-grained types, the classes of the DBpedia630k dataset are substituted with the subclasses resulting in 37 classes from the DBpedia hierarchy. The accuracy of the FCNN classifier on 37 classes, BERT yields the best results with 73.33%, 71.99%, and 91.59% whereas Node2Vec performs the second best with 69.06%, 67.06%, and 87.34% for DB1, DB2, and DB3 respectively.

7.5.2 *Results on FIGER*

IMPACT OF NLMS The FIGER dataset comprises entities belonging to more than one class, hence multi-label classification is used. The CAT2Type-BERT variant outperforms the SoTA model HMGCN with an improvement of 5.4% for $\mathrm{Mi}-\mathrm{F}_1$ and achieves comparable results with HMGCN for $\mathrm{Ma}-\mathrm{F}_1$ as shown in Table 7.2. Also, the performance of other variants with the NLMs for FIGER is low as compared to CAT2Type-BERT. It is observed that the misclassification of most of the entities occurs for the Freebase class *Person* and its subclasses. Most of the entities of class *Person* in FIGER are assigned to generic Wikipedia categories, such as *dbr:Debbie_Millman* is assigned to the only category *dbc:Living_people* which does not provide any information about the fine-grained type of the entity. Also, it contains 15 subclasses of the class *Person* along with the class *Person* itself as fine-grained types of the corresponding entities. Therefore, with such generic categories, the entities are assigned to the coarse-grained type *Person* and are considered as misclassification. However, similar to the DBpedia splits, the contextual embedding model works better on FIGER as well. Therefore, it can be inferred that each of the feature vectors has a significant impact on the type prediction of the entities. However, the class distribution in FIGER is unbalanced with the largest class *City* containing 18,686 entities and the smallest class *engine_device* containing 14 entities and only 5 classes out of 102 classes in FIGER containing more than 11,000 entities and 70 classes have less than 1000 entities. This contributes to the comparatively lesser accuracy in the results with the FIGER dataset for all the CAT2Type variants compared to the DBpedia splits.

IMPACT OF NODE EMBEDDINGS However, it is observed that the node2vec approach with the category-category network works better than the static NLM approaches because node2vec captures the underlying semantics of the categories shared between the entities in the dataset as shown in Table 7.2. Also, the CAT2Type-node2vec approach for FIGER outperforms all the baseline models for the $\mathrm{Mi}-\mathrm{F}_1$ metric and performs second best.

7.5.3 *Results on Unseen Data*

IMPACT OF NODE EMBEDDINGS This work focuses on the reusability of pre-trained models as it uses several pre-trained NLMs for predicting the types of entities in a KG. However, the node2vec model has been trained on the category-category network. The reusability of the pre-trained node embeddings and the robustness of the classification model is analyzed on the *Movie dataset (mdgenre)* [13]. This dataset is a subset of Wikidata in the movie domain, consisting of the movies that are recorded as ever having won or been nominated for an award. The movies are the entities in the KG and their corresponding types are given by genres. The Wikipedia categories of the movies are

Table 7.4: Results on Movie Dataset (Accuracy in %)

Models	mdgenre
RGCN	63
MRGCN	62
CAT2Type-node2vec	66.47

Table 7.5: Results on Unseen DBpedia entities (Accuracy in %)

Models	Unseen DBpedia entities
CAT2Type-Word2Vec	98.64
CAT2Type-GloVe	98.26
CAT2Type-Wikipedia2Vec	98.76

extracted via Wikidata entities. The dataset comprises 5996 entities (805 dev, 2347 test and 2844 train) and 12 classes. The pre-trained node2vec model trained on DBpedia is used to predict the types of movies. The results depicted in Table 7.4 show that the CAT2Type-pre-trained model outperforms the baseline models [13].

The main advantages of learning the category representations using a category-category network are as follows:

- For unseen entities, entity representations can be generated from their corresponding category embeddings to predict their types without training the whole model.
- KGs consist of many more number of entities than the Wikipedia categories. Therefore, the computational complexity of traversing a complete graph with $\frac{n(n-1)}{2}$ edges where n is the number of nodes is very high.

Therefore, the category-category network avoids computational complexity as well as implicitly encodes the entity information via the weights on its edges.

IMPACT OF NLMS For further analysis, 307,164 unseen entities with `rdf:type` information as `owl:Thing` is extracted from DBpedia 2016-10 version, out of which 222,385 entities have `rdf:type` information in the current DBpedia version [6]. Amongst these 222,835 entities, 118,608 entities have a type belonging to the classes of DBpedia630k. The end-to-end CAT2Type model trained on DB1 with the static NLMs namely Word2Vec, GloVe, and Wikipedia2Vec are tested on 118,608 entities and the predicted types are compared against the types of the entities extracted from the current DBpedia version. The results depicted in Table 7.5 show that CAT2Type is robust

[6] https://downloads.dbpedia.org/repo/dbpedia/mappings/instance-types/2021.06.01/

and the type information of the entities can be inferred only from the Category labels of the entities. A major portion of the unseen entities from DBpedia comprises entities from the class *dbo: Album*, whereas the other entities belonging to the classes *dbo: Artist*, *dbo: Film*, *dbo: Athelete*, *dbo: Company*, and *dbo: Building*. Further analysis shows that the test set without the entities from the class *dbo: Album*, also yields an average accuracy of 87%. Therefore, a pre-trained classifier of the CAT2Type model can be efficiently used to predict types of new unseen entities in a KG.

7.6 CONCLUSION AND OUTLOOK

This chapter presents a novel entity typing approach that considers the semantics of the Wikipedia categories to predict the missing types of entities. Two different types of Wikipedia category embeddings are generated for the purpose: **(i)** Text-based embedding from the Wikipedia category labels, and **(ii)** Structure-based embedding from the category-category network. A novel category-category network embedding is proposed by exploiting the connections between the categories with respect to the shared entities. Additionally, the NLMs are also leveraged to generate text-based embeddings. Experimental results show that CAT2Type achieves the SoTA results for entity typing on both benchmarks. Furthermore, the model acquires high accuracy in predicting the missing types of unseen data. As already mentioned in Section 7.1 of this chapter, the majority of the entities which are typed as *owl:Thing* in DBpedia 2016-10 version do not have properties associated with them. Therefore, it is not possible for the existing baseline models that use entity relations as their features, to predict the types of these entities. However, CAT2Type is capable of predicting the missing types of these entities with an accuracy of 98% as shown in Table 7.5. Furthermore, the model also can predict the missing types of entities from any other open KGs in the Linked Open Data (LOD) cloud, which has a connection to Wikipedia articles or Wikipedia categories. This chapter intends to address the research question presented in Section 7.2 and is given by

- *C1-RQ1: Do Wikipedia category labels and the connections between the categories have any impact on entity typing?*
  - The experimental results presented in Table 7.2 show that CAT2Type-BERT and CAT2Type-node2vec variant considerably outperform all the baseline models for all the DBpedia splits, and with Mi-F_1 measure for FIGER. Since CAT2Type does not consider the relations of entities as features, therefore, in contrast to the other embedding-based entity typing models, the proposed pre-trained node embedding model trained on the category network can be leveraged to predict missing types of entities without re-training. No information about the new entity other than the Wikipedia categories is required by this prediction model.

Therefore, the underlying semantics of the Wikipedia categories, which have received little attention in the literature up until now, have been extensively investigated in this work for predicting the missing types of entities.

The analysis of NLMs on Wikipedia category labels is one of the primary focuses of this chapter. The KGs do, however, also include free text descriptions of the entities. Investigating how textual entity descriptions impact the ability to predict the corresponding entity types would be interesting. The neighbourhood information of the entities in a KG is also a crucial aspect that characterizes particular traits of the entities. In order to predict the missing types of entities, it is beneficial to combine the two aforementioned attributes since it aids in understanding the nature of the entities. The subsequent chapter of this thesis, therefore, concentrates on addressing the *C2-RQ2* and *C2-RQ3*(refer to Section 1.2 of Chapter 1).

8

ENTITY TYPE PREDICTION LEVERAGING GRAPH WALKS AND ENTITY DESCRIPTIONS

The previous chapter proposes an entity typing model which considers the Wikipedia categories of the entities which proved to be beneficial for long-tailed entities for which not enough information is available in the KGs. However, in general, a KG contains a huge amount of relation information that defines the characteristic features of the entities. The neighbourhood information of the entities also contributes to their characteristics. Literature review on the existing entity type prediction models presented in Chapter 4 depicts that exploiting the structural contextual entities of the KGs remains unexplored for the task and so are the textual entity descriptions. This chapter proposes a novel entity typing framework [9] leveraging these two aforementioned features.

The remaining portion of the chapter is organized as follows: Section 8.1 gives the motivation behind the proposed approach and highlights the main contribution. Next, Section 8.2 provides a formal definition of the problem addressed in this chapter. Section 8.3 describes the proposed methodology, followed by experiments and results in Section 8.4 and Section 8.5 respectively. Finally, Section 8.6 provides the conclusion and an outlook of future work.

8.1 INTRODUCTION

Entity type information in a KG is one of its atomic building blocks and having this information missing in a KG drastically reduces its usage. It forms the focal point of various Natural Language Processing (NLP) based applications such as question-answering [136], etc. In order to answer questions like *"Is Inception a film or a novel?"*, the entity type information is necessary. Following these lines, this chapter focuses on the problem of entity typing which is the task of assigning or inferring the semantic type of an entity in a KG. Figure 8.1 shows an excerpt from DBpedia where the class *dbo:MusicalArtist*[1] is a subclass of *dbo:Artist* which is a subclass of *dbo:Person*. *dbo:Artist* and *dbo:MusicalArtist*, respectively, are the fine-grained entity types for *dbr:Hans_Zimmer*[2] and *dbo:Artist* is the missing type information. *dbo:Person* is the coarse-grained type.

Recent years have witnessed a few studies on entity typing approaches that are discussed thoroughly in Chapter 4. These models predict entity types using different KG features such as the anchor text mentions in the textual entity descriptions, relations between the entities, entity names, and Wikipedia categories. They learn the repre-

[1] prefix dbo: http://dbpedia.org/ontology/ [2] prefix dbr: http://dbpedia.org/resource/

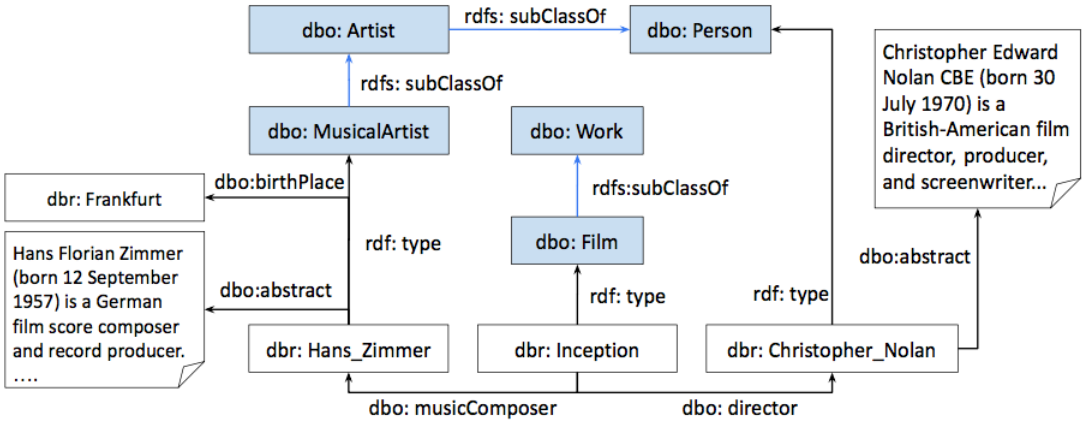


Figure 8.1: Excerpt from DBpedia

sentation of the entities from their KG structure by using translational models [76], GCN-based models [61], neighbourhood-based attention models [165] followed by the correlation between the entities and its types. These models exploit the neighbourhood information only by the entities directly connected, i.e., the triple information of the entities. However, the large amount of contextual information of the entities captured in the graph walks remains unexplored. The work presented in this chapter emphasizes modelling the KG by taking advantage of the semantics of graph walks to predict the entity types with the help of different kinds of walk generation strategies, such as classic random walks, entity walks, and property walks. The paths generated by these graph walk strategies are used within the RDF2vec model [114] to generate different entity representations.

Additionally, the textual entity descriptions in the KGs contain rich semantic information which is beneficial in predicting the missing entity types. For instance, as depicted in Figure 8.1, the textual entity descriptions of the entities clearly mention that *dbr: Christopher_Nolan* is a *director*, *dbr: Hans_Zimmer* is a *music composer*, and *dbr: Inception* is a *film*. Some of the existing baseline models such as MuLR [155] use non-contextual Neural Language Models (NLMs), whereas the other uses GCN model [61] on the words extracted from the entity descriptions. Therefore, to capture the contextual information of the textual entity description contextual NLM is used to generate the representation of the entities.

