NEURAL METHODS FOR ENTITY-CENTRIC KNOWLEDGE EXTRACTION AND REASONING IN NATURAL LANGUAGE

By

RAJARSHI BHOWMIK

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Prof. Gerard de Melo

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

October 2021

ABSTRACT OF THE DISSERTATION

Neural Methods for Entity-Centric Knowledge Extraction and Reasoning in Natural Language

by **Rajarshi Bhowmik**

Dissertation Director: Prof. Gerard de Melo

Entities are the cornerstone for the dissemination of factual knowledge in natural language. Human verbal and written communication invariably refer to entities, their properties, and their relationships. Moreover, human reasoning often relies on a proper understanding of relationships among various entities. It is perceived that representing entity-centric knowledge in a structured form is most suitable for machines to consume and reason about it. Over the past few decades, numerous methodological advances have been made in extracting entity-centric knowledge from unstructured and semi-structured sources, representing entity-centric knowledge as graph-structured data known as *Knowledge Graphs*, and using these knowledge graphs in various knowledge-intensive natural language processing tasks. Despite these advances, machines are yet to achieve human-level ability to extract and reason with factual knowledge and use it for various knowledge-intensive tasks. This dissertation proposes novel neural methods to narrow this gap. In particular, for *factual knowledge extraction*, it proposes efficient and effective methods for the tasks of Entity Linking and Relation Extraction. For *knowledge-based logical reasoning*, an explainable link prediction method for emerging entities in knowledge graphs is proposed. Furthermore, as a representative of knowledge-intensive natural language processing tasks, this dissertation studies the problem of *entity summarization* to retrieve relevant facts and generate fact-allegiant textual descriptions of entities.

To Sri Sri Thakur Anukulchandra,

whose supreme messages, The Pursuit of Truth, have kept me ever inspired.

ACKNOWLEDGMENTS

I express my sincere gratitude to my adviser Prof. Gerard de Melo, for guiding me throughout this journey. With his profound knowledge and expertise, he has persistently helped me to improve as a researcher. I am immensely grateful to him for giving me absolute freedom in pursuing the research topics of my choice. I am also extremely grateful to Prof. Karl Stratos for his advising. I thank him for making himself available for discussing various project ideas with me. I have learned a lot about research methods from him. A part of this dissertation is completed under his active guidance. I am grateful to Prof. Matthew Stone and Prof. Sameer Singh for being a part of my dissertation committee and providing insightful feedback.

I am also thankful to the Department of Computer Science, Prof. Gerard de Melo, and Mike Tung (CEO, Diffbot) for funding my research. I thank Diffbot, Google, and Prof. Karl Stratos for providing me computing resources to carry out my research work.

It was an absolute pleasure to work with excellent collaborators like Atharva Prabhat Paranjpe, Sreyasi Nag Chowdhury, and Hareesh Ravi. I thank them for engaging in fruitful collaborations on various projects. I also thank the administrative staffs, the faculty members, and my colleagues at Rutgers for making this journey a pleasant experience.

I express my gratitude to Prof. Murulidhar N. N., Prof. Keshava Prasad Halemane, and Prof. Robert John D'Souza, who had encouraged me to pursue a career in research. I am thankful to my high school teachers Arup Kumar Das and Baneswar Das, who kindled the light of knowledge in me and encouraged me to pursue my dreams.

Finally, I would like to thank my parents, Bidyut and Ila Bhowmik, and my late aunt Maya for strongly supporting all my endeavors and showering their love and blessings on me. Last but not the least, I thank my dear friends Vipina K. Keloth, Anindya Das, and Ayan Mandal for their companionship through every high and low of this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Human languages in both written and spoken forms are built around semantic relatedness of entities, their properties, and their relationships. For machines to interpret human languages as effortlessly as humans do, the machines must possess the ability to 'understand' the underlying semantic structures of entities and their relationships. It is often perceived that equipping machines with world knowledge of entities and their relationships is a key component of Artificial Intelligence [1]. A predominant approach towards this goal is constructing structured repositories of factual knowledge containing entities and their relationships, also known as *Knowledge Graphs* [2].

Over the last few decades, several open-domain and domain-specific knowledge graphs have emerged driven by Semantic Web technologies. For example, existing open-domain knowledge graphs such as DBpedia [3], YAGO [4], Wikidata [5], etc. contain factual knowledge about millions of entities of various types such as a person, organization, location, etc. Domain-specific knowledge graphs such as Amazon Product Graph, UMLS [6], etc. contain factual knowledge from a particular domain such as e-commerce and biomedicine. Many of these knowledge graphs are curated by experts (e.g., UMLS), extracted from semi-structured sources such as Wikipedia infobox (e.g., DBpedia, YAGO), or crowdsourced (e.g., Wikidata). Despite being large-scale repositories of factual knowledge, these knowledge graphs contain only a fraction of world knowledge. The Web predominately has data in textual form, and thus, a vast amount of factual knowledge is hidden in these textual data. The key challenge, therefore, is to extract factual knowledge from these textual data automatically.

In recent years, knowledge extraction from texts has gained momentum due to the advances in deep neural models. Information extraction techniques such as Named Entity

Recognition (NER), Entity Linking (EL), and Relation Extraction (RE) are widely used for the automatic extraction of factual knowledge from texts. Although pre-trained language models have excelled in some of these tasks, there is always a trade-off between the efficiency and effectiveness of these models. In this dissertation, an efficient yet effective neural method for these tasks is explored. Besides, many of the existing models rely on pipelined execution of these tasks which is susceptible to error propagation. To alleviate this issue, this dissertation explores end-to-end differentiable neural models for these tasks.

Despite the recent advances in extraction of factual knowledge from various structured, unstructured, and semi-structured sources, most of the open-domain knowledge graphs inherently suffer from incompleteness and sparsity due to the open-world assumption. For example, it might be possible to extract facts such as (*Joe Biden, position held, President of the United States*) from the text *"Joe Biden becomes the 46th president of the United States"*, but it may not be possible to extract Joe Biden's occupation only from the text. Thus, to answer questions such as *"What is Joe Biden's occupation?*, an intelligent system should be able to derive the fact that Joe Biden is a politician since the president of a country is usually a politician. Such logical reasoning ability is essential to knowledge graph completion as it enables answering any query that might not be directly available in the knowledge graph.

Despite recent progress in the knowledge graph completion tasks such as *link prediction* and *relation prediction*, there are various challenges to be solved. Existing knowledge graph completion methods perform link prediction only for a static snapshot of the knowledge graph (transductive setting) and often lack explainability. This dissertation proposes neural methods to deal with these two challenges. The proposed model performs link predictions for emerging entities in knowledge graphs using inductive representation learning and makes the link prediction process interpretable to humans through multi-hop reasoning.

Various knowledge-intensive natural language processing tasks rely on knowledge graphs as a source of factual knowledge. One such use case is the *semantic search* used by mod-

ern search engines. Entity-centric search queries that solicit specific factual information about an entity constitute a significant proportion of all search queries processed by popular search engines. Almost all modern search engines incorporate facts from an underlying knowledge graph that acts as a reliable source of factual information. However, it is crucial to render only the most relevant information about an entity aligned with the user's specific information needs. To this end, this dissertation studies the tasks of *entity summarization* and *entity description generation* that are deemed necessary in the context of semantic search and are representatives of knowledge-intensive natural language processing tasks. *Entity summarization* ensures retrieval of relevant facts pertaining to a query. For example, a search engine query such as *einstein education* ought to give preference to different facts than a query such as *einstein family*.

On the other hand, given a set of facts about an entity, *entity description generation* generates concise natural language texts that precisely entail those facts. For example, given the facts (*Beethoven, born in, Bonn*), (*Beethoven, occupation, composer and pianist*), and (*Bonn, located in, Germany*), an AI system should also be able to generate *"German-born composer and pianist."* to describe Beethoven. Such textual description often helps to discern the most important factual information that distinctively identifies an entity. Additionally, these descriptions can serve as fine-grained semantic types of the entities and help in named entity disambiguation.

1.1 Contributions

The main contributions of this dissertation are summarized in the following.

Knowledge Extraction This dissertation explores end-to-end differentiable models for extracting entity-centric knowledge from text. The contributions here are two-fold. (1) Many prior works have explored modular approaches for knowledge extraction from text. These methods deploy a pipeline of modules in which each module is specialized to per-

form a subtask such as mention span detection, candidate generation, entity disambiguation, and relation extraction. Although many of these models have been deployed for practical purposes, their major disadvantage is the error propagation that occurs when one of the pipeline's modules makes an incorrect prediction. To alleviate this issue, several recently proposed methods learn to perform a combination of these subtasks jointly [7]. This dissertation extends this line of work on multi-task learning to perform mention span detection, candidate generation, entity disambiguation, and relation extraction by training a single end-to-end differentiable BERT-based dual encoder model. (2) Recent advancements in entity linking using BERT-based models follow a *retrieve and rerank* paradigm [8, 9], where the candidate entities are first selected using a retriever model. Then the retrieved candidates are ranked by a reranking model. While this paradigm produces state-of-the-art results, they are slow both at training and inference time as they can process only one mention at a time. This dissertation proposes a BERT-based dual encoder model that can perform both retrieval and reranking of candidate entities and resolves multiple mentions in a document in one shot to mitigate these issues. The proposed model is multiple times faster than existing BERT-based models while being competitive in accuracy for the entity linking task.

Inductive Knowledge Graph Representation Learning and Explainable Reasoning Recent approaches to knowledge graph completion can broadly be classified into two categories: embedding-based methods [10], and path-based methods [11, 12]. In contrast to embedding-based methods, which operate in an uninterpretable latent semantic vector space of entities and relations, path-based methods operate in the symbolic space, making the inference process explainable. Traditionally, link prediction methods are studied with static snapshots of the knowledge graphs, thus severely restricting their applicability for a dynamic knowledge graph with many emerging entities. This dissertation proposes a joint model for representation learning and reasoning in knowledge graphs that aims at achiev-

ing inductive node representation learning capabilities applicable to a dynamic knowledge graph with many emerging entities while preserving the unique advantage of the path-based approaches in terms of explainability. For inductive node representation learning, the model uses a variant of *Graph Transformer* encoder that aggregates neighborhood information based on their relevance to the query relation. For explainability, the model deploys policy gradient-based reinforcement learning (REINFORCE [13]) to decode a reasoning path to the answer entity. Additionally, three benchmark datasets are introduced for link prediction in inductive settings.

Knowledge Retrieval for Entity Summarization Entity summarization requires the ranking of facts about an entity in a knowledge graph for a user's search query based on their *importance* and *relevance* or a combination of both. This dissertation proposes a cross-attention model with a pointwise and a pairwise loss function to address query-dependent fact retrieval for entity-centric search queries. In contrast to existing feature-based models that rely on various forms of statistical and ontological features extracted from a large-scale knowledge graph, the proposed model draws on recent advances in Transformers with self-attention [14]. It leverages the linguistic connection between the query and the candidate facts and can be applied even to entirely novel sets of candidate facts to answer ad-hoc search queries.

Fact-Allegiant Entity Description Generation Generating factually correct entity descriptions is a two-step process. First, a model needs to select the most discernible facts that can uniquely identify an entity and then generate a succinct textual description of the entity using the selected facts. This dissertation proposes two methods for generating textual descriptions from factual knowledge. First, a dynamic memory network-based model is proposed that performs adequately in generating short textual descriptions of Wikidata entities. However, the main limitation of this approach is its inability to generate out-of-vocabulary words in the output description. To mitigate this issue, a fact-to-sequence

encoder-decoder model is proposed that is equipped with an explicit copy mechanism. The ability to copy out-of-vocabulary words significantly improves the factual fidelity of the generated descriptions. Additionally, two new datasets are curated for this task which are publicly available to foster future research in this direction.

1.2 Dissertation Outline

The remaining chapters of this dissertation are organized as follows. Chapter 2 presents an overview of the prior work and relevant methods. Chapter 3 explores entity linking and relation extraction methods for extracting factual knowledge from texts. Chapter 4 contains a novel method for inductive representation learning and explainable reasoning in knowledge graphs. Chapter 5 presents an entity summarization method that improves retrieving relevant knowledge graph facts pertaining to a search query. Chapter 6 introduces a novel task of generating short textual descriptions from knowledge graph facts. Concluding remarks and a discussion on future research directions are presented in Chapter 7.

CHAPTER 2
BACKGROUND

This chapter contains a brief overview of Knowledge graphs, various methods for knowledge graph representation learning and reasoning, and methods for knowledge extraction. Additionally, this chapter introduces the relevant deep neural methods that are used in this dissertation.

2.1 Knowledge Graphs

Knowledge graphs are structured repositories of entity-centric factual knowledge containing entities, properties, and relationships among entities in a graph data structure [2]. Formally, a knowledge graph is defined as a directed multi-graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ such that each node $e \in \mathcal{E}$ represents an entity, each $r \in \mathcal{R}$ represents a unique relation or property, and each directed edge is a triple $(e_h, r, e_t) \in \mathcal{F}$ that represents a fact about entity e_h. In existing literature, entities e_h and e_t are often referred as *head entity* and *tail entity*, and relation $r \in \mathcal{R}$ is often called a *predicate*. In some cases, the tail entity e_t can also be a *literal*. For example, a person's birth date can be denoted as a triple *(Albert Einstein, born on, March 14, 1879)* where the tail entity is a date. Note that the above notation is strictly applicable to binary relations. In the case of N-ary relations, *reification* can be used to map them to binary relations. Some knowledge graphs mandate the use of a predefined set of predicates (relations) and a predefined set of semantic types for entities. Such knowledge graphs are called *schema-based* knowledge graphs. Much of the work in this dissertation assumes schema-based knowledge graphs unless stated otherwise.

The pioneering works on knowledge graphs are Cyc [15] and WordNet [16]. However, Cyc did not focus on explicit entity-centric knowledge that can be extracted from text. Rather, it was intended to capture implicit *common sense knowledge* to help AI agents

learn basic concepts and rules about how the world works. On the other hand, WordNet is a large lexical database of English words interlinked through conceptual-semantic and lexical relations. Contemporary to Cyc and Wordnet, several domain-specific ontologies also emerged. For example, UMLS [6] is a large meta-thesaurus in the biomedical domain that integrates information from over 200 biomedical source vocabularies that cumulatively account for millions of biomedical concepts. Most of these early knowledge graphs were curated manually by experts.

Since the 2000s, driven by the grand vision of *Semantic Web*, several large-scale open-domain knowledge graphs have been developed. There was also a significant paradigm shift in the approach of constructing these knowledge graphs. Instead of being hand-crafted by experts, these knowledge graphs were populated using automatic and semi-automatic methods. These methods help to extract knowledge from structured (e.g., WordNet), semi-structured (e.g., tabular data), and unstructured data (e.g., free form text) in the Web. For example, DBpedia [3] is constructed by extracting triples from Wikipedia infobox. YAGO [4] is automatically extracted from Wikipedia, WordNet, and other sources. NELL [17] was initialized with a set of basic semantic relationships between a few hundred predefined categories of entities and is constantly crawling web pages and collecting new factual knowledge automatically. MENTA [18] induced a large-scale multilingual entity taxonomy by leveraging Wikipedia articles, categories, infoboxes, and WordNet synsets from multiple languages.