This chapter presents a framework named **GRAND** (**G**raph Walks for **R**DF2vec **a**nd E**n**tity **D**escriptions), which exploits different variants of the RDF2vec model based on different graph walk strategies together with textual entity descriptions to predict the missing entity types in a KG. In this work, the entity typing problem is modelled as a classification problem. A flat and a hierarchical classification model are deployed on top of the feature vectors generated from the aforementioned entity representations to

predict the missing entity types. The empirical results based on the extensive experiments on two benchmark datasets FIGER [154] and DBpedia630k [162] show that the proposed approach is robust and outperforms the state-of-the-art (SoTA) models. The main contributions of this work are:

- A framework which leverages different graph walk strategies based RDF2vec models and a contextual NLM for textual entity descriptions is proposed to predict the missing entity types.
- A generalized classification framework consisting of three different modules namely multi-class, multi-label, and hierarchical classification is introduced to predict the missing entity types on different levels of granularity. It can be easily deployed for predicting entity types on entity representations from any KGs.
- Extensive experiments are conducted on the benchmark datasets to study the impact of several combinations of entity representations generated from the RDF2Vec variants and the NLM. An analysis of the weights in the classification has been conducted for analyzing which entity representations are suitable in which entity typing situations. Furthermore, the impact of dimensionality reduction of the entity representations on the local and global level using Principle Component Analysis (PCA) is studied.

8.2 PROBLEM FORMULATION

According to the KG definition from Chapter 2, a KG $\mathcal{G}$ consists of a set of triples $\mathcal{T}$, given by, $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{C})$, where $\mathcal{E}$, $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{C}$ are the set of entities, relations between the entities, literals, and semantic types of the entities respectively. $< e_h, r, e_t > \in \mathcal{T}$, represents a triple belonging to the set of triples $\mathcal{T}$ in the KG, where $(e_h, e_t) \in \mathcal{E}$ are the head and tail entities, and $r \in \mathcal{R}$ represents relation between them. `rdf:type` is an instance of `rdf:Property` that is used to state that a resource is an instance of a class. A triple of the form: $< e_i, \texttt{rdf:type}, C_k >$, states that $C_k \in C$, is an instance of `rdfs:Class` and $e_i \in E$ is an entity in G and is an instance of C_k. The proposed model intends to predict the missing types of the entities in a KG addressing the research question mentioned in Section 1.2 of Chapter 1 and is given by,

- *C2-RQ2: What is the impact of textual entity descriptions in predicting the corresponding missing types?*
- *C2-RQ3: Are strategic graph walks beneficial for entity typing?*

8.3 ENTITY TYPE PREDICTION: GRAND FRAMEWORK

An overview of the GRAND framework is illustrated in Figure 8.2. Component (A) represents the RDF2vec variants that use the different strategies for generating graph

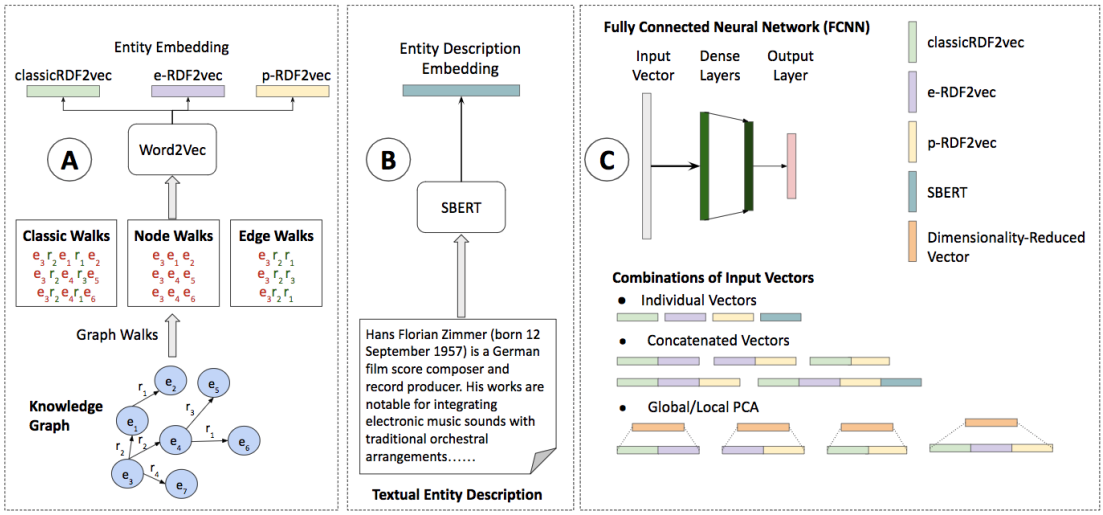


Figure 8.2: Architecture of the GRAND framework

walks, i.e., classic walks, node walks, and property walks. These walks are then given as input to the language models separately and finally, variants of entity representations are generated. Component (B) generates the representations of the entities from the textual entity description by using SBERT. Finally, component (C) shows combinations of the variants of entity representations used for flat as well as hierarchical classification. In the rest of this section, each of these aforementioned components of the framework is explained in more detail.

8.3.1 *Entity Embeddings from Strategic Graph Walks*

RDF2vec [114] is one of the first approaches to adopt statistical language modelling techniques to KGs. Similar approaches, such as *node2vec* [47] and *DeepWalk* [95], are proposed for unlabeled graphs while KGs are labelled by nature, i.e., they contain different types of relations. The key idea of RDF2vec is a two-step approach: first, random walks over the graph are executed, thereby collecting sequences of entities and relations. To employ language modelling techniques, these sequences are then considered as sentences where each entity and relation in the sequence are treated as words. In RDF2vec, those sentences are then processed by word2vec [84, 85], where both variants of word2vec, i.e., a continuous bag of words (CBOW) and skip-gram (SG), are possible.

One limitation of the word2vec algorithm is that it is not aware of the word order. For instance, for a window size of 4, the sentences "John ate a pizza" and "pizza ate a John" are equivalent. This is also the case with *RDF2vec*: For instance, the statements `<Severus> <loves> <Lily>` and `<Lily> <loves> <Severus>`, are considered equiva-

lent even though <loves> is not a symmetric property. To overcome this limitation, an order-aware version of RDF2vec has been proposed [106] which has shown improved performance on multiple machine learning datasets. This order-aware variant of RDF2vec uses a *structured word2vec* model [77] which incorporates the positional information of the words in a sentence. The main advantage of the order-aware RDF2vec model over the classical RDF2vec model is that it respects the positional information of the entities and relations in the random walks, thereby learning embeddings which are better in terms of type separation.

Another type of RDF2vec extension is to explore different strategies for performing graph walks. These strategies have been explored using an either variant of random walks (e.g., community hops [65], walklets [96], or hierarchical walks [119]), or by combining different random walk strategies, as the *ontowalk2vec* approach, which combines RDF2vec and node2vec walks [44]. In this chapter, the aforementioned order-aware, as well as different RDF2vec graph walk strategies [107], are leveraged to predict the missing types of entities.

GRAPH WALK GENERATION STRATEGIES RDF2vec combines the notion of similarity and relatedness. This can be easily observed when printing the most related concepts for "Berlin" on DBpedia via KGvec2go [104], i.e., many people who are *related* to the city are identified as politicians. However, those are not really *similar* – they do not share properties with Berlin (which is a city rather than a living being). This leads to further exploration of RDF2vec for entity typing.

In this chapter, six different RDF2vec configurations are presented and evaluated – stand-alone as well as combinations. For the task of entity typing, three different walk generation strategies are applied: (1) classic walks, (2) entity walks, and (3) predicate walks. Each strategy is explained below in more detail.

CLASSIC WALKS. The originally presented RDF2vec variant generates multiple random walks for each node in the graph. A random walk of length n (where n is an even number) is of the form

$$w = (w_{-\frac{n}{2}}, w_{-\frac{n}{2}+1}, ..., w_0, ..., w_{\frac{n}{2}-1}, w_{\frac{n}{2}}) \tag{8.1}$$

where $w_i \in \mathcal{V}$ if i is even, and $w_i \in \mathcal{R}$ if i is odd. For better readability, we stylize $w_i \in \mathcal{V}$ as e_i and $w_i \in \mathcal{R}$ as p_i:

$$w = (e_{-\frac{n}{2}}, p_{-\frac{n}{2}+1}, ..., e_0, ..., p_{\frac{n}{2}-1}, e_{\frac{n}{2}}) \tag{8.2}$$

ENTITY WALKS (E-RDF2VEC). An entity walk contains only entities without any other properties. Such an approach is also known as *e-RDF2vec*. It has the form:

$$w_e = (e_{-\frac{n}{2}}, e_{-\frac{n}{2}+2}, ..., e_0, ..., e_{\frac{n}{2}-2}, e_{\frac{n}{2}}) \tag{8.3}$$

For an entity walk, all elements are entities, i.e., $w_{n_i} \in \mathcal{V}$.

PREDICATE WALKS (P-RDF2VEC). A predicate walk contains only one entity together with object properties. Such an approach is also known as *p-RDF2vec*. It has the form:

$$w_p = (p_{-\frac{n}{2}+1}, p_{-\frac{n}{2}+3}, ..., e_0, ..., p_{\frac{n}{2}-3}, p_{\frac{n}{2}-1}) \tag{8.4}$$

The different walk strategies are visualized in component (A) in Figure 8.1.

GENERATING ENTITY EMBEDDINGS USING RDF2VEC VARIANTS. An embedding model is trained for each set of walks using word2vec [84, 85] and position-aware word2vec [77] (suffix oa in the following) which yields six sets of embeddings: (1) Classic RDF2vec, (2) e-RDF2vec, (3) p-RDF2vec, (4) Classic RDF2vec$_{oa}$, (5) e-RDF2vec$_{oa}$, and (6) p-RDF2vec$_{oa}$. The proposed model, GRAND, is evaluated by using the configurations presented in 8.3.1 on their own as well as in a fused way. Concerning the fusion of vectors, three modes are employed: (1) Vector concatenation, (2) Local PCA (LPCA), and (3) Global PCA (GPCA). PCA is a technique for reducing the dimensionality of the vectors with minimal loss in encoded information. It is used for the identification of a smaller number of uncorrelated variables known as principal components. The difference between (2) and (3) is that in the case of the LPCA, a principal component analysis is only performed for the subset of vectors that appear in the datasets (see Section 8.4) whereas, for the GPCA, one all vectors generated from the KG using RDF2vec variants are considered. Each of these configurations can be used as a vector within GRAND (see component (C) in Figure 8.2).

The main advantages of using different RDF2vec variants are:

- With a growing length of walks and training window, they can take advantage of large entity context ranges by effectively treating every entity as being connected to all the others in the graph – this is in contrast to the baseline models which are based on local aggregation, i.e. they learn the representation of each entity based on its adjacent entities in the KG [61, 165].
- The graph walk strategies are effective, robust, and equitable, i.e., all relations and nodes are given equal importance in generating the embeddings.
- The walk strategies put emphasis on certain semantic aspects – namely *relatedness* and *similarity* [107].
- RDF2vec is a very scalable embedding algorithm which can be easily used for large graphs such as DBpedia.
- Experimental results show that RDF2vec performs better than other KG embedding models for the class separation task as explained in [166]. The separability

task aims at measuring if embeddings from different classes can be linearly separable and in [166] the evaluation is done on 10,000 pairs of classes from DBpedia.

- Any classification algorithm can be deployed on top of entity embeddings to predict the missing types. Furthermore, the models are precomputed and can, therefore, also be used for further downstream tasks.
- The PCA variants help in reducing the correlated features from different embedding configurations as well as in reducing the chance of overfitting.

8.3.2 *Entity Description Representation*

The textual descriptions of an entity provide rich semantic information. Sentence-BERT (SBERT) [112] fine-tunes the BERT [36] model using the siamese and triplet networks to update the weights such that the resulting sentence embeddings are semantically meaningful and semantically similar sentences are closely positioned in the embedding space. For one epoch, a 3-way softmax classifier objective function is used for the fine-tuning of the BERT model. In the training phase of SBERT, two input sentences are passed through the BERT model followed by a pooling layer namely, MEAN-strategy, and MAX-strategy. A fixed-size representation for the input sentences are generated by this pooling layer. Next, they are concatenated with the element-wise difference and multiplied with a trainable weight. The cross-entropy loss is used for optimization. In order to encode the semantics, the twin network is fine-tuned on Semantic Textual Similarity data. SBERT model follows a two-step process in which it is first trained on Wikipedia via BERT and then fine-tuned on Natural Language Inference (NLI) data. NLI is a collection of 1,000,000 sentence pairs created by combining The Stanford Natural Language Inference (SNLI)[3] and Multi-Genre NLI (MG.NLI) datasets.

In this work, the same approach is followed to extract the embedding of the textual entity descriptions as mentioned in the evaluation of the quality of sentence embeddings in [112]. Given be a textual entity description D_{e_i} denoted by a sequence of words $\{W_1, W_2, ..., W_n\}$, where W_j is the j^{th} word in the entity description, and e_i is the corresponding entity. The entity description D_{e_i} is considered as a single sequence of words which is provided as an input to the SBERT model to get the embedding of the textual entity description $\mathbf{E_{D_i}}$. The pre-trained SBERT model used in GRAND is the SBERT-SNLI-STS-base model which is fine-tuned on SNLI and STS datasets which outperforms the baseline models as shown in[112]. The MEAN pooling strategy is used in the pooling layer.

The main advantages of using the pre-trained SBERT model are:

- Since the pre-trained SBERT model is fine-tuned with two different datasets, the entity description embeddings obtained lose domain-specific knowledge and bias, and learn task-agnostic properties of the language.

[3] https://nlp.stanford.edu/projects/snli/

- Unlike static word embedding models, such as word2vec, the contextual embedding model SBERT encodes the semantics of the words differently based on different contexts. Therefore, the entity description embeddings capture the contextual information for the task of entity typing unlike the baseline models [61, 155].
- They are computationally inexpensive as the model is pre-trained on a huge amount of text and can be easily fine-tuned based on the information available.
- A representation of the entities can be obtained from the textual entity description for long-tailed entities in the KG, i.e., entities with no or few properties.
- A task-specific classification model can be deployed on top of the entity description embeddings for entity typing tasks as illustrated in the proposed GRAND framework.

8.3.3 *Entity Type Prediction*

The proposed framework GRAND consists of three different classification modules: **(1)** Multi-class, **(2)** Multi-label, and **(3)** Hierarchical, which are discussed in details in this section.

8.3.3.1 *Entity Representation.*

The aforementioned approaches generate entity embeddings from various RDF2vec variants and from the contextual embedding model SBERT, which are provided as input to the classification modules. The input entity vectors are generated by concatenating the different vectors generated by the embedding models as depicted in component (C) in Figure 8.2. Formally, they are given by:

$$\begin{aligned} E_i &= \mathbf{E}^{\mathbf{V_i}}_{\mathbf{RDF2vec_{classic}}} \oplus \mathbf{E}^{\mathbf{V_i}}_{\mathbf{p-RDF2vec}} \oplus \mathbf{E}^{\mathbf{V_i}}_{\mathbf{e-RDF2vec}} \oplus \mathbf{E}_{\mathbf{D_i}} \\ &= \mathbf{E}^{\mathbf{V_i}}_{\mathbf{RDF2vec_{classic}}} \oplus \mathbf{E}^{\mathbf{V_i}}_{\mathbf{p-RDF2vec}} \\ &= \mathbf{E}^{\mathbf{V_i}}_{\mathbf{RDF2vec_{classic}}} \oplus \mathbf{E}^{\mathbf{V_i}}_{\mathbf{e-RDF2vec}} \\ &= \mathbf{E}^{\mathbf{V_i}}_{\mathbf{p-RDF2vec}} \oplus \mathbf{E}^{\mathbf{V_i}}_{\mathbf{e-RDF2vec}} \\ &= \mathbf{E}^{\mathbf{V_i}}_{\mathbf{p-RDF2vec}} \oplus \mathbf{E}^{\mathbf{V_i}}_{\mathbf{e-RDF2vec}} \end{aligned} \tag{8.5}$$

8.3.3.2 *Classifiers.*

MULTI-CLASS CLASSIFICATION A Fully Connected Neural Network (FCNN) consisting of two dense layers with ReLU as an activation function is deployed on the top of the entity representation. A softmax classifier with a cross-entropy loss function is

used in the last layer to calculate the probability of the entities belonging to different classes. Formally it is given by,

$$f(s)_i = \frac{e^{s_i}}{\sum_j^{C_T} e^{s_j}}, \quad \text{and} \quad CE_{loss} = -\sum_i^{C_T} t_i \log(f(s)_i), \tag{8.6}$$

where s_j are the scores inferred for each class in C_T given in Equation 8.6. t_i and s_i are the ground truth and the score for each class in C, respectively.

MULTI-LABEL CLASSIFICATION Here, an entity can belong to more than one class or type. Therefore, a certain entity e_i belonging to one class c_i has no impact on the decision of it belonging to another class c_j, where $c_i, c_j \in C_T$. A FCNN with RELU as an activation function is used for the two dense layers. A sigmoid function with binary cross-entropy loss is used in the last layer which sets up a binary classification problem for each class in C_T and is given by,

$$CE_{loss} = -t_i \log(f(s_i)) - (1 - t_i)\log(1 - f(s_i)), \tag{8.7}$$

where s_i and t_i are the score and ground truth for i^{th} class in C_T.

HIERARCHICAL CLASSIFICATION Hierarchical Classification can be broadly categorized into local and global classification. The local information in local classifier can be utilized in different ways leading to different types of local classifiers such as Local classifier Per Node (LPN), a Local classifier Per Parent Node (LPPN) and a Local classifier Per Level (LPL) [64]. The proposed framework GRAND uses LPL which consists of training a flat classifier for each level of the class hierarchy. A multi-class classifier is trained at each level of the class hierarchy and is used to discriminate among the classes at that level. The two main advantages of the LPL model are:

- It is computationally efficient compared to LPN for large KGs consisting of a large number of classes as the LPN model would have an equal number of classifiers. The number of classifiers in LPL is restricted to the number of levels in the class hierarchy.
- Since a single classifier is trained at each level, it reduces the horizontal class prediction inconsistencies.

In GRAND, a two-layered FCNN with ReLU activation function and cross-entropy loss has been deployed at each level of the class hierarchy. However, one of the drawbacks of LPL is that an entity can be classified as class 1 at one level and then it can be again classified as class 2.1 on the second level. Here, class, 2.1 is not a subclass of 1 and the entity should be classified as a subclass of 1. In order to tackle such inconsistencies,

Parameters	DB-1	DB-2	DB-3	FIGER
#Entities	210,000	210,000	210,000	201,933
#Entities train	105,000	105,000	105,000	101,266
#Entities test	63,000	63,000	63,000	60,447
#Entities validation	42,000	42,000	42,000	40,220
#Classes	48	48	48	102

Table 8.1: Statistics of the datasets

in this work, the entity which is misclassified as 2.1 in level 2 will be typed as 1 as its entity type as it was correctly identified in level 1.

8.4 EXPERIMENTS

This section provides details on the benchmark datasets, experimental setup, analysis of the results obtained, and the ablation study.

8.4.1 *Datasets*

GRAND is evaluated on the same datasets as CAT2Type, namely, the three DBpedia splits DB-1, DB-2, and DB-3 generated from DBpedia630k [162] and FIGER [154]. The statistics are provided in Table 8.1 and further details on the datasets are given in Section 7.4.1 of Chapter 7 .

8.4.2 *Experimental Setup*

The experiments are conducted on six sets of embeddings: (1) Classic RDF2vec, (2) e-RDF2vec, (3) p-RDF2vec, (4) Classic RDF2vec$_{oa}$, (5) e-RDF2vec$_{oa}$, and (6) p-RDF2vec$_{oa}$. The walks are generated with a depth of 8 and 500 walks per entity. Classic and OA embeddings are trained using SG with 200 dimensions and 5 epochs. For training, the order-aware variants (4-6), and walks from the corresponding non-order-aware variants (1-3) are reused. The training was performed using the jRDF2vec framework[4] [105]. The RDF2vec embeddings are calculated on a Debian 11 machine with 768GiB of RAM and 32 cores à 2.60GHz (Intel Xeon). All the classifiers are used with batch size 64, 100 epochs, and adam optimizer. The SBERT model and classification models are performed on an Ubuntu 16.04.5 LTS system with 503GiB RAM with TITAN X (Pascal) GPU.

[4] https://github.com/dwslab/jRDF2Vec

Table 8.2: Results of GRAND on benchmark datasets. The best result of each mode is printed in bold, the runner-up is underlined.

	Model	DB-1		DB2		DB3		FIGER	
		Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1
Baselines	CUTE [151]	0.679	0.702	0.681	0.713	0.685	0.717	0.743	0.782
	MuLR [155]	0.748	0.771	0.757	0.784	0.752	0.775	0.776	0.812
	FIGMENT [154]	0.740	0.766	0.738	0.765	0.745	0.769	0.785	0.819
	APE [60]	0.758	0.784	0.761	0.785	0.760	0.782	0.722	0.756
	HMGCN-no hier [61]	0.785	0.812	0.794	0.820	0.791	0.817	0.789	0.827
	CAT2Type-BERT [**BiswasSSA21**]	0.983	0.984	0.983	0.983	0.985	0.985	0.764	0.881
GRAND Coarse-grained	classic-RDF2vec$_{oa}$ ⊕ s-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ SBERT	**0.991**	**0.991**	**0.990**	**0.990**	**0.989**	**0.989**	**0.801**	**0.893**
	SBERT - only	0.972	0.972	0.97	0.97	0.97	0.97	0.648	0.844
Baselines Fine-grained	CAT2Type-BERT [**BiswasSSA21**]	0.402	0.732	0.369	0.721	0.847	0.915	0.703	0.835
	CAT2Type-node2vec [**BiswasSSA21**]	0.391	0.694	0.365	0.677	0.807	0.878	0.701	0.833
GRAND Fine-grained	classic-RDF2vec$_{oa}$ ⊕ s-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ SBERT	**0.745**	**0.870**	**0.723**	**0.851**	**0.880**	**0.931**	**0.706**	**0.881**
Baseline Hierarchical	HMGCN-hier [61]	**0.794**	0.816	**0.796**	0.824	**0.798**	0.819	**0.798**	0.836
GRAND Hierarchical	classic-RDF2vec$_{oa}$ ⊕ s-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$	0.731	**0.882**	0.729	**0.881**	0.726	0.877	0.701	0.880
	classic-RDF2vec$_{oa}$ ⊕ s-RDF2vec$_{oa}$ ⊕ p-RDF2vec$_{oa}$ ⊕ SBERT	0.731	0.875	0.718	0.869	**0.935**	**0.946**	0.712	**0.883**

8.5 RESULTS

In order to evaluate the proposed approach against the baseline models, Micro-averaged F_1 (Mi-F_1) and Macro-averaged F_1 (Ma-F_1) metrics are used along with the accuracy. Different variants of *RDF2vec* have been evaluated which serve as an ablation study. The baselines used for the experiments are: CUTE [151], MuLR [155], FIGMENT [154], APE [60], HMGCN [61], and CAT2Type [**BiswasSSA21**]. The results of the proposed framework on two benchmark datasets and their comparison with the baseline models are depicted in Table 8.2.

The results of GRAND as depicted in Table 8.2 can be obtained as follows:

- *Coarse-grained setting:* For DBpedia splits, the original dataset consisting of 14 non-overlapping classes is used. For FIGER, the number of coarse-grained classes is 30 and they are non-overlapping as well. Since none of the entities belongs to more than one class, *multi-class* classification settings have been used here.

- *Fine-grained setting:* The original DBpedia630k dataset is expanded with the DBpedia hierarchy to 37 fine-grained classes and these are non-overlapping classes. Therefore, a *multi-class* classification model is used here as well. On the other hand, the FIGER dataset consists of overlapping fine-grained classes, i.e., one entity can belong to multiple classes. Therefore, a *multi-label* classification is used for the fine-grained FIGER dataset.
- For *Hierarchical Classification*, a classifier on each level of the hierarchy is deployed. For DBpedia splits, it is a multi-class classification model and for FIGER it is a multi-label classification model at each level of the hierarchy.

The results show that GRAND outperforms the SoTA model CAT2Type with an improvement of 0.8% on Ma-F_1 and 0.7% on Mi-F_1 for DB-1, 0.7% and 0.4% on both the metrics for DB-2 and DB-3 respectively for the coarse-grained classes. The original dataset with 14 classes which do not contain the hierarchy is used for this coarse-grained non-hierarchical variant. Furthermore, for hierarchical classification, the proposed model significantly outperforms the SoTA HMGCN-hier model with an increment of 6.6% for DB-1, 5.7% for DB-2, and 12.7% for DB-3 on the Mi-F_1 measure. For FIGER, the coarse-grained approach is a multi-class classification whereas the fine-grained approach is a multi-label classification. GRAND achieves the best results for FIGER on the coarse-grained approach which outperforms the baseline models. Moreover, with the multi-label fine-grained settings, it achieves comparable results with the non-hierarchical baseline model CAT2Type and significantly outperforms the other non-hierarchical model HMGCN. One advantage of GRAND over CAT2Type is that it can be applied to any KGs and is not restricted to KGs containing information on Wikipedia Categories.

Table 8.3 and Table 8.4 show the experimental results of the proposed approach for the coarse-grained and fine-grained classes respectively with different variants of RDF2vec and their combinations. The experiments using the *Single* strategy show that *all* order-aware RDF2vec embeddings significantly outperform their classic counterparts. Therefore, the fusion strategies only focus on position-aware embeddings in order to reduce combinatorial complexity.

8.5.1 *Impact of RDF2vec on Different Classification Settings*

COARSE-GRAINED ENTITY TYPING Table 8.3 shows the results of the experiment for coarse-grained entity typing. On the DB1 Split of the dataset, the best results for GRAND are obtained where the models are combined, i.e., $\texttt{classic-RDF2vec}_{oa} \oplus \text{p-RDF2vec}_{oa} \oplus \text{e-RDF2vec}_{oa}$ (concat) outperforms HMGCN for Ma-F_1 by 0.1744 and for Mi-F_1 by 0.148 and achieves comparable results with CAT2Type. However, e-RDF2vec configurations perform the weakest on their own but introduce additional value when combined with other approaches as depicted in the concat model. The best-

performing configuration includes entity embeddings. Given the data, it appears that the PCA discards too much valuable information for DBpedia splits but not for FIGER. Overall, it can be observed that the performance differences between p-RDF2vec and classic-RDF2vec are minor. Nonetheless, the embeddings encode different information which is visible when combining the embeddings. Therefore, it can be concluded that the contextual information of the entities in form of a path captures the characteristic features of the entities. A similar observation has been made for both DB2, DB3 split and FIGER. A detailed analysis of the impact of different vector components is provided in Section 8.5.2.