Large-scale knowledge graphs such as Freebase (now obsolete) and Wikidata [5] adopted a different knowledge graph construction approach. Instead of manual curating or automatic and semi-automatic extraction, these knowledge graphs rely on community contributions to enrich the factual coverage. Due to its crowdsourcing approach, Wikidata expanded rapidly in size and, as of March 2021, contains factual knowledge for more than 93 million entities.

In the past decade, several technology companies have adopted knowledge graphs for a

variety of commercial applications. For example, Google Knowledge Graph, Microsoft Satori, Baidu Knowledge Graph have become an integral part of their search engines and virtual assistants. Amazon Product Graph, Alibaba e-Commerce Graph, etc. help in product search and recommendations in their respective e-commerce platforms. Besides, domain-specific knowledge graphs have emerged in finance, transportation, life sciences, etc. to support a variety of commercial and non-commercial applications.

2.2 Knowledge Graph Representation Learning

Since knowledge graphs are built to support machine intelligence, they are increasingly being used in various predictive and generative machine learning models. Various natural language processing tasks integrate knowledge graphs as a source of factual knowledge. Some examples are semantic search, open-domain question answering, fact checking, dialog generation, summarization, etc. [19]. Over the past decade, neural methods have been overwhelmingly successful in many of these tasks. However, knowledge graphs are symbolic and cannot be directly integrated into these neural models that operate on continuous and differentiable vector space. To solve this problem, knowledge graphs are mapped from discrete symbolic representations to distributed vector representations. Over the past decade, several methods have been proposed for learning knowledge graph representation, also popularly known as *knowledge graph embeddings*. The goal of knowledge graph representation learning is to learn low-dimensional distributed vector representations (embeddings) of entities and relations so that they preserve the semantics and inherent structure of knowledge graphs such as symmetry, antisymmetry, inversion, composition, semantic hierarchies, etc. The embeddings space can be broadly classified as Euclidean and non-Euclidean vector space.

Euclidean space is widely adopted for representing entities and relations in various knowledge graph embedding methods. For example, TransE [20] represents the head and tail entities of a triple (e_h, r, e_t) as d-dimensional vectors $\mathbf{e_h}, \mathbf{e_t} \in \mathbb{R}^d$ in the Euclidean space

and the relation r as a translation vector $\mathbf{r} \in \mathbb{R}^d$ such that $\mathbf{e_h} + \mathbf{r} \approx \mathbf{e_t}$. This method inspired many other translational models such as TransR [21], TransH [22], etc. TransR maps entities and relations to different vector space. TransH maps entities to relation-specific hyperplanes. These models can represent inversion and compositional patterns in knowledge graphs. Bilinear models such as DistMult [23] can represent symmetry in relations. To capture both symmetry and antisymmetry, ComplEx [24] extended the DistMult model by mapping entities and relations into the complex vector space where $\mathbf{e_h}, \mathbf{e_t}, \mathbf{r} \in \mathbb{C}^d$ and $\mathbf{e_h} = \mathrm{Re}(\mathbf{e_h}) + i\mathrm{Im}(\mathbf{e_h})$. RotatE [25] models relations as a rotation from the head entity to the tail entity in the complex vector space such that $\mathbf{e_t} \approx \mathbf{e_h} \circ \mathbf{r}$. RotatE captures relation patterns inversion and composition as well as symmetry and antisymmetry in the learned representations. HAKE [26] captures the semantic hierarchy of entities by mapping them into the polar coordinate system where the radial coordinate corresponds to hierarchy level and the polar coordinate corresponds to various semantic types in the same level of the hierarchy.

In recent years, convolutional neural networks (CNN) have been successfully used for knowledge graph representation learning. Examples of this approach are ConvE [27] and ConvKB [28]. ConvE models the interations between head entity e_h and relation r by reshaping the embeddings $\mathbf{e_h} \in \mathbb{R}^d$ and $\mathbf{r} \in \mathbb{R}^d$ into 2D tensors $\mathbf{M_h} \in \mathbb{R}^{d_w \times d_h}$ and $\mathbf{M_r} \in \mathbb{R}^{d_w \times d_h}$ where $d = d_w \times d_h$, and then apply 2D convolutional kernels and other non-linear functions to obtain a score for each triple (e_h, r, e_t) as follows

$$f(e_h, r, e_t) = \psi(\mathrm{vec}(\psi([\mathbf{M_h}, \mathbf{M_r}] * \omega))\mathbf{W})\mathbf{e_t} \tag{2.1}$$

where ψ is a non-linear activation function such as Rectified Linear Unit (ReLU) and ω is the convolutional kernels. While ConvE aims to maximize interactions between head entity and relation by 2D reshaping, ConvKB stacks the embeddings of head entity, relation, and tail entity and then applies convolutional filters.

Since knowledge graphs are graph-structured data, recent advances in graph representation learning using graph neural networks have been adopted for knowledge graph representation learning. Graph Convolution Networks (GCN) [29] are widely used for graph representation learning, particularly for homogeneous graphs. GCNs are instances of Message Passing Neural Networks (MPNN), in which the node representations are learned by aggregating information from the nodes' local neighborhood. A key disadvantage of GCN is its large memory footprint as the entire graph needs to be loaded and kept in the memory for this model to work, which prohibits this model's application to large-scale graphs. GraphSAGE [30] attempts to reduce the memory footprint of GCN by random sampling of the neighborhood. Graph Attention Networks (GAT) [31] is a variant of GCN that learns node representations as weighted averages of the neighborhood information. However, GCN and its variants, such as GAT and GraphSAGE, are not directly applicable for knowledge graph representation learning. They ignore the edge (relation) information for obtaining the node embeddings. To alleviate this issue, Schlichtkrull et al. proposed R-GCN [32] that operate on relational multi-graphs. However, similar to GCN, R-GCN also needs all nodes of the graphs to be present in memory and therefore are not scalable to large-scale knowledge graphs. Hamaguchi et al. [33] proposed a model for computing representations for out-of-KG entities using graph neural networks. The recent models such as SACN [34], and CompGCN [35] leverage the graph structure by inductively learning representations for edges (relations) and nodes (entities). More details on graph representation learning are given in Subsection 2.6.2 of this chapter, and a comprehensive survey of knowledge graph representation learning is provided by Ji et al. [36].

2.3 Knowledge Graph Reasoning

Knowledge graph representation learning methods have been successfully applied to knowledge graph completion tasks, such as *link prediction* and *relation prediction*. The link prediction task requires a model to predict the tail entity, given a head entity and a relation.

On the other hand, the relation prediction task requires a model to predict the relation between a head and a tail entity. Both tasks require a fair amount of reasoning ability which is implicitly done by many of the knowledge graph representation learning methods. Despite the embedding-based models' success, they provide limited human interpretable reasoning to support link prediction and relation prediction.

An alternative stream of research has explored means of identifying specific reasoning paths. To this end, the Path Ranking Algorithm (PRA) [37] uses random walks with restarts for multi-hop reasoning. Following PRA, other approaches [38, 39, 40, 41] also leverage random walk based inference. However, the reasoning paths that these methods follow are gathered by random walks independently of the query relation.

Recent approaches have instead adopted policy gradient based reinforcement learning for a more focused exploration of reasoning paths. Policy gradient based models such as DeepPath [42], MINERVA [43], MultiHop [12], and M-Walk [44] formulate the KG reasoning task as a Partially Observable Markov Decision Process and learn a policy conditioned on the query relation. The underlying principles of these approaches are described in Subsection 2.6.3.

Another sub-category of path-based methods, e.g., AMIE+ [45], AnyBURL [46], and RuleS [47] proceed by mining Horn rules from the the existing knowledge graphs for link prediction. The body of a Horn rule provides the reasoning path. Although these approaches are capable of fast rule mining, the quality of the learned rules are affected by the sparsity of the knowledge graph.

2.4 Automatic Knowledge Extraction

Many large-scale domain-specific and open-domain knowledge graphs contain millions of entities and billions of facts due to community contributions and expert annotations. These knowledge graphs are reliable as they provide high precision factual knowledge about entities. However, they lack recall as most of these knowledge graphs do not have enough

facts about the so-called *tail entities*. The Web is a rich source of factual knowledge and can be leveraged to increase the knowledge graphs' coverage. The factual knowledge in the web contents can be in semi-structured forms such as tables, lists, and DOM trees. Many existing open-domain knowledge graphs, including DBpedia and YAGO, uses automatic knowledge extraction methods to extract factual knowledge from semi-structured data. However, the Web predominantly consists of textual data. Thus, a large amount of factual knowledge is available in text format. Extracting factual knowledge from texts can potentially enhance the knowledge graphs' factual coverage. Several existing knowledge graphs, including NELL and YAGO, extract factual knowledge from texts using a combination of various techniques discussed in the following.

Some earliest approaches in knowledge extraction from texts rely on lexico-syntactic pattern matching. A fixed pattern such as a regular expression is used to locate specific facts in texts in these approaches. The earliest method in this direction is the *Hearst pattern* [48] which was proposed to extract hyponym relations between entities. Hearst pattern has been extended for the task of *relation extraction* that identifies semantic relations among entities. These hand-crafted patterns have the advantage of high precision. However, they often suffer from low recall.

Semi-supervised learning approaches automatically discover new patterns after they are initiated with a few high-precision seed patterns. This approach is often called *Bootstrapping*. Bootstrapping starts with finding sentences in text documents that contain the same pair of entities as in the seed patterns and then extracts new patterns from all such sentences. NELL is built using such a semi-supervised learning approach.

The supervised learning paradigm has also been widely adopted for knowledge extraction. In this approach, a gold standard dataset is prepared with manual annotations of entity mentions and their relations. In some datasets, the entity mentions are also canonicalized by mapping them to a predefined set of entities in a knowledge graph – a task known as *entity linking*. A model should perform the tasks of *mention span detection* to detect the

spans of entities in the input text and *relation classification* to identify any potential relation(s) between a pair of entity mentions. These models can also perform entity linking to map mention spans to their corresponding canonical form in a target knowledge graph.

A key disadvantage of supervised knowledge extraction is that it requires a large amount of annotated training data that is expensive to obtain. To this end, distant supervision methods combine the advantages of bootstrapping with supervised learning [49]. Instead of a handful of seed examples, distant supervision methods use an existing knowledge graph as seed examples. For each existing triple in the knowledge graph, this method then follows the same approach as bootstrapping to identify sentences that mention the head and tail entities of the triple. This approach collects many sentences that are potentially noisy but can be refined by using supervised methods.

The unsupervised learning approach has also been successful for knowledge extraction. The unsupervised approach is often called Open Information Extraction (OpenIE) [50]. In this approach, sets of entities and relations are not predefined. Rather, both entities and relations are captured in their surface forms without canonicalization. For example, ReVerb [51] uses the subject and object phrases in a sentence as the head and tail entities and the verb phrase as the relation between them. ClausIE [52] derives triples by analyzing the clauses in a sentence. MinIE [53] advanced ClausIE by making the extracted triples more concise.

2.5 Entity Summarization

Entities in knowledge graphs usually have tens of associated facts. However, an entity can be uniquely identified with only a handful of salient facts. Entity summarization is the task of identifying these salient descriptor facts. Many existing methods for entity summarization are extractive methods that select a size-constrained subset of facts from the set of all facts of an entity. It requires ranking the set of facts for an entity regarding its salience. Existing methods [54, 55, 56, 57] consider the task of ranking as selecting the

most important facts about an entity, typically to compile these into a concise listing that summarizes "important" information about the said entity. Thus, these works primarily focus on some general notion of fact importance as the basis of ranking.

In recent years, knowledge graphs are widely used in semantic search. Modern-day web search engines such as Google, Bing, and Baidu use knowledge graphs to answer entity-centric search queries that solicit specific factual information about entities. The retrieved facts ought to be relevant to the search query to meet users' specific information needs. Therefore, existing approaches of entity summarization that primarily focus on the notion of importance are not suitable. To this end, Hasibi et al. [58] proposed a supervised model of fact ranking that considers both the importance and relevance of facts concerning a given query. In Chapter 6 of this dissertation, Hasibi et al.'s line of work is extended further using pre-trained language models and a pairwise ranking objective.

2.6 Relevant Methods

This section outlines some methods that are used throughout the dissertation.

2.6.1 Text Representation

Texts are ordered sequences of discrete symbols called tokens. A token can be a word or a word-piece (sub-word). These discrete tokens are mapped to continuous vectors representation using Word2Vec [59] or other representation learning methods (e.g., GloVe [60], ELMo [61], etc.). Additionally, several neural models have been proposed to learn contextual representations of tokens in a sequence. Given an input sequence $S = w_1, w_2, \ldots, w_N$, these models learn contextual representations $H = \mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_N}$ such that $\mathbf{h_i} \in \mathbb{R}^d$. In the following, two popular neural architectures for representation learning are presented.

Recurrent Neural Network A recurrent neural network learns to represent a token in a sequence based on the input token's embeddings and the current hidden state of the network

that encodes contextual information. For each token $w_t \in S$, the network updates its hidden state as follows,

$$\mathbf{h_t} = \mathbf{RNN}(\mathbf{x_t}, \mathbf{h_{t-1}}) \tag{2.2}$$

where $\mathbf{x_t}$ is the embedding of input token w_t and $\mathrm{RNN}(\cdot)$ is a non-linear function. In Elman RNN [62], this non-linear function is represented as follows,

$$\mathbf{h_t} = \tanh(\mathbf{W_i x_t} + \mathbf{b_i} + \mathbf{W_h h_{t-1}} + \mathbf{b_h}) \tag{2.3}$$

where $\mathbf{W_i}, \mathbf{W_h} \in \mathbb{R}^{d \times d}$ are learnable parameters and $\mathbf{b_i}, \mathbf{b_h} \in \mathbb{R}^d$ are the biases.

However, in practice, RNNs are not suitable for modeling long-term dependencies between tokens in a long sequence. Backpropagation through time causes vanishing or exploding gradient problems. Two variants of RNN were proposed to combat these issues that are presented in the following.

LSTM Long Short-Term Memory (LSTM) [63] combats the long-term dependency problem by memorizing long contextual information in *cell states*. The LSTM can add or remove information to the cell state, regulated by structures called *gates*. LSTM uses three gates, namely input, forget, and output gates. These gates regulate information flow through the network. The LSTM network is represented as follows.

$$\begin{aligned}
\mathbf{i_t} &= \sigma(\mathbf{W_i}[\mathbf{x_t}, \mathbf{h_{t-1}}] + \mathbf{b_i}) \quad & \tilde{\mathbf{c_t}} &= \tanh(\mathbf{W_c}[\mathbf{x_t}, \mathbf{h_{t-1}}] + \mathbf{b_c}) \\
\mathbf{f_t} &= \sigma(\mathbf{W_f}[\mathbf{x_t}, \mathbf{h_{t-1}}] + \mathbf{b_f}) \quad & \mathbf{c_t} &= \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \tilde{\mathbf{c_t}} \\
\mathbf{o_t} &= \sigma(\mathbf{W_o}[\mathbf{x_t}, \mathbf{h_{t-1}}] + \mathbf{b_o}) \quad & \mathbf{h_t} &= \mathbf{o_t} \odot \tanh(\mathbf{c_t})
\end{aligned} \tag{2.4}$$

Here σ denotes the sigmoid activation function and $\odot$ denotes Hadamard product. $\mathbf{c_t}$ is the cell state, and $\mathbf{i_t}, \mathbf{f_t}, \mathbf{o_t}$ are input, forget, and output gates, respectively. $\mathbf{W_i}, \mathbf{W_c}, \mathbf{W_f}, \mathbf{W_o} \in \mathbb{R}^{d \times 2d}$ and $\mathbf{b_i}, \mathbf{b_c}, \mathbf{b_f}, \mathbf{b_o} \in \mathbb{R}^d$ are learnable parameters.

GRU The Gated Recurrent Unit [64] simplified the structure of LSTM as it does not have an output gate. Instead, the GRU uses a *reset gate* and an *update gate* as follows.

$$\begin{aligned}
\mathbf{r_t} &= \sigma(\mathbf{W_r}[\mathbf{x_t}, \mathbf{h_{t-1}}] + \mathbf{b_r}) \\
\mathbf{z_t} &= \sigma(\mathbf{W_z}[\mathbf{x_t}, \mathbf{h_{t-1}}] + \mathbf{b_z}) \\
\tilde{\mathbf{h_i}} &= \sigma(\mathbf{W_h}[\mathbf{r_t}, \mathbf{h_{t-1}}] + \mathbf{b_h}) \\
\mathbf{h_t} &= \mathbf{z_t} \odot \tilde{\mathbf{h_i}} + (1 - \mathbf{z_t}) \odot \mathbf{h_{t-1}}
\end{aligned} \tag{2.5}$$

Here σ denotes the sigmoid activation function and $\odot$ denotes Hadamard product. $\mathbf{r_t}$ and $\mathbf{z_t}$ are the reset gate and update gate respectively. $\mathbf{W_r}, \mathbf{W_z}, \mathbf{W_h} \in \mathbb{R}^{d \times 2d}$ and $\mathbf{b_i}, \mathbf{b_c}, \mathbf{b_f}, \mathbf{b_o} \in \mathbb{R}^d$ are learnable parameters.

Both LSTM and GRU encode contextual information from left to right. However, a bidirectional LSTM or a bidirectional GRU encode contexts from left to right and from right to left and have been more effective empirically. Various natural language processing tasks adopted BiLSTM and BiGRU as the default text encoder, often stacking multiple layers of BiLSTM and BiGRU.

This dissertation uses GRUs for text generation in Chapter 6.

Transformers Not every token in a sequence is of equal importance for any predictive or generative task. It is shown that attention mechanism-based encoder-decoder models that learn to assign variable weights to different tokens in a sequence are effective when applied along with an LSTM or GRU [65]. This led to the development of a purely attention-based encoder-decoded model called Transformer [14].

The basic building block of the Transformer architecture is the *scaled dot product attention*, also called *self-attention*. The self-attention models pairwise interactions between all tokens in a sequence. The embeddings $\mathbf{x_t}$ of each token w_t in the sequence is mapped to three linear projections – query $\mathbf{q_t} = \mathbf{W_Q x_t}$, key $\mathbf{k_t} = \mathbf{W_K x_t}$, and value $\mathbf{v_t} = \mathbf{W_V x_t}$. Thus, the input sequence of embeddings $\mathbf{X} = \mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}$ is projected to sequences of

queries $\mathbf{Q} = \mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_N}$, keys $\mathbf{K} = \mathbf{k_1}, \mathbf{k_2}, \ldots, \mathbf{k_N}$, and values $\mathbf{V} = \mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_N}$. The output of the self-attention is obtained as follows.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \tag{2.6}$$

where d_k is the dimension of the keys.

Instead of performing single attention, the Transformer model applies multiple attention heads. For each of the M heads, the model learns a different set of projection parameters $\mathbf{W_Q}^m, \mathbf{W_K}^m, \mathbf{W_V}^m \; \forall m \in \{1, 2, \ldots, M\}$. The outputs of the attention heads are concatenated and projected back to the d-dimensional vector space.

$$\hat{\mathbf{H}} = \mathbf{W_o} \left([\text{Attention}(\mathbf{Q^1}, \mathbf{K^1}, \mathbf{V^1}), \ldots, \text{Attention}(\mathbf{Q^M}, \mathbf{K^M}, \mathbf{V^M})] \right) \tag{2.7}$$

The model then adds this intermediate representation to the input embeddings $\mathbf{X}$ and performs layer normalization [66]. The model then applies a position-wise feed-forward network that has two fully connected layers with a non-linear layer (ReLU) in between. The output of the feed-forward network is added to its input, and another layer normalization is performed to obtain the final representations of the tokens.

$$\begin{aligned} \tilde{\mathbf{H}} &= \text{FFN}(\text{LayerNorm}(\hat{\mathbf{H}} + \mathbf{X})) \\ \mathbf{H} &= \text{LayerNorm}(\tilde{\mathbf{H}} + \hat{\mathbf{H}}) \end{aligned} \tag{2.8}$$

where $\text{FFN}(\mathbf{z}) = \mathbf{W_2}\text{ReLU}(\mathbf{W_1 z} + \mathbf{b_1}) + \mathbf{b_2}$ and $\text{ReLU}(x) = \max(0, x)$.

The architecture described above is called a *Transformer block*. Several such Transformer blocks are stacked to encode an input sequence. Each layer takes the previous layer's output as its input, except the first layer. In the first layer, as described above, the input is the embeddings of the tokens in the input sequence.

The Transformer encoder has been adopted by BERT [67] – a deep bidirectional trans-

former for language modeling. In traditional language modeling, the objective is to predict $p(w_t|w_{t-1}, w_{t-2}, \ldots, w_1)$, i.e., the likelihood for the next token w_t in the sequence given the previous tokens $w_{t-1}, w_{t-2}, \ldots, w_1$. BERT uses a different objective called *masked language modeling*. In masked language modeling, a small percentage of tokens in the input sequence is masked, and the model predicts those missing tokens. Additionally, BERT also performs next sentence prediction where the two sentences S_1 and S_2 are concatenated using a special separator token `[SEP]`. The model predicts whether S_2 is the next sentence of S_1. The first token of every input sequence is always a special token `[CLS]` that is used for various classification tasks. BERT is pre-trained with a large corpus obtained from the Book Corpus [68] and the English Wikipedia.

In this dissertation, BERT and its domain-specific variant BioBERT [69] are used as text encoder in Chapters 3 and 5. The Transformer architecture is adapted to graph neural networks in Chapter 4.

2.6.2 Graph Neural Network

Representation learning for graph-structured data is different from textual data as unlike texts, there is no specific order in which the nodes in a graph can be arranged. Graph representation learning must posses a vital property called *permutation invariance* so that the learned representation does not depend on the arbitrary ordering of nodes in a graph. Graph Neural Network (GNN) is a general framework for graph representation learning using deep neural networks. Prior to GNN, various *shallow* graph representation learning methods have been proposed (e.g., DeepWalk [70], LINE [71], Node2Vec [72], etc.). However, none of these approaches directly exploit the structure of the graph to learn representations. A GNN, on the other hand, learns representations of nodes by exploiting the neighborhood structure.

The defining feature of GNN is that it uses a form of *neural message passing* in which vector messages are exchanged between adjacent nodes and updated using neural networks.

Each node $u \in \mathcal{V}$ in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, receives messages from its immediate neighboring nodes $\mathcal{N}(u) = \{v,\ (u,v) \in \mathcal{E}\}$. These messages are aggregated using an aggregation function and the representation of node u is updated as follows.

$$\mathbf{h}_u^{(k)} = \mathrm{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k-1)}, \mathrm{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k-1)},\ \forall v \in \mathcal{N}(u)\})\right) \tag{2.9}$$

Here $\mathrm{UPDATE}(\cdot)$ and $\mathrm{AGGREGATE}(\cdot)$ are arbitrary differentiable functions. At each iteration k of the GNN, the $\mathrm{AGGREGATE}(\cdot)$ function takes as input the set of embeddings of the nodes in u's local neighborhood $\mathcal{N}(u)$ and generates a message $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ based on this aggregated neighborhood information. The update function $\mathrm{UPDATE}(\cdot)$ then combines the message $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ with the previous embedding $\mathbf{h}_u^{(k-1)}$ of node u to generate the updated embedding $\mathbf{h}_u^{(k)}$. The initial embeddings $\mathbf{h}_u^{(0)}$ are set to the features of node u. After K iterations of message passing, the final representation of a node is obtained.

Over the past few years, several variants of GNNs have been proposed. They differ in the formulation of the aggregate and update functions. One such popular variant is the Graph Convolutional Networks (GCN) [29]. A GCN employs a symmetric-normalized aggregation as well as a self-loop update approach.

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{h}_v^{(k-1)}}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right) \tag{2.10}$$

A GCN gives equal importance to all the neighboring nodes in aggregation. In contrast, a Graph Attention Network (GAT) [73] enables assigning different importance to different nodes in the neighborhood of a node. A GAT learns attention weights using the self attention mechanism as follows. For each pair of nodes (u, v), the model obtains an attention coefficient $e_{(u,v)}$ as shown in the following.

$$e_{u,v} = \sigma(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_u, \mathbf{W}\mathbf{h}_v]) \tag{2.11}$$

In the above equation, $\mathbf{a}$ and $\mathbf{W}$ are learnable parameters, and $\sigma(\cdot)$ is a non-linear activation function such as LeakyReLU. The model then obtains the importance weight of each neighbor by normalizing the attention coefficients as shown in the following.

$$\alpha_{u,v} = \frac{\exp(e_{u,v})}{\sum_{v' \in \mathcal{N}(u) \cup \{u\}} \exp(e_{u,v'})} \tag{2.12}$$

The normalized attention coefficients are used to compute a linear combination of the features corresponding to them, to serve as the final output features for every node as shown in the following.

$$\mathbf{h}_u = \sigma \left(\sum_{v \in \mathcal{N}(u) \cup \{u\}} \alpha_{u,v} \mathbf{W} \mathbf{h}_v \right) \tag{2.13}$$

A GAT also uses multiple attention heads for each node and concatenates the output of each attention head for the final representation of a node.

The GCN has been extented for relational multi-graphs (e.g., knowlegde graphs). Examples of this line of work includes R-GCN [32], CompGCN [35], and SACN [34]. These models also consider the edge type while performing message passing and aggregation.

This dissertation uses a variant of GNN in Chapter 4 that resembles the Transformer model for inductive representation learning in knowledge graphs.

2.6.3 Policy Gradient

Reinforcement learning is a general-purpose framework for sequential decision making. Many natural language processing tasks (e.g., knowledge-based question answering, dialog generation, text generation, etc.) can be formulated as sequential decision making tasks.

Reinforcement learning is often modeled as a Markov Decision Process (MDP). MDP is defined by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions that an agent can choose from, $\mathcal{P}$ is a set of probability distribution functions that define the transition probability $\Pr(s_{t+1} = s' | s_t = s \text{ and } a_t = a)$, i.e., the probability of transitioning from state s' to state s under action a, and $\mathcal{R}$ is a set of reward functions $r(s, a, s')$ that

is the immediate reward after transitioning from s to s' with action a . A policy $\pi(a_t|s_t)$ determines the action a_t that the agent chooses to perform at a given state s_t. The goal is to find an optimal policy $\pi^*(a_t|s_t)$ so that the total reward along the trajectory followed by the optimal policy is maximized.

Several learning algorithms have been proposed for policy learning. They are broadly categorized into model-based and model-free algorithms, and on-policy and off-policy learning algorithm. The details of these algorithms is available in [74]. This section focuses on an off-policy model-free learning algorithm called *policy gradient*.

Policy gradient optimizes parameterized policies with respect to the expected return using gradient descent. The objective function of policy gradient can be defined as the following.

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_t r(s_t, a_t)\right] \tag{2.14}$$

where $\tau = (s_0, a_0, s_1, a_1, ..., s_T)$ is a trajectory sampled from the policy π_θ parameterized by θ.

The gradient of this objective function is obtained as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\left(\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_t r(s_t, a_t)\right)\right] \tag{2.15}$$

In the policy gradient algorithm REINFORCE [13], the gradient is approximated by sampling N trajectories from policy π_θ as the following.

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_t r(s_t, a_t)\right)\right] \tag{2.16}$$

This dissertation uses policy gradient for explainable knowledge graph reasoning in Chapter 4.

CHAPTER 3
KNOWLEDGE EXTRACTION

3.1 Overview

This chapter presents automatic knowledge extraction methods using the supervised learning paradigm and large pre-trained language models. Automatic knowledge extraction requires the identification of entities and their relationships in text documents. In this chapter, we first introduce an efficient and effective model for entity disambiguation (aka. entity linking) where we assume that the mention spans are known following prior works. Then, we extend this model for end-to-end entity linking, where our proposed model jointly performs the tasks of mention span detection and entity disambiguation. Finally, we further extend our model for end-to-end knowledge triple extraction in which the model can predict the head and the tail entities and their relationship(s). Furthermore, the model can map these extracted triples to their canonical form in a target knowledge graph.

Entity Linking Entity linking is the task of identifying mentions of noun phrases in a text document and disambiguating them by mapping them to canonical entities listed in a reference knowledge base. This is an essential step in information extraction, and therefore has been studied extensively both in domain-specific and domain-agnostic settings. Recent models [8, 9] attempt to learn better representations of mentions and candidates using the rich contextual information encoded in pre-trained language models such as BERT [67]. These models follow a *retrieve and rerank* paradigm, which consists of two separate steps: First, the candidate entities are selected using a retrieval model. Subsequently, the retrieved candidates are ranked by a reranker model.

Although this approach has yielded strong results, owing primarily to the powerful contextual representation learning ability of BERT-based encoders, these models typically

process a single mention at a time. Processing one mention at a time incurs a substantial overhead both during training and test time, leading to a system that is slow and impractical.

In this chapter, we propose a *collective entity linking* method that processes an entire document only once, such that all entity mentions within it are linked to their respective target entities in the knowledge base in one pass.

Compared to the recently proposed retrieve and rerank paradigm-based entity linking model BLINK [9], our model is up to 25x faster. BLINK deploys two separately trainable models for the candidate retrieval and reranking tasks. In contrast, our method learns a single model that can perform both the retrieval and reranking steps of entity linking. Our model does not require candidate retrieval at inference time, as our dual encoder approach allows us to compare each mention to all entities in the target knowledge base, thus significantly reducing the overhead at inference time.

Contributions The key contributions of our work are as follows.

- We empirically demonstrate that it is possible to match the efficacy of a retrieve and rerank paradigm-based model by processing multiple mentions in a document in one shot using a collective dual encoder.

- We measure the efficiency of our proposed model in terms of training and inference time by comparing it to a per-mention entity disambiguation model that is used in retrieve and rerank paradigm-based models. We show that training our collective entity disambiguation model is 3x faster than other dual encoder models with the same number of parameters that perform per-mention entity disambiguation. At inference time, our model is 3-25x faster than other comparable models.

- Our model can also perform end-to-end entity linking when trained with the multi-task objective of mention span detection and entity disambiguation. We show that without using any semantic type information, our model significantly out-performs

two recent biomedical entity linking models – MedType [75] and SciSpacy [76] – on two benchmark datasets.

3.2 Related Work

3.2.1 Entity Linking

The task of entity linking has been studied extensively in the literature. In the past, most models relied on hand-crafted features for entity disambiguation using surface forms and alias tables. With the advent of deep learning, contextual representation learning for mention spans has become more popular [77, 78]. Recent advances in Transformer-based modeling of entity linking [9, 79] have achieved state-of-the-art performance on traditional benchmark datasets such as AIDA-CoNLL and TACKBP 2010.