FINE-GRAINED ENTITY TYPING GRAND is compared with the two best variants of CAT2Type namely BERT and node2vec as shown in Table 8.2 and results show that the proposed model significantly outperforms the CAT2Type model for all DBpedia splits and FIGER. In general, it is observed for uneven class distribution the evaluation metric Ma-F_1 achieves lower values compared to Mi-F_1. However, the Ma-F_1 results of GRAND for DB1 and DB2 splits are much better than that of CAT2Type. It strengthens the fact that the representation of entities obtained using strategic graph walks and contextual embedding of entity descriptions contain more information about entities compared to the embeddings used in CAT2Type.

HIERARCHICAL CLASSIFICATION. Table 8.5 shows the results of the hierarchical classification of the GRAND framework on different levels of the class hierarchy. The performance is computed for only $\texttt{classic-RDF2vec}_{oa} \oplus \text{p-RDF2vec}_{oa} \oplus \text{e-RDF2vec}_{oa}$ since it is the highest performing model based on experiments discussed in previous sections. The results show higher performances on level 1 since the number of classes are lesser i.e., 5, as compared to other levels. GRAND outperforms the baseline model HMGCN-*withHier* for Mi-F_1 metric as depicted in Table 8.2.

TEXTUAL ENTITY DESCRIPTIONS To analyze the impact of entity descriptions, a multi-class classification has been performed on the entity embeddings generated from the SBERT model. As shown in Table 8.2, GRAND with only SBERT performs better than all the baseline models except CAT2Type. Therefore, it can be concluded that contextual embeddings using SBERT provide the necessary relevant information as compared to the triple-based baseline models.

8.5.2 *Analysis of Vector Component Weight.*

As discussed above, in the experiments it can be seen that the concatenation of embeddings achieves the best result. Therefore, it is further evaluated (1) which components are the most and the least important for the predictions and (2) whether there is a difference in the weights given the coarse-grained and the fine-grained prediction tasks.

Dataset	Mode	Model	DB-1			DB-2			DB-3			FIGER		
			ACC	Ma-F_1	Mi-F_1	ACC	Ma-F_1	Mi-F_1	ACC	Ma-F_1	Mi-F_1	ACC	Ma-F_1	Mi-F_1
Coarse-Grained	Single	classic-RDF2vec	0.9163	0.9150	0.9163	0.9062	0.9043	0.9062	0.9123	0.9109	0.9123	0.931	**0.431**	0.778
		classic-RDF2vec$_{oa}$	**0.9448**	**0.9439**	**0.9448**	**0.9346**	**0.9330**	**0.9346**	**0.9457**	**0.9449**	**0.9457**	**0.933**	0.419	**0.781**
		e-RDF2vec	0.7352	0.7318	0.7352	0.7250	0.7308	0.7250	0.7357	0.7304	0.7357	0.927	0.421	0.771
		e-RDF2vec$_{oa}$	0.7665	0.7651	0.7665	0.7625	0.7453	0.7625	0.7694	0.7650	0.7694	0.927	0.422	0.771
		p-RDF2vec	0.8949	0.8946	0.8949	0.8999	0.8914	0.8999	0.8882	0.887	0.8882	0.922	0.426	0.778
		p-RDF2vec$_{oa}$	0.9412	0.9404	0.9412	0.9332	0.9303	0.9332	0.9430	0.9421	0.9430	0.928	0.422	0.779
	Concat	e-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	0.9518	0.9512	0.9518	0.9482	0.9412	0.9482	0.9502	0.9495	0.9502	0.912	0.414	0.77
		e-RDF2vec$_{oa}$ $\oplus$ classic-RDF2vec$_{oa}$	0.9450	0.9444	0.9450	0.9450	0.9144	0.9450	0.9452	0.9482	0.9452	0.908	0.418	0.772
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	0.9564	0.9555	0.9563	0.9560	**0.9546**	0.9560	**0.9582**	0.9513	**0.9592**	0.92	**0.429**	**0.774**
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$ $\oplus$ e-RDF2vec$_{oa}$	**0.9600**	**0.9594**	**0.9600**	**0.9667**	0.9544	**0.9667**	0.9572	**0.9564**	0.9574	**0.924**	0.424	0.772
	Local PCA	e-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	0.8855	0.8845	0.8855	0.8757	0.8770	0.8757	0.8918	0.8905	0.8918	0.921	0.422	0.769
		e-RDF2vec$_{oa}$ $\oplus$ classic-RDF2vec$_{oa}$	0.9323	0.9314	0.9324	0.9314	0.9122	0.9314	0.9015	0.9000	0.9015	0.919	0.419	0.770
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	**0.9471**	**0.9466**	**0.9472**	0.9442	**0.9300**	0.9442	0.9378	0.9217	0.9378	0.92	0.421	0.724
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$ $\oplus$ e-RDF2vec$_{oa}$	0.9405	0.9395	0.9405	**0.9551**	0.9195	**0.9551**	**0.9413**	**0.9402**	**0.9413**	**0.925**	**0.428**	**0.778**
	Global PCA	e-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	0.9325	0.9316	0.9325	0.9412	0.9330	0.9412	0.9321	0.9310	0.9321	0.923	**0.428**	0.778
		e-RDF2vec$_{oa}$ $\oplus$ classic-RDF2vec$_{oa}$	0.9413	0.9405	0.9414	0.9322	0.9311	0.9322	0.9416	0.9405	0.9416	0.925	0.428	0.776
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	**0.9499**	**0.9490**	**0.9499**	0.9356	0.9212	0.9356	**0.9490**	**0.9482**	**0.9490**	0.927	0.427	0.767
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$ $\oplus$ e-RDF2vec$_{oa}$	0.9476	0.9468	0.9476	**0.9568**	**0.9412**	**0.9568**	0.9489	0.9481	0.9489	**0.929**	**0.433**	**0.779**

Table 8.3: Evaluation of Single Classifier Results on the Coarse-Grained Dataset. The best result of each mode is printed in bold, the runner-up is underlined. The overall best configuration for each dataset is bold and underlined.

Dataset	Mode	Model	DB-1			DB-2			DB-3			FIGER		
			ACC	Ma-F_1	Mi-F_1	ACC	Ma-F_1	**Mi-F_1**	**ACC**	Ma-F_1	Mi $-$ F_1	ACC	Ma-F_1	Mi-F_1
Fine-Grained	Single	classic-RDF2vec	0.6716	0.374	0.672	0.6635	0.363	0.663	0.8402	0.736	0.840	**0.991**	0.467	0.774
		classic-RDF2vec$_{oa}$	0.704	**0.386**	0.704	0.701	0.356	0.701	**0.871**	**0.774**	**0.871**	0.987	**0.469**	**0.778**
		e-RDF2vec	0.564	0.297	0.5643	0.5231	0.3164	0.5231	0.6709	0.5632	0.6709	0.946	0.445	0.721
		e-RDF2vec$_{oa}$	0.5831	0.3064	0.5831	0.5542	0.3174	0.5442	0.6926	0.5747	0.6926	0.951	0.452	0.722
		p-RDF2vec	0.6500	0.3549	0.6499	0.6504	0.3449	0.6504	0.7848	0.6513	0.7848	0.949	0.467	0.77
		p-RDF2vec$_{oa}$	**0.706**	0.384	**0.706**	**0.702**	**0.381**	**0.7022**	0.847	0.732	0.8471	0.951	0.459	0.772
	Concat	e-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	0.699	0.378	0.6996	0.698	0.388	0.698	0.877	0.784	0.877	0.949	0.454	0.774
		e-RDF2vec$_{oa}$ $\oplus$ classic-RDF2vec$_{oa}$	0.698	0.374	0.6978	0.701	0.384	0.7011	0.881	0.7811	0.881	0.96	0.512	0.781
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	**0.707**	0.386	**0.707**	0.719	**0.396**	0.719	0.887	0.781	0.881	0.955	**0.519**	0.778
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$ $\oplus$ e-RDF2vec$_{oa}$	0.703	**0.393**	0.720	**0.7204**	0.3912	**0.720**	**0.890**	**0.801**	**0.8908**	**0.961**	**0.519**	**0.783**
	Local PCA	e-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	0.653	0.358	0.6538	0.648	0.385	0.648	0.806	0.695	0.8060	0.948	0.457	0.778
		e-RDF2vec$_{oa}$ $\oplus$ classic-RDF2vec$_{oa}$	0.6865	0.3683	0.6865	0.6952	0.3682	0.6952	0.8746	0.7770	0.8746	0.951	0.501	0.779
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	**0.7006**	**0.3902**	**0.7006**	0.7116	**0.3907**	0.7116	**0.8774**	**0.7801**	**0.8774**	0.950	0.504	0.771
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$ $\oplus$ e-RDF2vec$_{oa}$	0.6936	0.3839	0.6936	**0.7122**	0.3438	**0.7123**	0.864	0.764	0.864	**0.958**	0.514	0.781
	Global PCA	e-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	0.6845	0.3716	0.6844	0.66125	0.3189	0.6612	0.855	0.7525	0.8547	0.942	0.449	0.772
		e-RDF2vec$_{oa}$ $\oplus$ classic-RDF2vec$_{oa}$	0.6908	**0.3879**	0.6908	0.67143	0.3119	0.67143	0.8677	0.7686	0.8677	0.945	0.449	0.769
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$	0.6981	0.3778	0.6981	0.6881	0.3241	0.6881	**0.8754**	0.7771	**0.8754**	0.956	0.457	0.771
		classic-RDF2vec$_{oa}$ $\oplus$ p-RDF2vec$_{oa}$ $\oplus$ e-RDF2vec$_{oa}$	**0.7005**	0.3768	**0.7004**	**0.7014**	**0.3228**	**0.7014**	0.8709	**0.7780**	0.8709	**0.961**	**0.498**	**0.784**

Table 8.4: Evaluation of Single Classifier Results on the Fine-Grained Dataset. The best result of each mode is printed in bold, the runner-up is underlined. The overall best configuration for each dataset is bold and underlined.

Level	#classes	DB1		DB2		DB3	
		Ma-F_1	Mi-F_1	Ma-F_1	Mi-F_1	Ma-F_1	Mi-F_1
1	5	**0.961**	**0.962**	**0.960**	**0.960**	**0.959**	**0.959**
2	11	0.744	0.925	0.747	0.929	0.744	0.924
3	12	0.857	0.934	0.851	0.926	0.859	0.935
4	17	0.361	0.705	0.358	0.702	0.359	0.674

Table 8.5: Results of the GRAND-LPL classification model at each level

Dataset	Epoch	SBERT	Classic RDF2vec$_{oa}$	p-RDF2vec$_{oa}$	e-RDF2vec$_{oa}$
Coarse-Grained	1	-	35.5%	44.4%	20.0%
	10	-	32.9%	49.9%	17.1%
	1	58.04%	14.6%	16.28%	11.08%
	10	47.9%	18.5%	22.8%	10.8%
Fine-Grained	1	-	35.4%	42.1%	22.5%
	10	-	33.6%	46.4%	20.0%
	1	56.7%	15.36%	16.84%	11.1%
	10	51.19%	16.83%	19.5%	12.48%

Table 8.6: Relative network weights of each vector component group for DB-1 split.

EXPERIMENTAL SETUP. In order to analyze the weights each vector component receives in the neural network, an FCNN with one layer was trained on the combination of all ordered aware RDF2vec (depicted in 1st 2 rows in coarse-grained and 1st 2 rows in fine-grained in Table 8.6) and also with SBERT. It is important to note that the overall goal of this setup is to analyze how much weight each of the four-vector groups receives. Therefore, the sum of absolute weights in the network given to each vector is calculated for the first and the tenth epoch.

RESULTS. The relative weights can be found in Table 8.6. It is observed that the highest overall impact is independent of the dataset, achieved using the p-RDF2vec embeddings. This is followed by the classic RDF2vec embeddings. The least impact is achieved by the e-RDF2vec embeddings. Interestingly, a weight shift occurs when switching from coarse-grained entity typing to fine-grained entity typing, i.e., it is visible that the classic and the entity embeddings are more important for fine-grained predictions. The results suggest that p-RDF2vec is helpful for coarse-grained type prediction – an intuitive finding is given that p-RDF2vec encodes structural similarity. However, the more fine-grained the task gets the more important the *actual* neighbour vertices.

8.6 CONCLUSION AND OUTLOOK

This chapter proposes a novel entity type prediction framework, named **GRAND** based on different variants of RDF2vec and textual entity descriptions. These variants are constructed with different walk-generation strategies and a new order-aware variant of word2vec. GRAND is evaluated on DBpedia630k and FIGER datasets. The achieved results show that GRAND considerably outperforms all the baseline models. GRAND outperforms the CAT2Type model for fine-grained classification in all DBpedia splits as well as in FIGER as depicted in Table 8.2. This strengthens the fact that the contextual structural information provides more information about the entities as compared to the Wikipedia categories. Since the hierarchical classification is done at each level in the type hierarchy, therefore it overcomes the problem of the top-down hierarchical classification of the baseline model CUTE [151]. Section 8.2 formulates and presents the two main research questions. The following are the answers to these queries:

- *C2-RQ2: What is the impact of textual entity descriptions in predicting the corresponding missing types?*
  - The experimental results with only the embeddings obtained using the SBERT model show that GRAND outperforms all the baseline models except for CAT2Type. Also, the weight analysis in Table 8.6 shows that the text embeddings have higher weightage than the RDF2vec variants in the classification process. Therefore, the entity descriptions are rich in information content which helps in identifying the missing types of entities. Furthermore, the pre-trained SBERT lose domain-specific knowledge and bias and learns task-agnostic language-specific features. They are computationally inexpensive and can be easily fine-tuned.
- *C2-RQ3: Are strategic graph walks beneficial for entity typing?*
  - Evaluation results depicted in Tables 8.3 and 8.4 are from only RDF2vec models which consider the strategic graph walks and they outperform all the baseline models. It is also noted that order-aware RDF2vec performs better than classical RDF2vec for entity typing. This concludes that including the positional information of the entities and relations generates a better representation of the entities.

One major advantage of this model is that entity representations can be obtained for entities with only textual entity descriptions and no properties and vice-versa. The model can also integrate other text literals that are accessible for the entities in the KGs as labels, summaries, comments, etc. Also, given the weight analysis, further experimentation on more fine-granular type systems – such as in YAGO [130] or CaLiGraph [51] is to be conducted.

9 ENTITY TYPE PREDICTION LEVERAGING ENTITY NAMES

Entity Typing is a vital task in Knowledge Graph (KG) completion and construction. In the previous chapters, different methods have been proposed that uses different features from the KGs namely Wikipedia categories, random graph walks, and textual entity descriptions. The evaluation shows that the models outperform the existing baseline methods. However, this chapter focuses on predicting the missing types of entities without any information from the KGs and considers merely the names of the entities. In order to predict the missing entity types, this chapter proposes a framework [10] that makes use of contextual and non-contextual NLMs. The framework also enables multilinguality [8] and is not limited to entity names in English exclusively.

The rest of the chapter is structured as follows: Section 9.1 motivates the research challenge tackled in this chapter, followed by Section 9.2. The approach is discussed in Section 9.3 and then details of experiments and results on the entity typing on English dataset are provided in Section 9.4 and that of multilingual datasets is discussed in Section 9.5. Section 9.6 provides the conclusion and the outlook.

9.1 INTRODUCTION

One of the fundamental building blocks of KG is entity types. Recent years have witnessed research in the automated prediction of entity types in KGs as already discussed in detail in Chapter 4. The existing baseline models exploit the triples in the KGs whereas others consider the textual entity descriptions as well. While those approaches work well if there is a lot of information about an entity, it is still a challenge to type entities for which there is only scarce information. There are many non-popular entities or new entities that are added to the KG, i.e., entities with fewer or no triples associated with them. Therefore, this chapter focuses on predicting the entity types solely from their label names, e.g., *Is it possible to predict that the entity* `dbr:Berlin` *is a place only from its name?*.

Furthermore, to be able to predict the types of entities just by their names, one has to understand multiple languages. Therefore, this originates the necessity of an automated multilingual entity-type prediction framework for different chapters in DBpedia. For example, *Is it possible to predict the types of the entities dbr: Lachse, dbr: Saumon, dbr: Salmo, and dbr: Zalm from their names?* These are the names of *Salmon fish* in German, French, Spanish, and Dutch respectively. Therefore, this paper focuses on predicting the types of entities just by their names for different language chapters of DBpedia,

namely German (DE), French (FR), Spanish (ES), and Dutch (NL). The main challenges of this work are as follows:

- To predict the types of the entities for which significantly less or no triples are available in the KGs, and
- to predict the types of the entities in different languages.

This lack of available information is compensated by exploiting the NLMs. They are trained on a huge amount of monolingual as well as multi-lingual textual data, and they provide implicit contextual information about the entities in their corresponding language-agnostic vector representations. To do so, the continuous space-based Neural Language Models (NLM) such as Word2Vec, GloVe, Wikipedia2Vec, and BERT as well as a character embedding model is exploited for the entity names in English whereas the Multilingual Neural Language Models (Multilingual-NLMs), namely Wikipedia2Vec, and m-BERT are used for the multilingual entity typing.

In this work, the task of entity typing is considered a classification problem in which a neural network-based classifier is applied on top of the NLMs. Furthermore, an analysis of the performance of the different NLMs for this task is provided. The main contributions are:

- A multi-class classification framework is proposed to predict the missing entity types in English DBpedia as well as the multilingual DBpedia chapters exploiting the NLMs.
- A benchmark dataset for multilingual entity typing consisting of entities from German (DE), French (FR), Spanish (ES), and Dutch (NL) DBpedia chapters are published for re-usability purposes for future research.
- The results show that the entity typing model trained on the entities from English DBpedia also performs well for predicting the types of unseen entities from the CaLiGraph dataset [51].

9.2 PROBLEM FORMULATION

Following KG definition from Chapter 2, a KG $\mathcal{G}$ consists of a set of triples $\mathcal{T}$, given by, $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{C})$, where $\mathcal{E}$, $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{C}$ are the set of entities, relations between the entities, literals, and semantic types of the entities respectively. $< e_h, r, e_t > \in \mathcal{T}$, represents a triple belonging to the set of triples $\mathcal{T}$ in the KG, where $(e_h, e_t) \in \mathcal{E}$ are the head and tail entities, and $r \in \mathcal{R}$ represents relation between them. `rdf:type` is an instance of `rdf:Property` that is used to state that a resource is an instance of a class. A triple of the form: $< e_i, \texttt{rdf:type}, C_k >$, states that $C_k \in C$, is an instance of `rdfs:Class` and $e_i \in E$ is an entity in G and is an instance of C_k. The proposed model intends to predict the missing types of the entities in a KG addressing the research question mentioned in Section 1.2 of Chapter 1 and is given by,

- *C2-RQ4: Can the types of entities be predicted just from the entity names?*

9.3 ENTITY TYPE PREDICTION: NAMES-ONLY FRAMEWORK

This section discusses the NLMs and the classifiers used for the task of entity typing only from the names of the entities.

As mentioned earlier, to predict the missing types of entities in English DBpedia, the framework uses the non-contextual word embedding models Word2vec, GloVe, BERT, Wikipedia2vec, and Character embedding. On the other hand, for multilingual entity typing Multi-lingual BERT (m-BERT) [37] and Wikipedia2vec are used. The details of these models except for the character embedding and m-BERT are provided in Section 2.4. Chapter 2.

CHARACTER EMBEDDING Character embedding [161] represents the latent representations of characters trained over a corpus using CNN which helps in determining the vector representations of out-of-vocabulary words. A character embedding model includes the following steps: all the unique characters of a language are converted to a one-hot encoding, followed by 1-D CNN layers to learn the sequence.

M-BERT **B**idirectional **E**ncoder **R**epresentations from **T**ransformers [37] is a contextual embedding approach in which pretraining on bidirectional representations from the unlabeled text by using the left and the right context in all the layers is performed. Multilingual-BERT (m-BERT) supports 104 languages trained on text from Wikipedia content with a shared vocabulary across all languages. However, the size of Wikipedia varies greatly for different languages. The low-resource languages are underrepresented in the neural network model compared to the popular languages. The training on the low-resource languages of Wikipedia for a large number of epochs results in the overfitting of the model. To combat the content imbalance of Wikipedia, less popular languages are over-sampled, whereas popular languages are under-sampled. An exponential smoothing weighting of the data during the pre-training data creation is used. For tokenization, a 110k shared WordPiece vocabulary is used. The word counts are weighted following the same method for the pre-training data creation. Therefore, the low-resource languages are up-weighted by some factors. Given an entity name $E_i = (w_1, w_2, ..., w_n)$, the input sequence to the m-BERT model is given by $([CLS], w_1, w_2, ..., w_n, [SEP])$, where E_i is the i^{th} entity and $w_1, w_2, .., w_n$ are the n words in the entity name. [CLS] and [SEP] are special tokens that mark the beginning and the end of the input sequence.

EMBEDDINGS OF THE ENTITY NAMES. In this work, pre-trained Word2Vec model on Google News dataset[1], GloVe model pre-trained on Wikipedia 2014 version and Gi-

[1] https://code.google.com/archive/p/word2vec/

gaword 5[2], Wikipedia2Vec model pre-trained on Wikipedia 2018 version on as well as for each of the languages, i.e., DE, FR, ES, and NL [3], and pre-trained English character embeddings derived from GloVe 840B/300D dataset[4], is used with a vector dimension of 300. The average of all word vectors in the entity names is taken as the vector representation of the entities. For BERT, the average of the last four hidden layers of the model is taken as a representation of the names of entities and the dimension used is 768.

For multilingual entity typing, the m-BERT base model has been used, in which each position outputs a vector of dimension equal to that of its hidden layer and its corresponding dimension is 768 for the base model. Each entity name is considered as a sentence for the input to m-BERT. The average of the last four hidden layers is taken to represent the entities. For Wikipedia2vec, the average of all word vectors in each entity name is taken as the vector representation of the entity. The entity representation generated by the static non-contextual NLMs for both English and multilingual entity typing is formally given by,

$$E_{e_i}^{model_i} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{W}_j, \tag{9.1}$$

where $E_{e_i}^{model_i}$, is the entity representation, n is the number of words in the entity name, and $\mathbf{W}$ is the corresponding vector of the word W in the entity name. Furthermore, more multilingual entity typing concatenation of Wikipedia2vec and m-BERT vectors are being used to obtain the entity representation.

CLASSIFICATION In this work, entity typing is considered a classification task with the types of entities as classes. Two classifiers have been built on top of the NLMs for the typing the entities from English DBpedia: (**i**) Fully Connected Neural Network (FCNN), and (**ii**) Convolutional Neural Network (CNN). A three-layered FCNN model consisting of two dense layers with ReLU as an activation function has been used on the top of the vectors generated from the NLMs. The softmax function is used in the last layer to calculate the probability of the entities belonging to different classes. The CNN model consists of two 1-D convolutional layers followed by a global max-pooling layer. `ReLu` is used as an activation function in the convolutional layers and the output of the pooling layer is then passed through a fully connected final layer, in which the softmax function predicts the classes of the entities. Whereas for the multi-lingual entity typing only FCNN with the same configuration mentioned above is being deployed.

[2] http://nlp.stanford.edu/data/glove.6B.zip [3] https://wikipedia2vec.github.io/wikipedia2vec/
[4] https://github.com/minimaxir/char-embeddings/blob/master/output/

9.4 EXPERIMENTS AND RESULTS ON ENTITY NAMES IN ENGLISH

This section consists of a detailed description of the datasets used for evaluating the English-based entity typing model, followed by an analysis of the results obtained.

9.4.1 *Experimental Setup*

The same experimental setup has been used by both models. The classifiers used have batch size $\{32, 64\}$ and 100 epochs. The experiments are performed on an Ubuntu 16.04.5 LTS system with 503GiB RAM with TITAN X (Pascal) GPU.

9.4.2 *Datasets*

The experiments are conducted on the benchmark dataset DBpedia630k [162] extracted from DBpedia consisting of 14 non-overlapping classes[5] with 560,000 train and 70,000 test entities. However, predicting fine-grained type information of an entity only from its name is a non-trivial task. For e.g. identifying `dbr:Kate_Winslet` as an *Athlete* or *Artist* from only the entity name is challenging. Therefore, seven coarse-grained classes of the entities in this dataset are considered: *dbo:Organisation*, *dbo:Person*, *dbo:Place*, *dbo:MeanOfTransportation*, *dbo:Animal*, *dbo:Plant*, and *dbo:Work*. Also, 4.656% of the total entities in the train set and 4.614% entities in the test set have their type information mentioned in their RDF(S) labels. For example, *dbr:Cybersoft_(video_game_company)* has the label *Cybersoft (video game company)* stating that it is a *Company*. Therefore, the experiments are conducted both with and without the type information in the names for the DBpedia630k dataset. To evaluate the approaches independently of DBpedia, we use an additional test set composed of entities from CaLiGraph [51]. The latter is a Wikipedia-based KG containing entities extracted from tables and enumerations in Wikipedia articles. It consists of 70,000 entities that are unknown to DBpedia and evenly distributed among 7 classes.