In the biomedical domain, there are many existing tools, such as TaggerOne [80], MetaMap [81], cTAKES [82], QuickUMLS [83], among others, for normalizing mentions of biomedical concepts to a biomedical thesaurus. Most of these methods rely on feature-based approaches. Recently, [84] proposed a model that utilizes the latent semantic information of mentions and entities to perform entity linking. Other recent models such as [85] and [75] also leverage semantic type information for improved entity disambiguation. Our work is different from these approaches, as our model does not use semantic type information, because such information may not always be available. Recent works such as [85] and [86] deploy a BERT-based retrieve and re-rank model. In contrast, our model does not rely on a separate re-ranker model, which significantly improves its efficiency.

3.2.2 Collective Entity Linking

Collective entity linking aims to combine and exploit the local contextual information and global structural dependencies of mentions. To this end, NCEL [87] applies Graph Convolutional Networks to integrate both local contextual features and global coherence information for entity linking. [88] address entity disambiguation as a sequential decision task,

disambiguating mentions one by one, while using words and already disambiguated entities to disambiguate new mentions. [89] proposed a gradient tree boosting based structured learning method that applies bidirectional beam search to consider contextual information from the past and future to perform better collective entity resolution. [90] proposed a recurrent random walk based model that exploits external semantic information for sequentially disambiguating mentions. In contrast to these models, our model does not perform entity disambiguation sequentially and is capable of resolving every entity in the document in one shot. Recently, [79] proposed a BERT-based model for collective entity linking. However, their method requires external sources such as alias tables, phrase tables, and co-occurring entity mentions in Wikipedia pages for candidate retrieval during training. Hence, this method can only be used for wikification and is not applicable for entity linking with regard to other targets, as for example in the biomedical domain. Our dense retrieval approach, in contrast, is domain-agnostic. Moreover, in [79], the mentions along with their contexts are encoded using a BERT-based encoder, but the candidates are encoded using a linear projection to the dense vector space. This asymmetry in representation learning is avoided in our dual encoder model, where both the mentions along with its contexts and the candidate entities are encoded by two BERT-based encoders.

3.2.3 End-to-End Entity Linking

End-to-end entity linking refers to the task of predicting mention spans and the corresponding target entities jointly using a single model. Traditionally, span detection and entity disambiguation tasks were done in a pipelined approach, making these approaches susceptible to error propagation. To alleviate this issue, [7] proposed a neural end-to-end model that performs the dual tasks of mention span detection and entity disambiguation. However, for span detection and disambiguation, their method relies on an empirical probabilistic entity mapping $p(e|m)$ to select a candidate set $C(m)$ for each mention m. Such mention–entity prior $p(e|m)$ is not available in every domain, especially in the biomedical domain that we

consider in this chapter. In contrast, our method does not rely on any extrinsic sources of information.

3.2.4 Relation Extraction

Identifying named entities and relationships among them is the core task in information extraction. Prior works have explored various pipe-lined approaches where entity mentions in a document (or a sentence) are identified by a Named Entity Recognition (NER) model, and then a relation classifier is used to determine relationships between pairs of entity mentions. These models are susceptible to error propagation as they rely heavily on the accuracy of the NER model. To alleviate the problem of error propagation, many approaches adopted a multi-task learning setup where a model jointly learns to detect named entities and determine the relationships among entities. Many of these models rely on extrinsic linguistic information provided by NLP methods such as dependency parsers, POS tagger, etc. that may not be available in many domains. Some more recent approaches, such as [91, 92], proposed end-to-end models for entity recognition and relation extraction. These models fine-tune pre-trained language models (BERT) with various span detection and relation extraction methods. Other Non-BERT methods [93, 94, 95] have also produced state-of-the-art results on various benchmark relation extraction datasets. Although these models yield state-of-the-art results for relation extraction, they cannot produce canonicalized triples that can be directly integrated into a knowledge graph. In this chapter, we propose a model that directly extracts canonicalized triples to aid knowledge graph completion.

3.2.5 End-to-End Entity Linking and Relation Extraction

There exists some prior work on joint relation extraction and entity linking [96, 97, 98]. The goal is to extract named entities and their relations, and simultaneously link them to their canonical forms in a target knowledge graph.

[96] used the BC5CDR dataset to extract relations between diseases and chemicals.

However, in this dataset, there is only one relation *Chemically induced disease*. Therefore, a model only needs to make a binary decision for a pair of chemical and disease. Using a more complex dataset, [98] aim to canonicalize the extracted entities and the relations to their corresponding Wikidata items. [97] proposed a framework for joint entity linking and relation extraction that rely on Open IE techniques and additional side information.

3.3 Entity Linking Model

Given a document $d = [x_1^d, \ldots, x_T^d]$ of T tokens with N mentions $\{m_1, \ldots, m_N\}$ and a set of M entities $\{e_1, \ldots, e_M\}$ in a target knowledge base or thesaurus $\mathcal{E}$, the task of collective entity disambiguation consists of mapping each entity mention m_k in the document to a target entity $t_k \in \mathcal{E}$ in one shot. Each mention in the document d may span over one or multiple tokens, denoted by pairs (i, j) of start and end index positions such that $m_k = [x_i^d, \ldots, x_j^d]$.

3.3.1 Encoding Mentions and Candidates

Our model consists of two BERT-based encoders. The *mention encoder* is responsible for learning representations of contextual mentions and the *candidate encoder* learns representations for the candidate entities. A schematic diagram of the model is presented in Figure 3.1. Following the BERT model, the input sequences to these encoders start and end with the special tokens `[CLS]` and `[SEP]`, respectively.

Mention Encoder Given an input text document $[x_1^d, \ldots, x_T^d]$ of T tokens with M mentions, the output of the final layer of the encoder, denoted by $[\mathbf{h_1}, \ldots, \mathbf{h_T}]$, is a contextualized representation of the input tokens. For each mention span (i, j), we concatenate the first and the last tokens of the span and pass it through a linear layer to obtain the representations for each of the mentions. Formally, the representation of mention m_k is given as

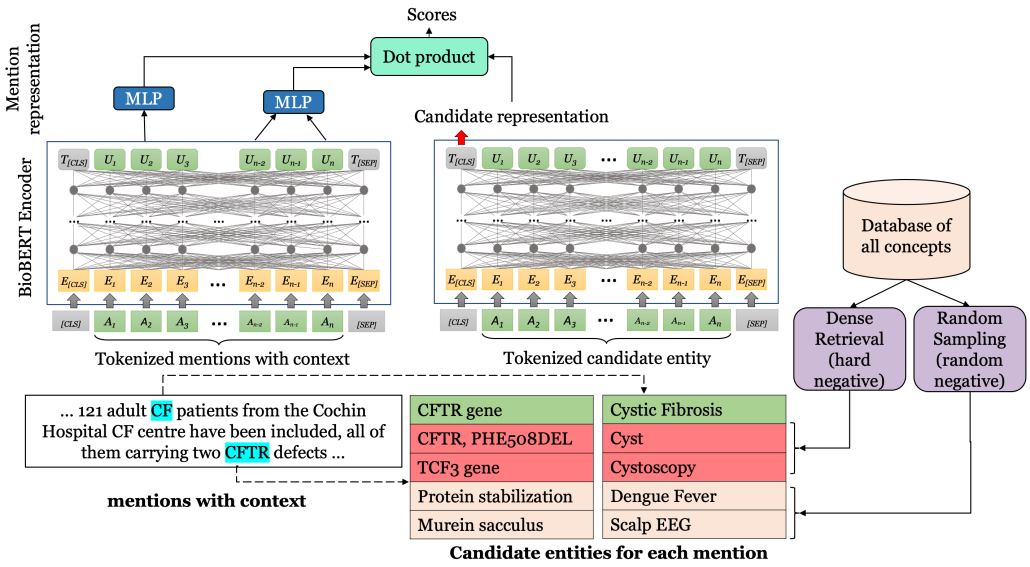


Figure 3.1: A schematic diagram of the Dual Encoder model for collective entity disambiguation. In this diagram, the number of mentions in a document and the number of candidate entities per mention are for illustration purpose only. The inputs to the BioBERT encoders are the tokens obtained from the BioBERT tokenizer.

$$\mathbf{u_k^m} = \mathbf{W}[\mathbf{h_i}; \mathbf{h_j}] + \mathbf{b} \tag{3.1}$$

Since the encoder module deploys a self-attention mechanism, every mention inherently captures contextual information from the other mentions in the document.

Candidate Encoder Given an input candidate entity $e = [y_1^e, \ldots, y_T^e]$ of T tokens, the output of the final layer corresponding to the `[CLS]` token yields the representation for the candidate entity. We denote the representation of entity e as $\mathbf{v_e}$. As shown in Figure 3.1, we use the UMLS concept name of each candidate entity as the input to the candidate encoder.

3.3.2 Candidate Selection

Candidate Retrieval Since the entity disambiguation task is formulated as a learning to rank problem, we need to retrieve negative candidate entities for ranking during training. To this end, we randomly sample a set of negative candidates from the pool of all entities

in the knowledge base. Additionally, we adopt the *hard negative mining* strategy used by [99] to retrieve negative candidates by performing nearest neighbor search using the dense representations of mentions and candidates described above. The hard negative candidates are the entities that are more similar to the mention than the gold target entity.

Candidate Scoring The retrieved set of candidate entities $\mathcal{C}_k = \{c_1^k, \ldots, c_l^k\}$ for each mention m_k are scored using a dot product between the mention representation $\mathbf{u_k^m}$ and each candidate representation $\mathbf{v_c}$. Formally, for each $c \in \mathcal{C}_k$

$$\psi(m_k, c) = (\mathbf{u_k^m})^\intercal \mathbf{v_c} \tag{3.2}$$

3.3.3 Training and Inference

Loss Function and Training We train our model using the cross-entropy loss function to maximize the score of the gold target entities.

Inference During inference, we do not require candidate retrieval per mention. The representations of all entities in the knowledge base $\mathcal{E}$ can be pre-computed and cached. The inference task is thus reduced to finding the maximum dot product between each mention representation and all entity representations.

$$\hat{t}_k = \arg\max_{e \in \mathcal{E}} \{(\mathbf{u_k^m})^\intercal \mathbf{v_e}\} \tag{3.3}$$

3.4 End-to-End Entity Linking Model

Many of the state-of-the-art entity disambiguation models assume that gold mention spans are available during test time and thus have limited applicability in real-world entity linking tasks, where such gold mentions are typically not available. To avoid this, recent works [7, 79, 100] have investigated end-to-end entity linking, where a model needs to perform both

mention span detection and entity disambiguation.

3.4.1 Mention Span Detection

We experiment with two different methods for mention span detection with different computational complexity. In our first method, following [79], we use a simple `BIO` tagging scheme to identify the mention spans. Every token in the input text is annotated with one of these three tags. Under this tagging scheme, any contiguous segment of tokens starting with a `B` tag and followed by `I` tags is treated as a mention. Although this method is computationally efficient ($\mathcal{O}(T)$), our empirical results suggest that it is not as effective as the following.

Following the recent work of [7] and [100], our mention span detection method enumerates all possible spans in the input text document as potential mentions. However, enumerating all possible spans in a document of length T is prohibitively large ($\mathcal{O}(T^2)$) and computationally expensive. Therefore, we constrain the maximum length of a mention span to $L \ll T$.

We calculate the probability of each candidate mention span (i, j) as follows.

$$p(m|(i,j)) = \sigma(\mathbf{w_s^\intercal h_i} + \mathbf{w_e^\intercal h_j} + \sum_{q=i}^{j} \mathbf{w_m^\intercal h_q}) \tag{3.4}$$

where $\mathbf{w_s}$, $\mathbf{w_e}$, and $\mathbf{w_m}$ are trainable parameters and $\sigma(x) = \frac{1}{1+e^{-x}}$.

3.4.2 Entity Disambiguation

We represent each mention (i, j) by mean pooling the final layer of the encoder, i.e., $\mathbf{u^m_{(i,j)}} = \frac{1}{j-i+1} \sum_{q=i}^{j} \mathbf{h_q}$. During training, we perform candidate selection as described in Subsection 3.3.2.

We jointly train the model by minimizing the sum of mention detection loss and entity disambiguation loss. We use a binary cross-entropy loss for mention detection with the

gold mention spans as positive and other candidate mention spans as negative samples. For entity disambiguation, we use the cross-entropy loss to minimize the negative log likelihood of the gold target entity given a gold mention span.

During inference, we choose only the candidate mentions with $p(m|(i,j)) > \gamma$ as the predicted mention spans. Then, as mentioned in Subsection 3.3.3, we determine the maximum dot product between the mention representations and all candidate entity representations to predict the entity for each predicted mention during inference.

3.5 Evaluation

3.5.1 Datasets

We evaluate our method on two challenging datasets from the biomedical domain. In recent times, there is an increased focus on information extraction from biomedical text such as biomedical academic publications, electronic health records, discharge summaries of patients, and clinical reports. Extracting named concepts from biomedical text requires domain expertise. Existing automatic extraction methods, including the methods and tools catering to the biomedical domain [82, 83, 81], often perform poorly due to the inherent challenges of biomedical text: (1) Biomedical text typically contains substantial domain-specific abbreviations and acronyms. For example, *CT* could stand for *Computed tomography* or *Copper Toxicosis*. (2) The target concepts in the knowledge base often have very similar surface forms, making the disambiguation task difficult. For example, *Pseudomonas aeruginosa* is a kind of bacteria, while *Pseudomonas aeruginosa infection* is a disease. (3) The colloquial form of many concept names differ significantly from standard medical terminology. For example, *reductions in mean arterial blood pressure* corresponds to *Hypotension* in medical terminology. Many existing biomedical information extraction tools rely on similarities in surface forms of mentions and candidates, and thus invariably falter in more challenging cases such as these. Besides, long mention spans (e.g., disease names) and the density of mentions per document make the biomedical datasets substan-

tially more challenging.

Our experiments are conducted on two challenging datasets from the biomedical domain – MedMentions [101] and the BioCreative V Chemical Disease Relation (BC5CDR) dataset [102]. In the following, we provide some details of these two datasets, while basic statistics are given in Table 3.1.

Datasets	Mentions	Mentions/Doc	Unique Concepts	Types
MedMentions	352,496	80	34,724	128
BC5CDR	28,559	19	9,149	2

Table 3.1: Details of the datasets used for evaluation.

MedMentions is a large-scale biomedical corpus annotated with UMLS concepts. It consists of a total of $4{,}392$ abstracts published on PubMed®. The dataset has $352{,}496$ mentions, and each mention is associated with a single UMLS Concept Unique Identifier (CUI) and one or more semantic types identified by a Type Unique Identifier (TUI). The concepts belong to 128 different semantic types. MedMentions also provides a 60% – 20% – 20% random partitioning of the corpus into training, development, and test sets. Note that 12% of the concepts in the test dataset do not occur in the training or development sets.

The BC5CDR corpus consists of $1{,}500$ PubMed® articles with $4{,}409$ annotated chemicals and $5{,}818$ diseases, which are equally partitioned into training, development, and test sets. Each entity annotation includes both the mention text spans and normalized concept identifiers, using MeSH as the target vocabulary. Apart from entity linking annotations, this dataset also provides $3{,}116$ chemical–disease relations. However, identifying relations between mentions is beyond the scope of our study on entity linking and hence, we ignore these annotations.

3.5.2 Baselines

We compare our model against some of the recent entity linking models from both the biomedical and non-biomedical domains. In the biomedical domain, LATTE [84] showed

state-of-the-art results on the MedMentions dataset. However, we find that LATTE adds the gold target entity to the set of candidates retrieved by the BM25 retrieval method during both training and inference.

The Cross Encoder model proposed by [8], which follows a *retrieve and rerank* paradigm, has been successfully adopted in the biomedical domain by [85] and [86]. We use our own implementation of the model by [8] for comparison.

We also compare with BLINK [9], a linking model that uses dense retrieval using dual encoders for candidate generation, followed by a cross-encoder for reranking.

Additionally, we use the dual encoder model that processes each mention independently as a baseline. In principle, this baseline is similar to the retriever model of [9] and [99].

For the task of end-to-end entity disambiguation, we compare our models with two recent state-of-the-art models – SciSpacy [76] and MedType [75]. SciSpacy uses overlapping character N-grams for mention span detection and entity disambiguation. MedType improves the results of SciSpacy by using a better candidate retrieval method that exploits the semantic type information of the candidate entities.

3.5.3 Experimental Details

In this section, we provide details pertaining to the experiments for the purpose of reproducibility.

Domain-Adaptive Pretraining Recent studies [8, 79, 9] have shown that pre-training BERT on the target domain provides additional performance gains for entity linking. Following this finding, we adopt BioBERT [69] as our domain-specific pretrained model. BioBERT is intitialzed with the parameters of the original BERT model, and further pretrained on PubMed abstracts to adapt to biomedical NLP tasks.

Data Wrangling In theory, our collective entity disambiguation model is capable of processing documents of arbitrary length. However, there are practical constraints. First, the

GPU memory limit enforces an upper bound on the number of mentions that can be processed together, and secondly, BERT stipulates the maximum length of the input sequence to be 512 tokens. To circumvent these constraints, we segment each document so that each chunk contains a maximum of 8 mentions or a maximum of 512 tokens (whichever happens earlier). After this data wrangling process, the $4{,}392$ original documents in the MedMentions dataset are split into $44{,}983$ segmented documents. We postulate that with more GPU memory and longer context [103], our collective entity disambiguation model will be able to process documents of arbitrary length without segmentation.

For the other baselines, we process each mention along with its contexts independently. We found that a context window of 128 characters surrounding each mention suffices for these models. We also experimented with longer contexts and observed that the performance of the models deteriorates.

Hyperparameters To encode mentions, we use a context window of up to 128 tokens for the single-mention Dual Encoder. The candidate entities are tokenized to a maximal length of 128 tokens across all Dual Encoder models. In the Cross Encoder and BLINK models, where candidate tokens are appended to the context tokens, we use a maximum of 256 tokens. For Collective Dual Encoder models, the mention encoder can encode a tokenized document of maximum length 512. For all our experiments, we use AdamW stochastic optimization. We also use linear scheduling for the learning rate of the optimizer. For the single-mention Dual Encoder, Cross Encoder and BLINK model, we find an initial learning rate of 0.00005 to be optimal. For collective Dual Encoder models, we find an initial learning rate of 0.00001 to be suitable for both the end-to-end and non-end-to-end settings. The ratio of hard and random negative candidates is set to 1:1, as we choose 10 samples from each.

3.5.4 Evaluation Metrics

Picking the correct target entity among a set of candidate entities is a learning to rank problem. Therefore, we use Precision@1 and Mean Average Precision (MAP) as our evaluation metrics when the gold mention spans are known. Since there is only one correct target entity per mention in our datasets, Precision@1 is also equivalent to the accuracy. One can consider these metrics in normalized and unnormalized settings. The normalized setting is applicable when candidate retrieval is done during inference and the target entity is present in the set of retrieved candidates. Since our model and other Dual Encoder based models do not require retrieval at test time, the normalized evaluation setting is not applicable in these cases.

3.5.5 Results

	Candidate retrieval method		Unnormalized		Normalized	
Model	Training	Test	P@1	MAP	P@1	MAP
† LATTE	BM25	BM25	-	-	88.5	92.8
† Cross Encoder	BM25	BM25	-	-	91.6	95.1
Cross Encoder	BM25	BM25	53.8	56.2	90.4	94.4
Dual Encoder (1 mention)	DR (random)	all entities	54.1	64.8	N/A	N/A
Dual Encoder (1 mention)	DR (random + hard)	all entities	62.9	69.7	N/A	N/A
BLINK	DR (random + hard)	DR (hard)	68.1	73.0	84.7	90.8
Dual Encoder (collective)	DR (random)	all entities	58.2	68.5	N/A	N/A
Dual Encoder (collective)	DR (random + hard)	all entities	**68.4**	**75.6**	N/A	N/A

Table 3.2: Precision@1 and Mean Average Precision (MAP) for the entity disambiguation task on the MedMentions dataset when the gold mention spans are known. † LATTE results are copied from the original paper and always incorporate gold entities as candidates (thus recall is always 100%). † Cross Encoder shows results in this setting as a reference point. Models without † do not add gold entities to the candidate set. 'N/A' stands for 'Not Applicable'. 'DR' stands for dense retrieval.

Entity Disambiguation We provide the results of our experiments for the entity disambiguation task on the MedMentions and BC5CDR datasets in Table 3.2 and Table 3.3, respectively. For the MedMentions dataset, our collective dual encoder model outperforms

	Candidate retrieval method		Unnormalized		Normalized	
Model	Training	Test	P@1	MAP	P@1	MAP
Cross Encoder	BM25	BM25	72.1	73.1	96.8	98.1
Dual Encoder (1 mention)	DR (random)	all entities	76.3	82.4	N/A	N/A
Dual Encoder (1 mention)	DR (random + hard)	all entities	**84.8**	**87.7**	N/A	N/A
BLINK	DR (random + hard)	DR (hard)	74.7	75.6	97.2	98.4
Dual Encoder (collective)	DR (random)	all entities	69.0	77.2	N/A	N/A
Dual Encoder (collective)	DR (random + hard)	all entities	80.7	85.1	N/A	N/A

Table 3.3: Precision@1 and Mean Average Precision (MAP) for the entity disambiguation task on the BC5CDR dataset when the gold mention spans are known. 'N/A' stands for 'Not Applicable'. 'DR' stands for dense retrieval.

all other models, while being extremely time efficient during training and inference. On the BC5CDR dataset, our method performs adequately as compared to other baselines. Our model compares favorably against the state-of-the-art entity linking model BLINK on both datasets. Surprisingly, for the BC5CDR dataset, BLINK is outperformed by the Dual Encoder baselines that process each mention independently, despite the fact that BLINK's input candidates are generated by this model. We conjecture that BLINK's cross encoder model for re-ranking is more susceptible to overfitting on this relatively small-scale dataset. Our model consistently outperforms the Cross Encoder model, which reinforces the prior observations made by [9] that dense retrieval of candidates improves the accuracy of entity disambiguation models. Finally, comparisons with an ablated version of our model that uses only random negative candidates during training show that hard negative mining is essential for the model for better entity disambiguation.

Training and Inference Speed We perform a comparative analysis of the training speed of our collective dual encoder model with the single-mention dual encoder model. We show in Figure 3.2 and Figure 3.3 that our model achieves higher accuracy and recall@10 much faster than the single-mention dual encoder model. To be precise, our model is 3x faster than the single-mention Dual Encoder model.

We also compare the inference speed of our model with BLINK and the single-mention Dual Encoder model. The comparisons of inference speed for the two datasets are presented

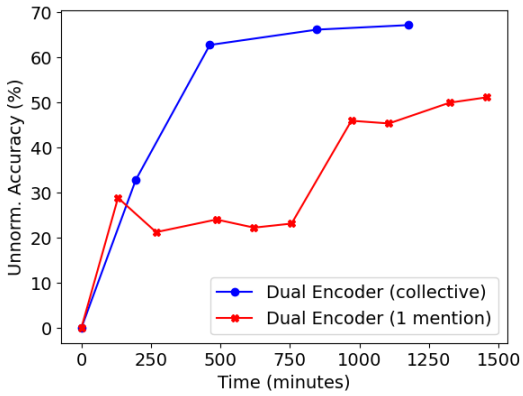


Figure 3.2: Comparative analysis of training speed measured in terms of accuracy achieved in first 24 hours of training. Both models were trained on 4 NVIDIA Quadro RTX GPUs with 24 GB memory.

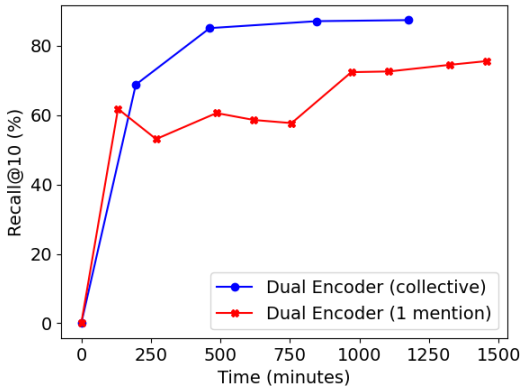


Figure 3.3: Comparative analysis of training speed measured in terms of recall@10 achieved in first 24 hours of training. Both models were trained on 4 NVIDIA Quadro RTX GPUs with 24 GB memory.

in Table 3.4 and Table 3.5, respectively. The inference speed is measured on a single NVIDIA Quadro RTX GPU with batch size 1. We observe that our collective dual encoder model is 3-4x faster than the single-mention Dual Encoder model and up to 25x faster (an average over two datasets) than BLINK. Since our model can process a document with N mentions in one shot, we achieve higher entity disambiguation speed than the single-mention Dual Encoder and the BLINK model – both require N forward passes to process the N mentions in a document. Caching the entity representations also helps our model and the single-mention Dual Encoder model at test time. The cross encoder of BLINK prevents

it from using any cached entity representations, which drastically slows down the entity resolution speed of BLINK.

Model	mentions/sec
BLINK (reranker)	11.5
Dual Encoder (1 mention)	65.0
Dual Encoder (collective)	**192.4**

Table 3.4: Inference speed comparison for the MedMentions dataset.

Model	mentions/sec
BLINK (reranker)	11.5
Dual Encoder (1 mention)	87.0
Dual Encoder (collective)	**402.5**

Table 3.5: Inference speed comparison for the BC5CDR dataset.

Candidate Recall We compare the recall@10 metrics of BM25 retrieval method used in LATTE and Cross Encoder to the dense retrieval method used in BLINK and in our model. We present our results in Table 3.6 for the MedMentions and BC5CDR datasets, respectively. Similar to the observations made for BLINK and [99], we also find that dense retrieval has a superior recall than BM25. However, we observe that the recall value of dense retrieval depends on the underlying entity disambiguation model. For instance, on the MedMentions dataset, our model has much higher recall@10 than the Dual Encoder model that processes each mention independently, while both models are trained using a combination of hard and random negative candidates.

	MedMentions		BC5CDR	
Model	Dev	Test	Dev	Test
BM25	59.8	59.5	76.3	74.5
Dense retrieval (1 mention)	80.2	80.6	92.1	**92.3**
Dense retrieval (collective)	**87.5**	**87.6**	**92.2**	**92.3**

Table 3.6: Comparison of Dev and Test set Recall@10 for MedMentions and BC5CDR datasets

Model	MedMentions						BC5CDR					
	Partial match			Strict match			Partial match			Strict match		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SciSpacy	40.9	40.2	40.6	37.7	36.6	37.1	15.5	53.4	24.0	14.5	48.4	22.3
MedType	44.7	44.1	44.4	41.2	40.0	40.6	16.6	57.0	25.7	15.3	51.0	23.5
Dual Encoder (BIO tags)	44.5	37.6	40.7	41.2	34.9	37.8	29.2	31.5	30.3	10.2	10.8	10.5
Dual Encoder (Exhaustive)	**56.3**	**56.4**	**56.4**	**52.9**	**53.8**	**53.4**	**76.0**	**74.4**	**75.2**	**74.6**	**73.1**	**73.8**

Table 3.7: Micro Precision (P), Recall (R) and F1 scores for the end-to-end entity linking task on the MedMentions and BC5DCR datasets.

End-to-End Entity Disambiguation For the end-to-end entity linking task, we evaluate the models with two different evaluation protocols. In the *strict match* protocol, the predicted mention spans and predicted target entity must match strictly with the gold spans and target entity. In the *partial match* protocol, if there is an overlap between the predicted mention span and the gold mention span, and the predicted target entity matches the gold target entity, then it is considered to be a true positive. We evaluate our models using micro-averaged precision, recall, and F1 scores as evaluation metrics. For a fair comparison, we use the off-the-shelf evaluation tool `neleval`[1], which is also used for MedType. We follow the same evaluation protocol and settings as used for MedType.

We present the results of our collective Dual Encoder model and the baselines in Table 3.7. The results show that exhaustive search over all possible spans for mention detection yields significantly better results than the `BIO` tagging based method, despite the additional computational cost. Moreover, our dual encoder based end-to-end entity linking model significantly outperforms SciSpacy and MedType. Note that there are highly specialized models such as TaggerOne [80] that perform much better than our model on the BC5CDR dataset. However, TaggerOne is suitable for a few specific types of entities such as *Disease* and *Chemical*. For a dataset with entities of various different semantic types (e.g., MedMentions), [101] show that TaggerOne performs inadequately. For such datasets where the target entities belong to many different semantic types, our proposed model is more effective as compared to highly specialized models like TaggerOne.

[1] https://github.com/wikilinks/neleval

3.6 Triple Extraction Model

The task of triple extraction directly extracts canonicalized fact triples from an input text document. Given the sentence *The Black Crusade is a 2004 horror novel by Richard Harland*, our model learns to extract fact triples `(Q7718294, P50, Q1771570)` where `Q7718294` and `Q1771570` are the Wikidata IDs for entities *The Black Crusade* and *Richard Harland* respectively, and `P50` is the Wikidata ID for property *author* which is the relationship between the two entities.

We frame the triple extraction task as a multi-task learning problem. Our model performs *mention span detection* to identify potential mentions of entities, *entity disambiguation* to link mention span to a target knowledge graph, and *relation classification* to identify the relationship between a pair of entity mentions using a single end-to-end differentiable model.

For mention span detection and entity disambiguation we follow the proposed model in Section 3.4. It is shown in [104] that both entity mentions and contextual information contributes in neural relation extraction. Therefore, we use the mean pooling of mention spans to represent the head and tail entities and the last hidden state of the BERT encoder's `[CLS]` token as the sentence representation for relation classification. The representations are concatenated and mapped to a d-dimensional vector $\mathbf{u}_{\mathrm{C+M}}$ as follows.

$$\mathbf{u}_{\mathrm{C+M}} = \mathbf{W}[\mathbf{u}_h; \mathbf{u}_t; \mathbf{v}_d] \tag{3.5}$$

Here $\mathbf{u}_h \in \mathbf{R}^d$ is the head entity's mention embedding, $\mathbf{u}_t \in \mathbf{R}^d$ is the tail entity's mention embedding, and $\mathbf{v}_d$ is the context sentence embedding.