9.4.3 *Results*

The results in Table 9.1 depict that for all the NLMs, FCNN works better compared to the CNN model. This is because the CNN model does not work well in finding patterns in the label names of the entities. Also, BERT performs the worst in predicting the type of entities from their label names. Further error analysis shows that only 4.2% of the total person entities in the test set *with Types in Labels* variation of the dataset have been correctly identified as `dbo:Person` for BERT. Since the names of persons can be ambiguous and BERT is a contextual embedding model, the vector representations of the entities generated only from their label names do not provide

[5] https://zenodo.org/record/7688590

Table 9.1: Results on the DBpedia630k dataset (in accuracy %)

Embedding	Types in Labels		no Types in Labels		CaLiGraph Test Set	
Models	FCNN	CNN	FCNN	CNN	FCNN	CNN
word2vec	80.11	46.71	72.08	44.39	48.93	25.91
GloVe	83.34	54.06	82.62	53.41	61.88	31.3
wikipedia2vec	91.14	60.47	90.68	57.36	75.21	36.97
BERT	67.37	62.27	64.63	60.4	53.42	35.55
character embedding	73.43	58.13	72.66	58.3	54.91	45.73

a proper latent representation of the entity. However, FCNN achieves an accuracy of 84.74% on the same dataset without the class `dbo:Person` for BERT. On the other hand, Wikipedia2Vec works best amongst all the NLMs for FCNN with an accuracy of 91.14% and 90.68% on the *Types in Labels* and *no Types in Labels* variants of the dataset respectively. Also, on the removal of the class `dbo:Person` from the dataset, it achieves an accuracy of 91.01% on *Types in Labels* variant. Therefore, the decrease of 0.13% in the accuracy infers that entities of the class `dbo:Person` are well represented in the entity vectors obtained from the pre-trained Wikipedia2Vec model.

However, after removing the type information from the name labels, a slight drop in the accuracy for each model has been observed for both classifiers. Wikipedia2Vec and the character embedding model experience the smallest drop in accuracy of 0.46% and 0.77% with the FCNN classifier. This is because DBpedia entities are extracted from Wikipedia articles, therefore the vectors of the entities are well represented by the Wikipedia2Vec model. Also for character embedding, the removal of the type information from their labels has a low impact because the vector representation of the entity names depends on the corresponding character vectors and not word vectors. Furthermore, an unseen test set from CaLiGraph has been evaluated on the classification model trained on the *no Types in Labels* variation of the dataset. On the CaLiGraph test set, the FCNN model achieves the best results with the Wikipedia2Vec model with an accuracy of 75.21%. The entities in the CaLiGraph test set are not contained in DBpedia, hence the representations of these entities are not learned during the training of the Wikipedia2Vec model. This depicts the robustness of the proposed model and the entity vectors generated by taking the average of the word vectors present in the names of the entities to provide a better latent representation.

9.5 EXPERIMENTS AND RESULTS ON MULTILINGUAL ENTITY NAMES

The datasets used to evaluate the multilingual entity type model are thoroughly described in this section, which is followed by an analysis of the findings.

Table 9.2: Dataset Statistics

DBpedia chapters	Train	Test	Valid	Total Entities	#coarse-grained class	#fine-grained class
German	38500	23100	15400	77000	38	77
French	57999	34799	23199	115997	51	116
Spanish	42000	25200	16800	84000	45	84
Dutch	44000	26400	17600	88000	42	88

Table 9.3: Entity Typing Results on DE, FR, ES, and NL DBpedia Chapters

DBpedia chapters	#classes	m-BERT			Wikipedia2vec			m-BERT + Wikipedia2vec		
		Accuracy	Ma-F1	Mi-F1	Accuracy	Ma-F1	Mi-F1	Accuracy	Ma-F1	Mi-F1
German	38	0.818	0.760	0.818	0.870	0.817	0.870	**0.918**	**0.884**	**0.918**
	77	0.674	0.676	0.674	0.763	0.762	0.763	0.831	0.829	0.831
French	51	0.794	0.689	0.794	0.833	0.718	0.833	**0.867**	**0.780**	**0.867**
	116	0.544	0.542	0.544	0.611	0.612	0.611	0.678	0.680	0.678
Spanish	45	0.782	0.694	0.782	0.843	0.764	0.843	**0.894**	**0.846**	**0.894**
	84	0.629	0.627	0.629	0.681	0.682	0.681	0.788	0.788	0.788
Dutch	42	0.885	0.825	0.885	0.812	0.735	0.812	**0.908**	**0.859**	**0.908**
	88	0.664	0.665	0.664	0.753	0.757	0.753	0.825	0.825	0.825

9.5.1 *Datasets*

The work focuses on predicting the types of entities in different DBpedia chapters, namely, DE, FR, ES, and NL. The entities are extracted from the language versions of DBpedia-version 2016-10[6]. The most popular classes from each DBpedia chapter are chosen with 1000 entities per class. The coarse-grained classes are the parent classes of the fine-grained classes in the hierarchy tree. In this dataset, no entity belongs to two different classes in different hierarchy branches. Further details about the dataset are provided in Table 9.2 and are made available via Github[7].

9.5.2 *Results*

It is observed from the results as depicted in Table 9.3 that the static NLM Wikipedia2Vec trained on different languages of Wikipedia performs better than the m-BERT model for all the DBpedia chapters. BERT is a contextual embedding model that generates better latent representations where the context is available in the input sequence. The entity names are considered input sentences to the m-BERT model that do not provide

[6] http://downloads.dbpedia.org/wiki-archive/downloads-2016-10.html
[7] https://github.com/russabiswas/MultilingualET_with_EntityNames

any contextual information. On the other hand, the Wikipedia2Vec models trained on different Wikipedia languages perform better as they provide the fixed dense representation of the words or entities in the pre-trained models. It is noticeable that the concatenated vectors from m-BERT and Wikipedia2vec yield the best result as both features are combined. Furthermore, Table 9.3 shows that the model performs better for coarse-grained classes compared to the fine-grained because it is often not possible to identify if a certain entity is of the type *Scientist* or an *Actor* from its name. However, it is possible to identify if the entity is of type *Person*.

9.6 CONCLUSION AND OUTLOOK

In this chapter, different NLMs for entity typing in a KG have been analyzed. The achieved results imply that NLMs can be exploited to relevant information to predict the types of entities in a KG only from their names. Furthermore, multilingual NLMs for entity typing in a KG using entity names are also analysed. This leads to answering the research question mentioned in Section 9.2,

- *C2-RQ4: Can the types of entities be predicted just from the entity names?*
  - The results of the entity typing obtained for both English and other languages claim that the NLMs serve as a good resource to predict the missing types of entities in a KG, especially the unpopular ones. However, the static NLMs perform better for both the monolingual and multilingual settings compared to BERT or m-BERT. This is because the entity names do not have enough context for the contextual NLM. Nevertheless, the main advantage of this framework is that it can predict the missing types of entities without any information from the KGs.

In the future, fine-grained type prediction using textual entity descriptions from the KG using the multilingual NLMs is to be explored. Also, a combination of entity names together with the Wikipedia categories for multilingual entity typing is to be investigated for long-tailed entities.

Part V

CONCLUSION AND OUTLOOK

10 CONCLUSION AND OUTLOOK

This dissertation investigates the methods exploiting different features from the KGs for KGC. It focuses on two main challenges of KGC: *Link Prediction* in KGs that include head and tail prediction, triple classification, and *Entity Type Prediction*. The literature review presented in Chapters 3 and 4 shows that the embeddings of entities and relations into a low-dimensional vector space have proven to be beneficial for KGC. However, the different features in a KG are still not being used to their best extent. In this thesis, several embedding-based models leveraging these features are proposed to predict the missing links in the KG. The contributions made in this thesis are summarized in this chapter and the prospective areas for future research.

10.1 CONCLUSIONS

The importance of structural contextual information and textual entity descriptions for link prediction is studied in this dissertation.

Chapter 5 answers the research question *C1-RQ1* and *C1-RQ2* from Section 1.2 of Chapter 1 by presenting a novel KG embedding model *MADLINK*. The model exploits the structural contextual information of the entities by generating random walks together with the textual entity descriptions to generate latent representations of the entities and relations. In a KG, all the relations associated with an entity are not equally important. Therefore, a path selection technique is proposed to consider only the paths that contain the relations important to a certain entity. This assures that the structural contextual information generated using random walks provides relevant information about the entity. A novel encoder-decoder model is proposed to encapsulate the path information. Also, the contextual NLM is exploited to generate entity embeddings from the textual entity descriptions. The results show that the combination of these two features in the KG considerably improves the performance in link prediction compared to the existing baseline models. Furthermore, it is observed that the attention mechanism applied to the encoder-decoder model significantly uplifts the prediction results. One of the key benefits of the model is that it uses both the structural and textual information independently, enabling the generation of a latent representation of an entity for link prediction if either of the features is observed in the KG. It is to be mentioned that the proposed model is computationally expensive, however, the main focus is less on the runtime and more on the effectiveness of the model which is easily demonstrated from the results obtained.

This dissertation also studied if KGC can be achieved by fine-tuning the NLMs on KGs. In Chapter 6, a triple classification model is proposed in which GPT-2 is fine-tuned on triples from KG. The triples and entity descriptions from KGs are provided as inputs to the GPT-2 model and a binary classifier is trained on the last layer to identify if a given triple is true or not. Despite having more parameters than the BERT model, GPT-2 model does not outperform the KG-BERT model. In BERT and GPT-2, context-specificity presents very differently. It is observed in the literature that in BERT, two words in the same sentence are more dissimilar to each other in the upper layers but are still similar compared to two randomly sampled words. For GPT-2, the words in the same sentence are as dissimilar as randomly chosen words [38]. Therefore, this has an impact on the training of the triples. Also unlike BERT, it is a unidirectional decoder model, therefore it looks over for the tokens only in one direction during training as well as fine-tuning. However, it is also been observed in the results for triple classification that GPT-2 performs better for longer text compared to shorter ones. The proposed GPT-2 variant with textual entity descriptions as input performs better than the variant which uses only triples.

The second part of this dissertation examines the significance of Wikipedia categories, textual entity descriptions, and entity names for entity type prediction.

Chapter 7 explores the uncharted territory of Wikipedia categories and intends to find the answers to *C2-RQ1* from Section 1.2 of Chapter 1. By taking advantage of the links between the Wikipedia categories with regard to the shared entities, a novel category-category network embedding is presented. The NLMs are additionally used to generate text-based embeddings from the Wikipedia category labels. A framework for both multi-class and multi-label classification is proposed to predict the missing entity types. According to the experimental findings, the proposed model achieves SoTA results for entity typing on both benchmark datasets. Additionally, the model develops higher accuracy in predicting the types of missing data from unseen data. The major advantage of using the category-category network is that the number of categories is much lesser than the number of entities, thereby the computational complexity of graph traversing is less expensive. Furthermore, the pre-trained network embedding model can be leveraged to predict the missing types of new entities in a KG.

Chapter 8 studies the impact of the structural information and the textual entity descriptions to predict the missing entity types addressing the research questions mentioned in *C2-RQ2* and *C2-RQ3* from Section 1.2 of Chapter 1. Different strategic graph walks are proposed in this chapter to capture the contextual information of the entities in a KG. These graph walks are then used in a novel order-aware RDF2vec model to generate the entity embeddings that are used for predicting the missing types of entities. The main advantage of using different variants of the RDF2vec model over other embedding models is that the classical RDF2vec model performs better in class separation tasks compared to the other KG embedding models. The contextual NLMs are also exploited in this chapter for generating text embeddings. The classification

framework proposed in the previous Chapter 7, is extended with a new hierarchical classification model which predicts the types of entities at different levels of the hierarchy. The results show that the textual entity descriptions have a bigger impact on the classification than the structural information. However, the combination of both features outperforms the SoTA entity typing models both on coarse-grained as well as fine-grained classes.

Therefore, it can be inferred from the results that, to predict the missing types of entities which have no or less amount of property information in the KG, CAT2Type performs well over GRAND. The performance of CAT2Type is incredibly good for coarse-grained entity types. However, results show that for fine-grained entity typing GRAND performs better than CAT2Type. This is because the relation information of the entities captured in the random walks helps is encapsulated by the embedding model into the entity representation.

Finally, Chapter 9 focuses on predicting the missing types of the entities merely from their names finding solutions to the research question *C2-RQ4* from Section 1.2 of Chapter 1. The information contained in both the contextual and non-contextual NLMs is exploited to its full length to predict the missing types. The two main advantages of this model are **(i)** it does not require any extra information about the entity except for its name, and **(ii)** no training of the features is required. This model is evaluated for entities in different languages namely English, German, French, and Spanish. Furthermore, the evaluation shows that the trained classifier can be successfully used to predict missing types of entities from a different KG.

10.2 OPEN ISSUES AND OUTLOOK

It is anticipated that this dissertation will encourage different opportunities to pursue open challenges that have not been addressed so far. This section discusses the open issues of this thesis and possible directions for future work.

This thesis focuses on predicting the missing links within a KG. However, link prediction can also be performed across different KGs to predict the missing links between the two same entities across KGs which is also known as Entity Alignment. This can be achieved in two ways: **(i)** learning the embedding of the entities and the relations of the source and the target KGs separately followed by learning a supervised model to align the entities and the relations, **(ii)** joint learning of the embeddings of the two KGs into the unified space. The two proposed KG embedding models in this thesis *MADLINK*, and *GRAND* can be used as a base model to embed the KGs.