For each mention pair, we score the candidate relations using a dot product between $\mathbf{u}_{\mathrm{C+M}}$ and relation vectors $\mathbf{v}_r \quad \forall r \in \mathcal{R}$ where $\mathcal{R}$ is the set of all relations. Note that we use the same BERT encoder that is used to learn entity representations to learn relation

representation. Formally,

$$\psi(h, r, t) = \mathbf{u}_{\mathrm{C+M}}{}^{\top}\mathbf{v}_r \tag{3.6}$$

During inference, we extract a triple (h, r, t) from the input document d if the likelihood $p(h, r, t|d)$ is greater than a threshold γ. We obtain the likelihood using the chain rule as shown in the following.

$$p(h, r, t|d) = p(r|h, t, d)p(h|d)p(t|d) \tag{3.7}$$

The value of γ is determined using the validation set.

3.7 Evaluation of Triple Extraction

3.7.1 Dataset

TACRED [105] is a popular benchmark dataset for the relation extraction task. However, in TACRED, entity mentions are not aligned to their canonical forms. Therefore, this dataset is not suitable for the triple extraction task. Recently, Trisedya et al. [98] proposed a dataset of Wikipedia sentences mapped to Wikidata triples. The collected dataset contains $255{,}654$ sentence-triple pairs. For each pair, the maximum number of triples is four (i.e., a sentence can produce at most four triples). The dataset contains $330{,}005$ mapped triples from Wikidata, including $279{,}888$ entities and 158 Wikidata properties that serve as relations in the triples.

3.7.2 Experiments

We compare our model to N-gram Attention model proposed by [98]. Addtionally, we also use the results reported in [98] for unsupervised relation extraction [53] followed by end-to-end entity linking [7] and dictionary-based paraphrase detection for relation canonical-

Model	Micro-P	Micro-R	Micro-F1
† MinIE (+ E2E EL [7])	36.7	48.6	41.8
† N-gram Attention [98]	84.7	67.6	75.2
⋆ Our Model ($\gamma > 0.4$)	31.06	42.83	<u>36.01</u>

Table 3.8: Micro-averaged precision, recall, and F-1 scores of the proposed model and other baselines for the WIKI dataset. ⋆ We report the results of our model after 10 epochs of training. † results are taken from Trisedya et al. [98].

ization. We use Micro-averaged precision, recall, and F-1 scores as the evaluation metrics. We show the results in Table 3.8.

3.8 Discussion

This chapter introduces a collective entity linking approach using BERT-based dual encoders to disambiguate multiple mentions of entities in a document in a single shot. We show empirically that our method achieves higher precision than other competitive baseline models in significantly less training and inference time. We also showed that our end-to-end entity linking approach is substantially better than two recently proposed biomedical entity linking models for the end-to-end entity disambiguation task.

Although supervised methods are widely adopted for relation extraction for academic publications, their practical use is limited due unavailability of training data across various domains. Recently, a large-scale dataset called BioREL [106] is proposed for biomedical relation extraction. For open-domain relation extraction, T-REx [107] is emerging as a benchmark dataset that contains millions of annotated sentences.

The WIKI dataset and the other relation extraction datasets have sentence-level annotation. A recently proposed dataset DocRED [108] extended the relation extraction task to the document level. Document-level relation extraction often requires the extraction of relations beyond the sentence boundaries. This involves detection and aggregation of multiple entity mentions and their co-references present in a document. However, like TACREC, DocRED also does not have canonicalized entity annotations.

CHAPTER 4
INDUCTIVE REPRESENTATION LEARNING AND EXPLAINABLE REASONING FOR KNOWLEDGE GRAPHS

4.1 Overview

Recent years have seen a surge in the usage of large-scale cross-domain knowledge graphs [**KnowledgeGraphs2020**] for various tasks, including factoid question answering, fact-based dialogue engines, and information retrieval [109]. Knowledge graphs serve as a source of background factual knowledge for a wide range of applications [2]. For example, Google's knowledge graph is tightly integrated into its search engine, while Apple adopted Wikidata as a source of background knowledge for its virtual assistant Siri. Many such applications deal with queries that can be transformed to a structured relational query of the form $(e_{\mathrm{s}}, r_{\mathrm{q}}, ?)$, where e_{s} is the source entity and r_{q} is the query relation. For example, the query "*Who is the director of World Health Organization?*" can be mapped to the structured query *(World Health Organization, director, ?)* while executing it on a knowledge graph. Unfortunately, due to the inherent sparsity and incompleteness of knowledge graphs, answers to many such queries cannot be fetched directly from the existing facts but need to be inferred indirectly.

Furthermore, with the ever-increasing volume of knowledge graphs, the number of emerging entities also increases. Many of these emerging entities have a small number of known facts when they are integrated into the knowledge graphs. Therefore, their connectivity to pre-existing entities in the knowledge graph is often too sparse.

In recent years, embedding-based models [10] have widely been adopted to infer missing relationships in a knowledge graph. In such embedding-based models, distributed vector representations of entities and relations in the knowledge graph are used to learn a

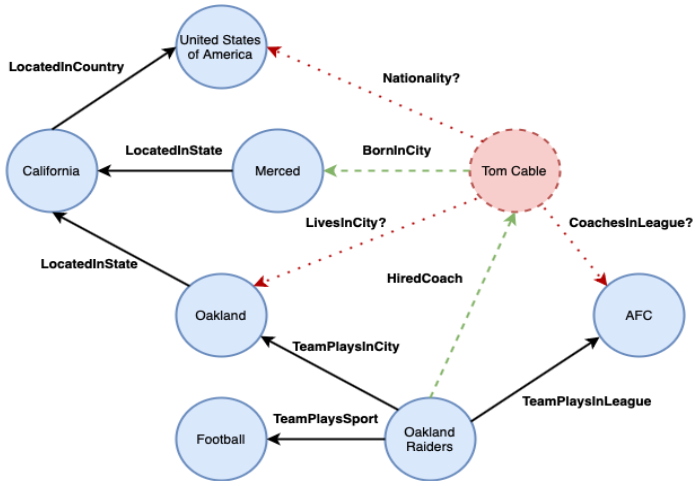


Figure 4.1: A subgraph of NELL with *Tom Cable* as an emerging entity. The solid-lined circles and arrows represent the existing entities and relations. The dashed-lined circles and arrows denote an emerging entity and some known relationships to other existing entities. The unknown relationships that need to be inferred through inductive representation learning and explainable reasoning are shown as dotted arrows.

scoring function $f(e_{\mathrm{s}}, r_{\mathrm{q}}, e_{\mathrm{o}})$ in a latent embedding space to determine the plausibility of inferring a new fact. However, these models lack in terms of the interpretability and explainability of the decisions they make. One does not obtain any clear explanation of why a specific inference is warranted. For example, from the embeddings of facts *(A, born_in, California)* and *(California located_in, US)*, the fact *(A, born_in, US)* could be deduced. But logical composition steps like this one are learned implicitly by knowledge graph embeddings. This means that this approach cannot offer such logical inference paths as support evidence for an answer.

In contrast, path-based reasoning approaches operate in the symbolic space of entities and relations, leveraging the symbolic compositionality of the knowledge graph relations, thus making the inference process explainable. This means that the user can inspect the inference path, consisting of existing edges in the knowledge graph, as supportive evidence. To this end, purely symbolic and fast rule-mining systems, e.g., PRA [37], AMIE+ [45], and AnyBURL [46] may attain a level of performance that is comparable to embedding-based methods but neglect many of the statistical signals exploited by the latter. To leverage

the advantages of both path-based and embedding-based models, some neural-symbolic approaches [38, 39, 40, 41, 47] have as well been proposed. Some recent path-based reasoning approaches [43, 12] formulate the path-finding problem as a Partially Observable Markov Decision Process (POMDP), in which the model learns a policy to find an inference path from the source entity to the target entity using REINFORCE [13], a policy gradient-based reinforcement learning algorithm.

However, most of these approaches are studied with static snapshots of the knowledge graphs, thus severely restricting their applicability for a dynamically evolving knowledge graph with many emerging entities. Except for the purely symbolic rule-mining systems mentioned above, most existing approaches that depend on learning latent representations of entities require that all entities are present during training. Therefore, these models are incapable of learning representations of arbitrary newly emerging entities not seen during training. Some recent approaches such as HyTE [110], and DyRep [111] have considered dynamically evolving temporal knowledge graphs. However, similar to embedding-based models, these models are not explainable.

To overcome this issue, we propose a joint framework for representation learning and reasoning in knowledge graphs that aims at achieving inductive node representation learning capabilities applicable to a dynamic knowledge graph with many emerging entities while preserving the unique advantage of the path-based approaches in terms of explainability. For inductive node representation learning, we propose a variant of *Graph Transformer* encoder [112, 113] that aggregates neighborhood information based on its relevance to the query relation. Furthermore, we use policy gradient-based reinforcement learning (REINFORCE) to decode a reasoning path to the answer entity. We hypothesize that the inductively learned embeddings provide prior semantic knowledge about the underlying knowledge environment to the reinforcement learning agent.

This chapter is based on a published work [114]. We summarize the contributions of this chapter in the following.

- We introduce a joint framework for inductive representation learning and explainable reasoning capable of learning representations for unseen emerging entities during inference by leveraging only a small number of known connections to the other pre-existing entities in the knowledge graph. Our approach can not only infer new connections between an emerging entity and any other pre-existing entity in the knowledge graph but also provides an explainable reasoning path as support evidence for the inference.

- We introduce new train/development/test set splits of existing knowledge graph completion benchmark datasets appropriate for inductive representation learning and reasoning.

4.2 Related Work

4.2.1 Embedding-based Methods

Due to advances in representation learning, embedding-based methods have become the most popular approach for knowledge base completion. Such methods learn d-dimensional distributed vector representations of entities and relations in a knowledge graph. To this end, the translation embedding model TransE [20] learns the embedding of a relation as a simple translation vector from the source entity to the target entity such that $\mathbf{e}_s + \mathbf{e}_r \approx \mathbf{e}_o$. Its variants, e.g., TransH [22], TransR [21], TransD [115] consider similar objectives. Trilinear models such as DistMult [23], along with its counterpart ComplEx [24] in the complex embedding space, use a multiplicative scoring function $f(s, r, o) = \mathbf{e}_s^\top \mathbf{W}_r \mathbf{e}_o$, where $\mathbf{W}_r$ is a diagonal matrix representing the embedding of relation r. Convolutional neural network models such as ConvE [27] and ConvKB [28] apply convolutional kernels over entity and relation embeddings to capture the interactions among them across different dimensions. These models obtain state-of-the-art results on the benchmark link prediction datasets. However, none of the above-mentioned approaches deliver the full reasoning

paths that license specific multi-hop inferences, and hence they either do not support multi-hop inference or do not support it in an interpretable manner. Moreover, these approaches assume a static snapshot of the knowledge graph to train the models and are not straightforwardly extensible to inductive representation learning with previously unseen entities.

4.2.2 Path-based Methods

An alternative stream of research has explored means of identifying specific paths of inference, which is the task we consider in this paper. To this end, the Path Ranking Algorithm (PRA) [37] uses random walks with restarts for multi-hop reasoning. Following PRA, other approaches [38, 39, 40, 41] also leverage random walk based inference. However, the reasoning paths that these methods follow are gathered by random walks independently of the query relation.

Recent approaches have instead adopted policy gradient based reinforcement learning for a more focused exploration of reasoning paths. Policy gradient based models such as DeepPath [42], MINERVA [43], MultiHop [12], and M-Walk [44] formulate the KG reasoning task as a Partially Observable Markov Decision Process and learn a policy conditioned on the query relation. Although the inference paths are explainable in these models, there may be a substantial performance gap in comparison with embedding-based models.

4.2.3 Graph Convolution-based Methods

Graph Convolution Networks (GCNs) can be used for node classification in a homogeneous graph [29]. They are an instance of Message Passing Neural Networks (MPNN), in which the node representations are learned by aggregating information from the nodes' local neighborhood. GraphSAGE [30] attempts to reduce the memory footprint of GCN by random sampling of the neighborhood. Graph Attention Networks (GAT) [73] are a variant of GCN that learn node representations as weighted averages of the neighborhood information. However, GCN and its variants such as GAT and GraphSAGE are not di-

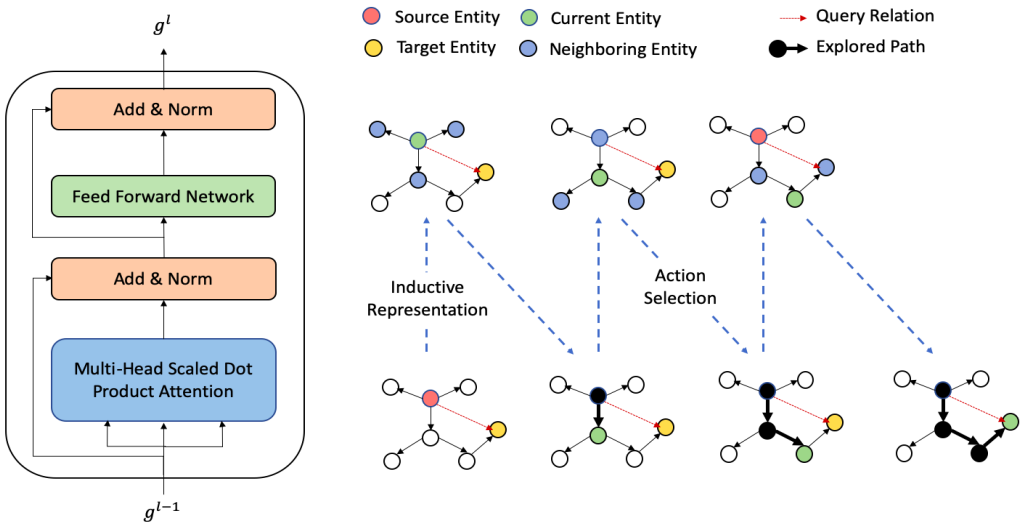


Figure 4.2: A schematic diagram of a Graph Transformer block, along with an illustration of the workflow of our model, demonstrating successive applications of inductive node representation learning and action selection to find a reasoning path.

rectly applicable for link prediction in knowledge graphs, as they ignore the edge (relation) information for obtaining the node embeddings. To alleviate this issue, R-GCNs operate on relational multi-graphs [32], but, similar to GCNs, R-GCNs also need all nodes of the graphs to be present in memory and therefore are not scalable to large-scale knowledge graphs. Hamaguchi et al. [33] proposed a model for computing representations for out-of-KG entities using graph neural networks. The recent models such as SACN [34] and CompGCN [35] leverage the graph structure by inductively learning representations for edges (relations) and nodes (entities). However, unlike our model, these methods are not explainable.

4.3 Model

Our model consists of two modules that are subject to joint end-to-end training. The encoder module learns inductive entity embeddings while accounting for the query relation and the local neighborhood of an entity (Subsection 4.3.2). The decoder module operates on this learned embedding space of entities and relations. By leveraging the embeddings of

the source entity and the query relation, the decoder module infers a reasoning path to the target entity using policy gradient-based reinforcement learning (Subsection 4.3.3). Before describing these components in more detail, Subsection 4.3.1 first provides preliminary definitions.

4.3.1 Problem Statement

Formally, we consider knowledge graphs $\mathcal{G}(\mathcal{E}, \mathcal{R}, \mathcal{F})$ defined as directed multi-graphs such that each node $e \in \mathcal{E}$ represents an entity, each $r \in \mathcal{R}$ represents a unique relation, and each directed edge $(e_{\mathrm{s}}, r, e_{\mathrm{o}}) \in \mathcal{F}$ represents a fact about the subject entity e_{s}.