The methods proposed in this dissertation focus on structural contextual information, textual entity descriptions, Wikipedia categories, and entity names. However, a vast amount of unstructured information in the form of Wikipedia lists, tables, etc. still remains unexplored in the literature. One of the future research directions would be

incorporating information from these sources into the embedding models to predict the different kinds of missing links in a KG.

Another limitation of the thesis is that the proposed models are evaluated on benchmark datasets which are derived from open general-purpose KGs. Furthermore, open KG such as Wikidata has not been explored. In future, it would be interesting to analyse the performance of the proposed link prediction and entity type prediction models on domain-specific KGs such as Scholarly data, biomedical data etc. Furthermore, Chapter 9 throws some light on multilingual entity typing. The proposed entity typing models such as CAT2Type, and GRAND can be leveraged to predict the missing types of entities in different languages. The KG embeddings generated by exploiting different KG features can also be used for different NLP-based applications in future work. Furthermore, the proposed entity embeddings for entity typing methods can be used for predicting types of entities in automated KG creation.

In conclusion, it is anticipated that the contributions from this thesis will lead to rapid advancement in the techniques used for KG-based representation and its applications in different domains.

BIBLIOGRAPHY

[1] Mehwish Alam, Aleksey Buzmakov, Víctor Codocedo, and Amedeo Napoli. "Mining Definitions from RDF Annotations Using Formal Concept Analysis." In: *Twenty-Fourth International Joint Conference on Artificial Intelligence 2015*.

[2] Mohamed Ben Aouicha, Mohamed Ali Hadj Taieb, and Malek Ezzeddine. "Derivation of "is a" taxonomy from Wikipedia Category Graph." In: *Eng. Appl. Artif. Intell.* (2016).

[3] Nino Arsov and Georgina Mirceva. "Network embedding: An overview." In: *arXiv preprint arXiv:1911.11726* (2019).

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. "Dbpedia: A nucleus for a web of open data." In: *The semantic web*. 2007.

[5] Ivana Balažević, Carl Allen, and Timothy M Hospedales. "Hypernetwork knowledge graph embeddings." In: *Proceedings of the International Conference on Artificial Neural Networks*. 2019.

[6] Ivana Balažević, Carl Allen, and Timothy Hospedales. "TuckER: Tensor Factorization for Knowledge Graph Completion." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 2019.

[7] Russa Biswas, Mehwish Alam, and Harald Sack. "MADLINK: Attentive Multihop and Entity Descriptions for Link Prediction in Knowledge Graphs." In: *Semantic Web* (2022).

[8] Russa Biswas, Yiyi Chen, Heiko Paulheim, Harald Sack, and Mehwish Alam. "It's All in the Name: Entity Typing Using Multilingual Language Models." In: *The Semantic Web: ESWC 2022 Satellite Events Proceedings*. Vol. 13384. Lecture Notes in Computer Science. Springer, 2022, pp. 36–41.

[9] Russa Biswas, Jan Portisch, Heiko Paulheim, Harald Sack, and Mehwish Alam. "Entity Type Prediction Leveraging Graph Walks and Entity Descriptions." In: *Accepted at ISWC*. 2022.

[10] Russa Biswas, Radina Sofronova, Mehwish Alam, Nicolas Heist, Heiko Paulheim, and Harald Sack. "Do Judge an Entity by Its Name! Entity Typing Using Language Models." In: *The Semantic Web: ESWC 2021 Satellite Events Proceedings*. Vol. 12739. Lecture Notes in Computer Science. Springer, 2021, pp. 65–70.

[11] Russa Biswas, Radina Sofronova, Mehwish Alam, and Harald Sack. "Contextual Language Models for Knowledge Graph Completion." In: *Machine Learning with Symbolic Methods and Knowledge Graphs co-located with ECML PKDD*. CEUR Workshop Proceedings. 2021.

[12] Russa Biswas, Radina Sofronova, Harald Sack, and Mehwish Alam. "Cat2type: Wikipedia Category Embeddings for Entity Typing in Knowledge Graphs." In: *Proceedings of the 11th on Knowledge Capture Conference*. 2021, pp. 81–88.

[13] Peter Bloem, Xander Wilcke, Lucas van Berkel, and Victor de Boer. "kgbench: A Collection of Knowledge Graph Datasets for Evaluating Relational and Multimodal Machine Learning." In: *European Semantic Web Conference*. 2021.

[14] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching Word Vectors with Subword Information." In: *arXiv preprint arXiv:1607.04606* (2016).

[15] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching word vectors with subword information." In: *Transactions of the association for computational linguistics* 5 (2017), pp. 135–146.

[16] Kurt D. Bollacker, Robert P. Cook, and Patrick Tufts. "Freebase: A Shared Database of Structured General Human Knowledge." In: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*. 2007.

[17] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. "Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge." In: *Proceedings of the ACM SIGMOD international conference on Management of data*. 2008.

[18] Antoine Bordes, Sumit Chopra, and Jason Weston. "Question Answering with Subgraph Embeddings." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 2014.

[19] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. "Translating embeddings for modeling multi-relational data." In: *Proceedings of the Advances in neural information processing systems* (2013).

[20] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. "Learning structured embeddings of knowledge bases." In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. 2011.

[21] Alison Callahan, Jose Cruz-Toledo, Peter Ansell, and Michel Dumontier. "Bio2RDF release 2: improved coverage, interoperability and provenance of life science linked data." In: *Extended semantic web conference*. Springer. 2013, pp. 200–212.

[22] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. "Knowledge graph completion: A review." In: *Ieee Access* 8 (2020), pp. 192435–192456.

[23] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches." In: *Syntax, Semantics and Structure in Statistical Translation* (2014), p. 103.

[24] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 2014.

[25] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." In: *NIPS 2014 Workshop on Deep Learning, December 2014*. 2014.

[26] Luca Costabello, Sumit Pai, Chan Le Van, Rory McGrath, Nick McCarthy, and Pedro Tabacof. *AmpliGraph: a Library for Representation Learning on Knowledge Graphs*. 2019.

[27] Silviu Cucerzan. "Large-Scale Named Entity Disambiguation Based on Wikipedia Data." In: *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 2007.

[28] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. "A survey on network embedding." In: *IEEE transactions on knowledge and data engineering* 31.5 (2018), pp. 833–852.

[29] Tu Dinh Nguyen Dai Quoc Nguyen, Dat Quoc Nguyen, and Dinh Phung. "A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network." In: *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 2018.

[30] Andrew M Dai and Quoc V Le. "Semi-supervised sequence learning." In: *Advances in neural information processing systems* 28 (2015).

[31] Arjun Das, Debasis Ganguly, and Utpal Garain. "Named Entity Recognition with Word Embeddings and Wikipedia Categories for a Low-Resource Language." In: *ACM Trans. Asian Low Resour. Lang. Inf. Process.* (2017).

[32] Daniel Daza, Michael Cochez, and Paul Groth. "Inductive Entity Representations from Text via Link Prediction." In: *Proceedings of the Web Conference 2021*. 2021.

[33] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. "Convolutional 2d knowledge graph embeddings." In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[34] Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. "Convolutional 2D Knowledge Graph Embeddings." In: *Proceedings of the 32th AAAI Conference on Artificial Intelligence*. 2018.

[35] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2019.

[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2019.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *NAACL-HLT)*. 2019.

[38] Kawin Ethayarajh. "How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings." In: *arXiv preprint arXiv:1909.00512* (2019).

[39] Leonhard Euler. "Solutio problematis ad geometriam situs pertinentis." In: *Commentarii academiae scientiarum Petropolitanae* (1741), pp. 128–140.

[40] Jun Feng, Minlie Huang, Yang Yang, and Xiaoyan Zhu. "GAKE: Graph Aware Knowledge Embedding." In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016.

[41] Kunihiko Fukushima. "Neocognitron: A hierarchical neural network capable of visual pattern recognition." In: *Neural networks* 1.2 (1988), pp. 119–130.

[42] Alberto García-Durán and Mathias Niepert. "Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features." In: *arXiv preprint arXiv:1709.04676* (2017).

[43] Genet Asefa Gesese, Russa Biswas, Mehwish Alam, and Harald Sack. "A survey on knowledge graph embeddings with literals: Which model links better literally?" In: *Semantic Web* 12.4 (2021), pp. 617–647.

[44] Blerina Gkotse. "Ontology-based Generation of Personalised Data Management Systems: an Application to Experimental Particle Physics." PhD thesis. Université Paris sciences et lettres, 2020.

[45] Yoav Goldberg. "Neural network methods for natural language processing." In: *Synthesis lectures on human language technologies* 10.1 (2017), pp. 1–309.

[46] Daniel Graupe. *Principles of artificial neural networks*. Vol. 7. World Scientific, 2013.

[47] Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks." In: *22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016.

[48] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. "Jointly embedding knowledge graphs and logical rules." In: *Proceedings of the 2016 conference on empirical methods in natural language processing*. 2016, pp. 192–202.

[49] Ferras Hamad, Issac Liu, and Xian Xing Zhang. *Food Discovery with Uber Eats: Building a Query Understanding Engine. Uber Engineering Blog.* https://eng.uber.com/uber-eats-query-understanding/. Accessed: 2022-07-03. 2018.

[50] Nicolas Heist and Heiko Paulheim. "Uncovering the Semantics of Wikipedia Categories." In: *18th International Semantic Web Conference*. 2019.

[51] Nicolas Heist and Heiko Paulheim. "Entity Extraction from Wikipedia List Pages." In: *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*. Lecture Notes in Computer Science. Springer, 2020, pp. 327–342.

[52] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[53] J. Hoffart, M. A. Yosef, and I. Bordino et al. "Robust Disambiguation of Named Entities in Text." In: *Proceedings of the 2011 Conf. on Empirical Methods in Natural Language Processing*. 2011.

[54] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. "Knowledge graphs." In: *Synthesis Lectures on Data, Semantics, and Knowledge* 12.2 (2021), pp. 1–257.

[55] John J Hopfield. "Neurons with graded response have collective computational properties like those of two-state neurons." In: *Proceedings of the national academy of sciences* 81.10 (1984), pp. 3088–3092.

[56] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 328–339.

[57] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. "Knowledge graph completion with adaptive sparse transfer matrix." In: *Proceedings of the Thirtieth AAAI conference on artificial intelligence*. 2016.

[58] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. "A survey on knowledge graphs: Representation, acquisition, and applications." In: *IEEE Transactions on Neural Networks and Learning Systems* 33.2 (2021), pp. 494–514.

[59] Ningning Jia, Xiang Cheng, and Sen Su. "Improving Knowledge Graph Embedding Using Locally and Globally Attentive Relation Paths." In: *European Conference on Information Retrieval*. 2020.

[60] Hailong Jin, Lei Hou, Juanzi Li, and Tiansi Dong. "Attributed and Predictive Entity Embedding for Fine-Grained Entity Typing in Knowledge Bases." In: *27th International Conference on Computational Linguistics*. 2018.

[61] Hailong Jin, Lei Hou, Juanzi Li, and Tiansi Dong. "Fine-Grained Entity Typing via Hierarchical Multi Graph Convolutional Networks." In: *Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 2019.

[62] Kun Jing and Jungang Xu. "A survey on neural network language models." In: *arXiv preprint arXiv:1906.03591* (2019).

[63] Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. "Bag of Tricks for Efficient Text Classification." In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 2017, pp. 427–431.

[64] Carlos Nascimento Silla Jr. and Alex Alves Freitas. "A survey of hierarchical classification across different application domains." In: *Data Min. Knowl. Discov.* 22.1-2 (2011), pp. 31–72. DOI: 10.1007/s10618-010-0175-9. URL: https://doi.org/10.1007/s10618-010-0175-9.

[65] Mohammad Mehdi Keikha, Maseud Rahgozar, and Masoud Asadpour. "Community aware random walk for network embedding." In: *Knowledge-Based Systems* 148 (2018), pp. 47–54.

[66] Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. "Multi-task learning for knowledge graph completion with pre-trained language models." In: *Proceedings of the 28th International Conference on Computational Linguistics*. 2020.

[67] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." In: *5th International Conference on Learning Representations, ICLR 2017*. 2017.

[68] Philipp Koehn. *Statistical machine translation*. Cambridge University Press, 2009.

[69] Bhushan Kotnis and Vivi Nastase. "Analysis of the impact of negative sampling on link prediction in knowledge graphs." In: *arXiv preprint arXiv:1708.06816* (2017).

[70] Arun Krishnan. *Making search easier: How Amazon's Product Graph is helping customers find products more easily. Amazon Blog*. https://blog.aboutamazon.com/innovation/making-search-easier. Accessed: 2022-07-03. 2018.

[71] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. "Incorporating literals into knowledge graph embeddings." In: *Proceedings of the International Semantic Web Conference*. 2019.

[72] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. "Backpropagation applied to handwritten zip code recognition." In: *Neural computation* 1.4 (1989), pp. 541–551.

[73] John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. "Attention models in graphs: A survey." In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13.6 (2019), pp. 1–25.

[74] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. "A survey of convolutional neural networks: analysis, applications, and prospects." In: *IEEE transactions on neural networks and learning systems* (2021).