Given a structured relational query $(e_{\mathrm{s}}, r_{\mathrm{q}}, ?)$, where e_{s} is the source entity, r_{q} is the query relation, and $(e_{\mathrm{s}}, r_{\mathrm{q}}, e_{\mathrm{o}}) \notin \mathcal{F}$, the goal is to find a set of plausible answer entities $\{e_{\mathrm{o}}\}$ by navigating paths through the existing entities and relations in $\mathcal{G}$ leading to answer entities. Note that, unlike previous methods that consider transductive settings with a static snapshot of the knowledge graph, we allow for dynamic knowledge graphs, where e_{s} may be an emerging entity, and therefore, previously unseen. Moreover, while embedding-based methods only deliver candidate answer entities, we here also seek the actual paths, i.e., sequences of nodes and edges for better interpretability.[1]

4.3.2 Graph Transformer for Inductive Representation Learning

The state-of-the-art embedding based models either focus on learning entity embeddings by using only the query relations, ignoring the subject entity's neighborhood, or use message passing neural networks to learn embeddings conditioned on neighboring entities and relations while being oblivious of the query relation. However, we observe that in many cases a new fact can be inferred by using another existing fact. For example, the fact *(PersonX, Place of Birth, Y)* can often help to answer to the query *(PersonX, Nationality, ?)*. Motivated by this observation, we propose a Graph Transformer architecture that learns the

[1] From here onwards, we will use the terms *node* and *entity*, as well as *edge* and *relation(ship)* interchangeably.

embedding of the source entity by iterative aggregation of neighborhood information (messages) that are weighted by their relevance to the query relation. To learn the relevance weights, our Graph Transformer model deploys *multi-head scaled dot product attention*, also known as *self-attention* [14].

Formally, we denote the local neighborhood for each entity $e_i \in \mathcal{E}$ as $\mathcal{N}_i$ such that $\mathcal{N}_i = \{e_j \mid e_j \in \mathcal{E} \wedge (e_i, r, e_j) \in \mathcal{F} \wedge r \in \mathcal{R}_{ij}\}$, where $\mathcal{R}_{ij}$ is the set of relations between entities e_i and e_j.

Each neighboring entity $e_j \in \mathcal{N}_i$ connected to e_i by a relation r sends in a message to entity e_i. The message $\mathbf{m}_{ijr}$ is a linear transformation of the fact (e_i, r, e_j) followed by the application of a non-linear function, specifically, the leaky rectified linear unit (LeakyReLU) function with a negative slope of 0.01. Formally,

$$\mathbf{m}_{ijr} = \text{LeakyReLU}(\mathbf{W}_f[\mathbf{e_i}; \mathbf{r}; \mathbf{e_j}]), \tag{4.1}$$

where $\mathbf{W}_f \in \mathbb{R}^{d \times 3d}$ is a shared parameter for the linear transformation and [;] is the concatenation operator.

To compute an attention head, our model performs linear projections of the query relation r_q, the neighborhood relations $r \in \mathcal{R}_{ij}$, and the neighborhood messages $\mathbf{m}_{ijr}$ to construct queries Q, keys K, and values V, respectively, such that $Q = \mathbf{W}_Q \mathbf{r}_\text{q}$, $K = \mathbf{W}_K \mathbf{r}$, and $V = \mathbf{W}_V \mathbf{m}_{ijr}$, where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d' \times d}$ are learnable parameters.

Next, we use the queries Q to perform a dot-product attention over the keys K. Formally,

$$\alpha_{ijr} = \frac{\exp((\mathbf{W}_Q \mathbf{r}_\text{q})^\top (\mathbf{W}_K \mathbf{r}))}{\sum_{z \in \mathcal{N}_i} \sum_{r' \in \mathcal{R}_{ij}} \exp((\mathbf{W}_Q \mathbf{r}_\text{q})^\top (\mathbf{W}_K \mathbf{r}'))} \tag{4.2}$$

We adopt the common procedure of scaling the dot products of Q and K by a factor of $\frac{1}{\sqrt{d'}}$ [14].

The attention weights are then used to aggregate the neighborhood messages. Note

that *self-attention* deploys multiple attention heads, each having its own query, key, and value projectors. The aggregated messages from N attention heads are concatenated and added to the initial embedding $\mathbf{e_i}$ through a residual connection to obtain new intermediate representation

$$\hat{\mathbf{e}}_\mathbf{i} = \mathbf{e_i} + \|_{n=1}^{N} \left(\sum_{j \in \mathcal{N}_i} \sum_{r \in \mathcal{R}_{ij}} \alpha_{ijr}^n \mathbf{W}_V^n \mathbf{m}_{ijr} \right), \tag{4.3}$$

where $\|$ is the concatenation operator.

Layer normalization (LN) [66] is applied to the intermediate representation $\hat{\mathbf{e}}_i$, followed by a fully connected two-layer feed forward network (FFN) with a non-linear activation (ReLU) in between. Finally, the output of the feed forward network is added to the intermediate representation through another residual connection. The resulting embedding is again layer normalized to obtain the new representation $\mathbf{g_i}^l$ for e_i. Formally,

$$\mathbf{g_i}^l = \mathrm{LN}(\mathrm{FFN}(\mathrm{LN}(\hat{\mathbf{e}}_\mathbf{i})) + \mathrm{LN}(\hat{\mathbf{e}}_\mathbf{i})) \tag{4.4}$$

This pipeline is called a *Transformer block*. Figure 4.2 represents a schematic diagram of a Transformer block in Graph Transformers. We stack L layers of Transformer blocks to obtain the final embedding $\mathbf{g_i}^L$ for e_i.

4.3.3 Policy Gradient for Explainable Reasoning

To infer the answer entity, we could leverage the entity representations obtained by the Graph Transformers. However, our goal is not only to infer the answer entity, but to find a symbolic reasoning path to support the inference. Following previous work [43, 12], we formulate the reasoning task as a finite horizon, deterministic partially observable Markov Decision Process (POMDP). A knowledge graph can be seen as a partially observable environment with out-going relations at each entity node corresponding to a set of discrete actions that an agent can explore to reach the target answer from the source entity.

Knowledge Graph Environment Formally, a Markov Decision Process is defined by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}$ captures state transition probabilities, and $\mathcal{R}$ is the reward function. In a knowledge graph environment, the state space is defined as a set of tuples $s_t = (e_t, r_{\mathrm{q}}) \in \mathcal{S}$, where e_t is an entity node in the knowledge graph, and r_{q} is the query relation. The action space $A_t \in \mathcal{A}$ for a state s_t is defined as the set of outgoing edges from entity node e_t in the knowledge graph. Formally, $A_t = \{(r_{t+1}, s_{t+1}) \mid (e_t, r_{t+1}, s_{t+1}) \in \mathcal{G}\}$. Since state transitions in a KG environment are deterministic, the transition probabilities $P(s_{t+1} \mid s_t, a_t) = 1 \; \forall P \in \mathcal{P}$. The agent receives a terminal reward of 1 if it arrives at the correct answer entity at the end.

Graph Search Policy To find a plausible path to the answer entity, the model must have a policy to choose the most promising action at each state. Note that in the KG environment, the decision of choosing the next action is not only dependent on the current state, but also on the sequence of observations and actions taken so far in the path. We use a multi-layer LSTM as a sequence encoder to encode the path history.

Formally, each state s_t is represented by a vector $\mathbf{s}_t = [\mathbf{e}_t; \mathbf{r}_q] \in \mathbb{R}^{2d}$ and each possible action $a_t \in A_t$ is represented by $\mathbf{a}_t = [\mathbf{e}_{t+1}; \mathbf{r}_{t+1}] \in \mathbb{R}^{2d}$, where $\mathbf{e}_t$, $\mathbf{e}_{t+1} \in \mathbb{R}^d$ are the embeddings of the entity nodes at timesteps t and $t+1$, respectively, that are obtained from Graph Transformer encoders. $\mathbf{r}_{t+1} \in \mathbb{R}^d$ is the embedding of an out-going relation from entity e_t, and $\mathbf{r}_q \in \mathbb{R}^d$ corresponds to the embedding of the query relation r_{q}. Each of these embeddings is also obtained from the Graph Transformer encoder. The path history is encoded as $\mathbf{h}_t = \mathrm{LSTM}(\mathbf{h}_{t-1}, \mathbf{a}_{t-1})$. Given the embedded action space $\mathbf{A}_t \in \mathbb{R}^{2|A_t|}$, i.e., the stacked embeddings of actions $a_t \in A_t$, and the path history $\mathbf{h}_t$, we define the parameterized policy as:

$$\pi_\theta(a_t \mid s_t) = \mathrm{Softmax}(\mathbf{A}_t(\mathbf{W_2}\mathrm{ReLU}(\mathbf{W_1}[\mathbf{h}_t; \mathbf{e}_t; \mathbf{r}_q])))$$

Policy Optimization. The policy network is trained to maximize the expected reward for all $(e_{\mathrm{s}}, r_{\mathrm{q}}, e_{\mathrm{o}})$ triples in the training sub-graph. The agent learns an optimal policy π_θ by exploring a state space of all possible actions. The objective of the agent is to take actions to maximize the expected end reward. Formally:

$$J(\theta) = \mathbb{E}_{(e_{\mathrm{s}}, r_{\mathrm{q}}, e_{\mathrm{o}})} \left[\mathbb{E}_{a_1, \ldots, a_{T-1} \sim \pi_\theta} [R(s_T | e_{\mathrm{s}}, r_{\mathrm{q}})] \right] \tag{4.5}$$

Since policy gradient uses gradient-based optimization techniques, the estimated gradient of the objective function can be derived as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{a_{1:T} \sim \pi_\theta} [\nabla_\theta \log \pi_\theta(a_{1:T} | e_{\mathrm{s}}, r_{\mathrm{q}}) R(s_T | e_{\mathrm{s}}, r_{\mathrm{q}})] \tag{4.6}$$

$$\approx \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta \log \pi_\theta(a^n_{1:T} | e_{\mathrm{s}}, r_{\mathrm{q}}) R \tag{4.7}$$

Here, N is the number of policy rollouts.

Each policy rollout explores a sequence of actions $a_{1:T}$. At each timestep $t \in \{1 : T\}$, the agent selects an action a_t conditioned on the current state s_t. Therefore, the gradient of the log-likelihood in Equation 4.7 can be expressed as

$$\nabla_\theta \log \pi_\theta(a_{1:T} | e_{\mathrm{s}}, r_{\mathrm{q}}) = \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t, e_{\mathrm{s}}, r_{\mathrm{q}}) \tag{4.8}$$

Reward Shaping. Previous work [12] observed that a soft reward for the target entities is more beneficial than a binary reward. Following their work, we use pre-trained ConvE [27] embeddings for the observed entities and relations to shape the reward function. If the agent reaches the correct answer entity, it receives reward 1. Otherwise, the agent receives a reward estimated by the scoring function of the pre-trained ConvE. Note that the ConvE model is trained only on the training sub-graph of seen entities. ConvE plays no role during inference. Its only purpose is to provide *soft reward* signals during training to help the model in learning a better policy.

4.4 Evaluation

4.4.1 Datasets

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{F}\rvert$			
				train	dev	test	aux
FB15k-237-Inductive	13,119	237	1,389	227,266	17,500	32,197	61,330
WN18RR-Inductive	35,928	11	4,029	67,564	3,000	11,015	19,395
NELL-995-Inductive	71,578	200	776	137,221	500	1,679	2,267

Table 4.1: Evaluation datasets for inductive setting

We evaluate our model based on three standard benchmark knowledge graph completion datasets. (1) FB15k-237 [116], introduced as a replacement for the FB15k dataset [20]. In FB15k-237, the reverse relations are removed, rendering the dataset more challenging for inference. (2) WN18RR [27] is a subset of the WN18 benchmark dataset. Similar to FB15k-237, the reverse relations are removed for this dataset. (3) NELL-995 [42] is a subset of the 995-th iteration of NELL.

To test the effectiveness of our model for inductive representation learning and reasoning, we create new splits of training, development, and test sets for each of the three benchmark datasets mentioned above. This new split of the data is necessary, as in an inductive setting, the subject entities in the test set must not be present anywhere in the training subgraph. To satisfy this requirement, we first sample 10% of all the entities present in each of the benchmark datasets. We denote this as the set of *unseen entities* $\mathcal{U}$, while the remaining entities are denoted as *seen entities* $\mathcal{E}$. Then, we proceed to split the triples in the datasets into three disjoint sets. The first set contains the triples in which both the head and the tail entities are in $\mathcal{E}$. The second set consists of the triples with head entities belonging to $\mathcal{U}$, but tail entities in $\mathcal{E}$. In the third set, the head entities belong to $\mathcal{E}$, but the tail entities are in $\mathcal{U}$. We further split the first set into *train* and *dev* triples. The second set becomes the *test* triples, and the union of the second and the third set is denoted as *auxiliary* data. Auxiliary triples are required to obtain the local neighborhood of a source entity at inference time.

Note that an emerging entity in the test set is not disconnected from the training graph. It has at least one seen entity in its neighborhood. This ensures that our model can find a path to the target entity during inference. If the emerging entity were completely disconnected from the training graph (i.e. all neighboring nodes were in $\mathcal{U}$), finding a path to the target entity would not be possible.

We append the suffix *"-Inductive"* to distinguish these newly derived datasets from their original counterparts. A summary of these datasets is presented in Table 4.1. To help with the reproducibility for future research on this topic, we make the datasets and our source code publicly available [2].

4.4.2 Baselines

Embedding-based Models We compare our model to a set of embedding based models that perform well under the transductive setting of link prediction. Although these models are particularly unsuitable for the inductive setting, we include them to better demonstrate the challenges of applying such algorithms in an inductive setting. In particular, we compare our model to ConvE [27], TransH [22], TransR [21], and RotatE [25]. For these experiments, we adapted the PyKEEN[3] implementations of these models.

Graph Convolution Models We choose a state-of-the-art graph convolution-based method CompGCN [35] as a baseline. Our choice is motivated by two factors: (1) CompGCN performs strongly in the transductive setting by outperforming the other baselines for most of the datasets, and (2) since its encoder module deploys neighborhood integration through Graph Convolution Networks, it has similar characteristics to our model, and therefore, is a good candidate for inductive representation learning. We also compare our model to R-GCN [32] and SACN [34], which also leverage the graph structure to learn node representations by aggregating neighborhood information. For CompGCN and SACN, we

[2] https://github.com/kingsaint/InductiveExplainableLinkPrediction
[3] https://github.com/pykeen/pykeen

adapted the source code made available by the authors to make them suitable for inductive representation learning and link prediction. For R-GCN, we adapted the source code available in the DGL library[4].

Symbolic Rule Mining Model We compare our model with AnyBURL [46], a purely symbolic rule mining system. AnyBURL is capable of extremely fast rule mining, has outperformed other rule mining approaches including AMIE+ [45], and produces comparable results to existing embedding-based models.

Path-based Model Finally, we compare our model to a policy gradient-based multihop reasoning approach [12] that is similar to the decoder module of our model. We modified the source code[5] of this model to adapt it to our task.