[75] Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. "Modeling Relation Paths for Representation Learning of Knowledge Bases." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015.

[76] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. "Learning entity and relation embeddings for knowledge graph completion." In: *Proceedings of the 29th AAAI conference on artificial intelligence*. 2015.

[77] Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. "Two/Too Simple Adaptations of Word2Vec for Syntax Problems." In: *NAACL HLT 2015*. ACL, 2015, pp. 1299–1304.

[78] Qi Liu, Matt J. Kusner, and Phil Blunsom. "A Survey on Contextual Embeddings." In: *CoRR* (2020).

[79] Qiaoling Liu, Kaifeng Xu, Lei Zhang, Haofen Wang, Yong Yu, and Yue Pan. "Catriple: Extracting triples from wikipedia categories." In: *Asian Semantic Web Conference*. 2008.

[80] Xiaofei Ma, Zhiguo Wang, Patrick Ng, Ramesh Nallapati, and Bing Xiang. "Universal text representation from bert: An empirical study." In: *arXiv preprint arXiv:1910.07973* (2019).

[81] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[82] A. Melo, H. Paulheim, and J. Völker. "Type Prediction in RDF Knowledge Bases Using Hierarchical Multilabel Classification." In: *WIMS*. 2016.

[83] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." In: *CoRR* (2013).

[84] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." In: *arXiv preprint arXiv:1301.3781* (2013).

[85] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. "Distributed Representations of Words and Phrases and their Compositionality." In: *NIPS*. 2013.

[86] George A Miller. "WordNet: a lexical database for English." In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[87] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[88] Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al. "A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

[89] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. "Holographic Embeddings of Knowledge Graphs." In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[90] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. "Holographic embeddings of knowledge graphs." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2016.

[91] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. "A three-way model for collective learning on multi-relational data." In: *Proceedings of the International Conference on Machine Learning*. 2011.

[92] Heiko Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods." In: *Semantic web* 8.3 (2017), pp. 489–508.

[93] Heiko Paulheim and Christian Bizer. "Type inference on noisy RDF data." In: *International semantic web conference*. Springer. 2013, pp. 510–525.

[94] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[95] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 701–710.

[96] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. "Don't Walk, Skip! Online learning of multi-scale network embeddings." In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. 2017, pp. 258–265.

[97] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 2227–2237.

[98] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. "Language Models as Knowledge Bases?" In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 2463–2473.

[99] Pouya Pezeshkpour, Liyan Chen, and Sameer Singh. "Embedding Multimodal Relational Data for Knowledge Base Completion." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018.

[100] Giuseppe Pirrò. "Explaining and Suggesting Relatedness in Knowledge Graphs." In: *Proceedings of the 14th International Semantic Web Conference*.

[101] R.J. Pittman, Srivastava. Amit, Sanjika Hewavitharana, Ajinkya Kale, and Saab Mansour. *Cracking the Code on Conversational Commerce. eBay Blog*. https://www.ebayinc.com/stories/news/cracking-the-code-on-conversationalcommerce/. Accessed: 2022-07-03. 2017.

[102] Simone Paolo Ponzetto and Michael Strube. "Taxonomy induction based on a collaboratively built knowledge repository." In: *Artificial Intelligence* (2011).

[103] Simone Paolo Ponzetto, Michael Strube, et al. "Deriving a large scale taxonomy from Wikipedia." In: *AAAI*. 2007.

[104] Jan Portisch, Michael Hladik, and Heiko Paulheim. "KGvec2go - Knowledge Graph Embeddings as a Service." In: *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*. European Language Resources Association, 2020, pp. 5641–5647.

[105] Jan Portisch, Michael Hladik, and Heiko Paulheim. "RDF2Vec Light - A Lightweight Approachfor Knowledge Graph Embeddings." In: *Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC 2020), Globally online, November 1-6, 2020 (UTC)*. Vol. 2721. CEUR Workshop Proceedings. CEUR-WS.org, 2020, pp. 79–84.

[106] Jan Portisch and Heiko Paulheim. "Putting RDF2vec in Order." In: *Proceedings of the ISWC 2021 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, October 24-28, 2021*. Ed. by Oshani Seneviratne, Catia Pesquita, Juan Sequeda, and Lorena Etcheverry. Vol. 2980. CEUR Workshop Pro-

ceedings. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2980/paper352.pdf.

[107] Jan Portisch and Heiko Paulheim. "Walk this Way! Entity Walks and Property Walks for RDF2vec." In: *CoRR* abs/2204.02777 (2022).

[108] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. "Pre-trained models for natural language processing: A survey." In: *Science China Technological Sciences* (2020).

[109] Alec Radford and Karthik Narasimhan. "Improving Language Understanding by Generative Pre-Training." In: 2018.

[110] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. "Language models are unsupervised multitask learners." In: *OpenAI blog* 1.8 (2019), p. 9.

[111] Yves Raimond, Tristan Ferne, Michael Smethurst, and Gareth Adams. "The BBC world service archive prototype." In: *Journal of web semantics* 27 (2014), pp. 2–9.

[112] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP*. 2019.

[113] Petar Ristoski and Heiko Paulheim. "Rdf2vec: Rdf graph embeddings for data mining." In: *Proceedings of the International Semantic Web Conference*. Springer. 2016.

[114] Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. "RDF2Vec: RDF graph embeddings and their applications." In: *Semantic Web* 10.4 (2019), pp. 721–752. DOI: 10.3233/SW-180317. URL: https://doi.org/10.3233/SW-180317.

[115] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[116] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. "Knowledge graph embedding for link prediction: A comparative analysis." In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* (2021).

[117] Afshin Sadeghi, Damien Graux, Hamed Shariat Yazdi, and Jens Lehmann. "MDE: Multiple Distance Embeddings for Link Prediction in Knowledge Graphs." In: *Proceedings of the European Conference on Artificial Intelligence*. 2020.

[118] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. "Modeling relational data with graph convolutional networks." In: *Proceedings of the European semantic web conference*. 2018.

[119] Jörg Schlötterer, Martin Wehking, Fatemeh Salehi Rizi, and Michael Granitzer. "Investigating Extensions to Random Walk Based Graph Embedding." In: *2019 IEEE International Conference on Cognitive Computing (ICCC)*. IEEE. 2019, pp. 81–89.

[120] Edward W Schneider. "Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis." In: (1973).

[121] S Selva Birunda and R Kanniga Devi. "A review on word embedding techniques for text classification." In: *Innovative Data Communication Technologies and Application* (2021), pp. 267–281.

[122] Baoxu Shi and Tim Weninger. "Open-world knowledge graph completion." In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[123] WANG Shuo, DU Zhijuan, and MENG Xiaofeng. "Research progress of large-scale knowledge graph completion technology." In: *Scientia Sinica Informationis* 50.4 (2020), pp. 551–575.

[124] Robert F Simmons. *Synthetic language behavior*. System Development Corporation, 1963.

[125] Amit Singhal. *Introducing the Knowledge Graph: things, not strings. Google Blog*. https://www.blog.google/products/search/introducing-knowledge-graph-things-not/. Accessed: 2022-07-03. 2012.

[126] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. "Reasoning with neural tensor networks for knowledge base completion." In: *Proceedings of the Advances in neural information processing systems*. 2013.

[127] Sampath Srinivas. "A generalization of the noisy-or model." In: *Uncertainty in artificial intelligence*. Elsevier. 1993, pp. 208–215.

[128] Fabian M Suchanek, Serge Abiteboul, and Pierre Senellart. "PARIS: Probabilistic Alignment of Relations, Instances, and Schema." In: *Proceedings of the VLDB Endowment* 5.3 (2011).

[129] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: a core of semantic knowledge." In: *Proceedings of the 16th international conference on World Wide Web*. 2007.

[130] 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, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. Ed. by Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy. ACM, 2007, pp. 697–706. DOI: 10.1145/1242572.1242667. URL: https://doi.org/10.1145/1242572.1242667.

[131] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: A large ontology from wikipedia and wordnet." In: *Journal of Web Semantics* 6.3 (2008), pp. 203–217.

[132] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. "RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space." In: *Proceedings of the 7th International Conference on Learning Representations*. 2019.

[133] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks." In: *Proceedings of the Advances in neural information processing systems*. 2014.

[134] Yi Tay, Luu Anh Tuan, Minh C Phan, and Siu Cheung Hui. "Multi-task neural network for non-discrete attribute prediction in knowledge graphs." In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017.

[135] Avijit Thawani, Jay Pujara, Pedro A Szekely, and Filip Ilievski. "Representing Numbers in NLP: a Survey and a Vision." In: *arXiv preprint arXiv:2103.13136* (2021).

[136] Peihao Tong, Qifan Zhang, and Junjie Yao. "Leveraging domain context for question answering over knowledge graph." In: *Data Science and Engineering* (2019).

[137] Kristina Toutanova and Danqi Chen. "Observed versus latent features for knowledge base and text inference." In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. 2015.

[138] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is All you Need." In: *Proceedings of the Advances in Neural Information Processing Systems*. 2017.

[139] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In: *Advances in neural information processing systems* 30 (2017).

[140] Denny Vrandečić and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase." In: *Communications of the ACM* (2014).

[141] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. "Knowledge graph embedding by translating on hyperplanes." In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. 2014.

[142] Zhigang Wang, Juanzi Li, Zhiyuan Liu, and Jie Tang. "Text-enhanced representation learning for knowledge graph." In: *Proceedings of International Joint Conference on Artificial Intelligent (IJCAI)*. 2016, pp. 4–17.

[143] Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. "Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances." In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 2015, pp. 1331–1340.

[144] Paul Werbos. "Beyond regression:" new tools for prediction and analysis in the behavioral sciences." In: *Ph. D. dissertation, Harvard University* (1974).

[145] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. "Knowledge base completion via search-based question answering." In: *Proceedings of the 23rd international conference on World wide web*. 2014, pp. 515–526.

[146] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. "Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction." In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013.

[147] Yanrong Wu and Zhichun Wang. "Knowledge Graph Embedding with Numeric Attributes of Entities." In: *Proceedings of the Rep4NLP@ACL*. 2018.

[148] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. "SSP: semantic space projection for knowledge graph embedding with text descriptions." In: *Proceedings of the Thirty-First AAAI conference on artificial intelligence*. 2017.

[149] Ruobing Xie, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. "Image-embodied knowledge representation learning." In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2017.

[150] Ruobing Xie, Zhiyuan Liu, Maosong Sun, et al. "Representation Learning of Knowledge Graphs with Hierarchical Types." In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 2016.

[151] Bo Xu, Yi Zhang, Jiaqing Liang, Yanghua Xiao, Seung-won Hwang, and Wei Wang. "Cross-Lingual Type Inference." In: *Database Systems for Advanced Applications - 21st International Conference, DASFAA*. 2016.

[152] Hu Xu, Bing Liu, Lei Shu, and S Yu Philip. "BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 2324–2335.

[153] Jiacheng Xu, Xipeng Qiu, Kan Chen, and Xuanjing Huang. "Knowledge graph representation with jointly structural and textual encoding." In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2017.

[154] Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. "Corpus-Level Fine-Grained Entity Typing." In: *J. Artif. Intell. Res.* (2018).

[155] Yadollah Yaghoobzadeh and Hinrich Schütze. "Multi-level Representations for Fine-Grained Typing of Knowledge Base Entities." In: *15th Conference of the European Chapter of the Association for Computational Linguistics*. 2017.

[156] Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. "Wikipedia2vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from wikipedia." In: *arXiv preprint arXiv:1812.06280* (2018).

[157] 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 3rd International Conference on Learning Representations*. 2015.

[158] Liang Yao, Chengsheng Mao, and Yuan Luo. "KG-BERT: BERT for knowledge graph completion." In: *arXiv preprint arXiv:1909.03193* (2019).

[159] Dong Yu and Li Deng. *Automatic Speech Recognition.* Springer, 2016.

[160] SHUAI ZHANG, Yi Tay, Lina Yao, and Qi Liu. "Quaternion Knowledge Graph Embeddings." In: 2019.

[161] Xiang Zhang and Yann LeCun. "Text understanding from scratch." In: *arXiv preprint arXiv:1502.01710* (2015).

[162] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. "Character-level Convolutional Networks for Text Classification." In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*. 2015.

[163] Peixiang Zhao, Charu Aggarwal, and Gewen He. "Link prediction in graph streams." In: *Proceedings of the IEEE 32nd International Conference on Data Engineering*. 2016.

[164] Yu Zhao, Anxiang Zhang, Ruobing Xie, Kang Liu, and Xiaojie Wang. "Connecting Embeddings for Knowledge Graph Entity Typing." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 6419–6428.

[165] Jianhuan Zhuo, Qiannan Zhu, Yinliang Yue, Yuhong Zhao, and Weisi Han. "A Neighborhood-Attention Fine-grained Entity Typing for Knowledge Graph Completion." In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 2022, pp. 1525–1533.

[166] Amal Zouaq and Felix Martel. "What is the schema of your knowledge graph? leveraging knowledge graph embeddings and clustering for expressive taxonomy learning." In: *Proceedings of the international workshop on semantic big data*. 2020, pp. 1–6.

DECLARATION

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, February 27, 2023

M.Sc. RUSSA BISWAS