4.4.3 Experimental Details

Training Protocol. Since the benchmark knowledge graph completion datasets contain only unidirectional edges $(e_{\mathrm{s}}, r_{\mathrm{q}}, e_{\mathrm{o}})$, for all methods, we augment the training sub-graph with the reverse edges $(e_{\mathrm{o}}, r_{\mathrm{q}}^{-1}, e_{\mathrm{s}})$. During the Graph Transformer based inductive representation learning, $n\%$ of local neighboring entities are randomly selected and masked. During training, we mask 50%, 50%, and 30% of neighboring nodes, respectively, for the FB15k-237, WN188RR, and NELL-995 datasets. Neighborhood masking helps in learning robust representations and reduces the memory footprint, and has been shown to be effective [30]. Following previous work [43, 12], during training of the policy network, we also retain the top-k outgoing edges for each entity that are ranked by the PageRank scores of the neighboring entities. We set the value of k for each dataset following Lin et al. [12]. Such a cut-off threshold is necessary to prevent memory overflow. Finally, we adopt the false-negative masking technique in the final timestep of the policy rollouts to guide the

[4]https://github.com/dmlc/dgl/tree/master/examples/pytorch/rgcn
[5]https://github.com/salesforce/MultiHopKG

agent to the correct answer entities as described in previous work [43, 12], where it was found helpful when multiple answer entities are present in the training graph.

Hyperparameters For a fair comparison to the baselines, we keep the dimensionality of the entity and relation embeddings at 200. For our model, we deploy one layer of a Transformer block ($L = 1$) and 4 attention heads ($N = 4$). We choose a minibatch size of 64 during training due to limited GPU memory. We rely on Adam [117] stochastic optimization with a fixed learning rate of 0.001 across all training epochs. Additionally, we adopt entropy regularization to improve the learning dynamics of the policy gradient method. The regularizer is weighted by a hyperparameter β set to a value within $[0, 0.1]$. We apply dropout to the entity and relation embeddings, the feedforward networks, and the residual connections. The policy rollout is done for $T = 3$ timesteps for every dataset.

Evaluation Protocol Following previous work [12], we adopt beam search decoding during inference with a beam width of 512 for NELL-995 and 256 for the other datasets. If more than one path leads to the same target entity, then the path with the maximum log-likelihood is chosen over the others. During evaluation, the auxiliary graph augments the training graph to construct the KG environment with unseen entities and their relations to the seen entities. For our model and the baselines, the embeddings of all unseen entities are initialized with Xavier normal initialization [118] at inference time.

Evaluation Metrics We adopt the ranking based metrics *Mean Reciprocal Rank* and *Hits@k* that are also used by prior work for evaluation. We follow the *filtered setting* [20] adopted by prior approaches. In the filtered setting, the scores for the false negative answer entities are masked to facilitate correct ranking of the target entity.

	WN18RR-Inductive				FB15K-237-Inductive				NELL-995-Inductive			
		Hits@N				Hits@N				Hits@N		
Model	MRR	@1	@3	@10	MRR	@1	@3	@10	MRR	@1	@3	@10
TransR [21]	0.8	0.6	0.7	0.9	5.0	4.0	5.2	6.6	5.3	4.9	5.3	6.5
TransH [22]	0.0	0.0	0.0	0.0	6.2	5.4	6.3	8.0	3.6	3.4	3.6	3.6
RotatE [25]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ConvE [27]	1.9	1.1	2.1	3.5	26.3	20.0	28.7	38.8	43.4	32.5	50.3	60.9
R-GCN [32]	14.7	11.4	15.1	20.7	19.1	11.5	20.9	34.3	58.4	50.9	62.9	71.6
SACN [34]	17.5	9.7	20.3	33.5	29.9	20.5	32.8	50.0	42.4	37.0	42.9	53.2
CompGCN [35]	2.2	0.0	2.2	5.2	26.1	19.2	28.5	39.2	42.8	33.1	47.9	61.0
AnyBURL [46]	-	**48.3**	50.9	53.9	-	28.3	43.0	56.5	-	8.7	11.0	12.3
MultiHopKG [12]	45.5	39.4	49.2	56.5	38.6	29.3	43.4	56.7	74.7	69.1	78.3	84.2
Our Model w/ RS	**48.8**	42.1	**52.2**	**60.6**	**39.8**	**30.7**	**44.5**	**57.6**	**75.2**	**69.7**	**79.1**	**84.4**

Table 4.2: Evaluation results of our model as compared to alternative baselines on inductive variants of the WN18RR, FB15K-237, and NELL-995 datasets. The Hits@N and MRR metrics are multiplied by 100.

4.4.4 Results

We present the experimental results of our method and the baselines in Table 4.2. The results of the embedding-based models TransH, TransR, and RotatE across all datasets demonstrates their inability to deal with entities that are unseen during training. These models are thus rendered as ineffective for inductive representation learning and reasoning. ConvE performs better than other embedding-based models we consider. Still, the much inferior performance of ConvE compared to our model shows that ConvE is not particularly suitable for inductive representation learning.

We observe that our model significantly outperforms the graph convolution network baselines CompGCN, SACN, and R-GCN across all datasets. Although these models use the neighborhood information for learning representations, unlike our method, their neighborhood integration methods do not explicitly consider the query relations.

We find AnyBURL and MultiHopKG to be the most competitive methods to ours. AnyBURL performs adequately for the WN18RR and FB15K-237 dataset while performing poorly on the NELL-995 dataset. MultiHopKG adapts surprisingly well to our dataset despite the unseen entities being initialized with Xavier normal initialization. We conjecture that the learned representations of the query and the outgoing edges (relations) have enough

	WN18RR-Inductive				FB15K-237-Inductive				NELL-995-Inductive			
		Hits@N				Hits@N				Hits@N		
Model	MRR	@1	@3	@10	MRR	@1	@3	@10	MRR	@1	@3	@10
Our Model w/ RS	**48.8**	42.1	**52.2**	**60.6**	**39.8**	**30.7**	**44.5**	**57.6**	**75.2**	**69.7**	**79.1**	**84.4**
Our Model w/o RS	48.2	40.1	53.0	62.2	37.8	29.4	42.6	54.0	71.1	65.3	75.0	79.9
GT + ConvTransE	1.1	0.6	1.1	1.8	22.9	17.3	24.9	33.3	47.9	40.6	51.1	61.9

Table 4.3: Ablation study. The Hits@N and MRR metrics are multiplied by 100.

semantic information encoded in them to navigate to the target entity by simply exploiting the edge (relation) information. However, our proposed model holds an edge over this model with 7.2%, 3.1%, and 0.7% gains in the MRR metric for the WN18RR, FB15K-237, and NELL-995 datasets respectively.

4.5 Analysis

In this section, we perform further analysis of our proposed model. First, we conduct a set of ablation studies. Then, we qualitatively analyze our model's ability to provide reasoning paths as supporting evidence for inference. Finally, we analyze the effect of the cardinality of relation types on the inference process.

4.5.1 Ablation Study

To better understand the contribution of reward shaping in our model, we perform an ablation study, where our model is deprived of the *soft reward* signals provided by ConvE. In general, we observe that replacing reward shaping with hard binary reward deteriorates the performance of our model across all datasets. Note that our ablated version still mostly outperforms the other baseline methods.

Additionally, we experiment with a non-explainable variant of our model, in which we retain the Graph Transformer (GT) encoder, but we replace the policy gradient-based decoder with an embedding-based decoder called ConvTransE, which is also used in SACN as a decoder module. With this model, we observe a significant drop in performance. Thus, we conjecture that the policy gradient-based decoder not only provides explainability, but

also is crucial for decoding.

4.5.2 Qualitative Analysis of Explainability

Since explainability is one of the key objectives of our model, we provide examples of explainable reasoning paths for queries that involve previously unseen source entities at inference time. Table 4.4 contains examples of 1-hop, 2-hop, and 3-hop reasoning paths. These examples demonstrate our model's effectiveness in learning inductive representations for the unseen entities, which helps to infer the reasoning paths.

Query	(William Green, worksFor, ?)
Answer	Accenture
Explanation	William Green $\xrightarrow{\text{personLeadsOrganization}}$ Accenture
Query	(Florida State, organizationHiredPerson, ?)
Answer	Bobby Bowden
Explanation	Florida State $\xleftarrow{\text{worksFor}}$ Bobby Bowden
Query	(Messi, athleteHomeStadium, ?)
Answer	Camp Nou
Explanation	Messi $\xrightarrow{\text{athletePlaysForTeam}}$ Barcelona $\xrightarrow{\text{teamHomeStadium}}$ Camp Nou
Query	(Adrian Griffin, athleteHomeStadium, ?)
Answer	United Center
Explanation	Adrian Griffin $\xrightarrow{\text{athletePlaysForTeam}}$ Knicks $\xleftarrow{\text{athletePlaysForTeam}}$ Eddy Curry $\xrightarrow{\text{athleteHomeStadium}}$ United Center
Query	(Bucks, teamPlaysInLeague, ?)
Answer	NBA
Explanation	Bucks $\xrightarrow{\text{organizationHiredPerson}}$ Scott Stiles $\xleftarrow{\text{organizationHiredPerson}}$ Chicago Bulls $\xrightarrow{\text{teamPlaysInLeague}}$ NBA

Table 4.4: Example queries from the NELL-995 test set with unseen source entities. The answers are supported by the explainable reasoning paths derived by our model.

4.5.3 Effect of Relation Types

Following Bordes et al. [20], we categorize the relations in the seen snapshot of the knowledge graph into Many-to-1 and 1-to-Many relations. The categorization is done based on the ratio of the cardinality of the target answer entities to the source entities. If the ratio is

Dataset	to-Many		to-1	
	%	MRR	%	MRR
FB15k-237-Inductive	77.4	31.6	22.6	75.5
WN18RR-Inductive	48.1	60.8	51.9	30.1
NELL-995-Inductive	7.6	41.4	92.4	78.5

Table 4.5: MRR for the test triples in inductive setting with *to-Many* and *to-1* relation types. The % columns show the percentage of test triples for each relation type.

greater than 1.5, we categorize the relation as *to-Many*, otherwise as *to-1*. We analyzed the results of the test set for these two types of relations. We report the percentage of triples with these two types of relations and the corresponding MRR achieved by our model in Table 4.5. For FB15k-237 and NELL-995, our model performs better for *to-1* relations than for *to-many* relations. On the contrary, we observe a reverse trend for the WN18RR dataset. Note however that *to-many* relations have alternative target entities. In the current evaluation protocol, our model is punished for predicting any alternative target entity other than the ground truth target.

4.6 Discussion

In this chapter, we proposed an end-to-end trainable framework for explainable link prediction for emerging entities. New emerging entities are continuously added to open-domain knowledge graphs such as Wikidata. Predicting missing links for these entities help to enhance the fact coverage of a knowledge graph. To this end, our proposed method has two clear advantages. First, our model's inductive representation learning obviates the need to retrain the model for link prediction when new emerging entities are added to a knowledge graph. Secondly, it provides explainable reasoning paths which a human judge can use to verify the correctness if necessary.

Although our proposed method performs better than the baselines, the empirical results suggest there is a scope for further improvements. A key disadvantange of policy gradient-based reasoning is that the RL agent often needs to explore in large action space due to

large out-degrees of some nodes. Reducing the size of the action space could make the path exploration process more efficient. To this end, one can leverage the schema structure of the knowledge graph to find the reasoning paths.

Another limitation of our proposed model is that our model can not verify the quality of the rules in terms of semantic correctness. It might be possible that our model finds a spurious path that does not yield a semantically correct explanation. A thorough human evaluation of the explanation paths generated by a our model could reveal some of these spurious paths. However, large scale human evaluation is expensive and is beyond the scope of this research work.

CHAPTER 5
KNOWLEDGE RETRIEVAL FOR ENTITY SUMMARIZATION

5.1 Overview

Motivation Entity-centric search queries that solicit specific factual information about an entity (e.g., a person, place, organization, etc.) constitute a significant proportion of all search queries processed by the popular search engines. Almost all modern search engines incorporate facts from an underlying knowledge graph [2] that acts as a reliable source of factual information. Most notably, in recent years, knowledge panels have become an integral part of the contemporary search engines. Knowledge panels are information boxes that appear on the search engine result pages when a user searches for entity-related information. Such cards provide a series of facts taken from the knowledge graph and enable the user get a brief overview of pertinent key facts about the entity without the need to navigate to various Web pages. Although knowledge panels are commonplace in contemporary search engines, it is not very well understood how the search engines determine which pieces of information ought to be retrieved from the underlying knowledge graph and incorporated into the knowledge panels.

Goal In practice, different entity-related queries may pertain to quite different aspects of an entity. A search engine query such as *einstein education* ought to give preference to other facts than a query such as *einstein family*. Recent work by Hasibi et al. [58] makes an effort to explain and evaluate how knowledge panels are dynamically populated to cater to such distinct information needs of different search engine users. They argue that the factual information that is shown on the knowledge panels is often determined by two measures: the *importance* and *relevance*. Indeed, importance and relevance are two mutually complementary measures bearing very different information. A fact that is deemed important in

general about an entity may be irrelevant in a given query context and vice versa. For instance, for the query *einstein education*, Einstein's alma mater and academic achievements merit a high ranking, whereas for the query *einstein family*, siblings, spouses, and children might be preferred.

Approach To address this task of dynamic query-specific fact ranking, Hasibi et al. propose a model called DynES [58] that performs fact retrieval and entity summarization based on a linear combination of both importance and relevance, and compared the results to human judgements.

However, there are several points about DynES worth noting. DynES is based on hand-crafted features extracted from the input query and from the knowledge graph. Many of these features are statistical ones that need to be extracted beforehand from the set of all facts in the large-scale knowledge graph, rendering this method unsuitable for ad hoc settings, i.e., sets of candidate facts different from those in the knowledge graph. Moreover, DynES performs a simple pointwise ranking of the facts, where each fact is considered in isolation, using Gradient Boosted Regression Trees, which learn an ensemble of weak prediction models.

In this chapter, we propose a novel model that obviates the need for a cumbersome process of designing hand-crafted features based on the large knowledge graph. Similar to DynES, our proposed model accounts for both the *importance* and *relevance* of each fact with respect to the query. Instead of hand-crafted features, we exploit the linguistic and semi-linguistic nature of the search query and the candidate facts. Our model implicitly learns the semantic relatedness between the search queries and the facts to determine the importance and relevance of a fact with respect to the query. In addition, we formulate the resulting *learning to rank problem* as a pairwise and a pointwise ranking problem, and compare their performances across different metrics.

Contributions This chapter is based on a published work [119]. The key contribution of this chapter are the following.

1. We propose a novel deep neural model with a pointwise and a pairwise loss function to address the task of query-dependent fact retrieval for entity-centric search queries.

2. Rather than requiring a large knowledge graph from which various forms of statistics and features are extracted, our model draws on recent advances in Transformers with self attention [14] to better model the linguistic connection between the query and the candidate facts, and thus can be applied even to entirely novel sets of candidate facts.

5.2 Preliminaries

In this section, we first define relevant terminology that is used throughout the remainder of this chapter.

We consider a fact as a predicate–object pair returned when a query is made with regard to an entity, with that entity serving as the subject.

Definition 1. *(Fact) Given a subject entity s, a fact $f = \langle p, o \rangle$ is a tuple of predicate p and object o for a subject–predicate–object triple $\langle s, p, o \rangle$ in the knowledge graph.*

We now define the importance and relevance of such facts, given an entity and a query.

Definition 2. *(Importance) Importance is an attribute of a fact f that determines its relation to the subject entity s in absolute terms, irrespective of the provided query. It is denoted as $i_s(f)$.*

Definition 3. *(Relevance) Relevance, in turn, describes to what extent a given candidate fact f is pertinent with regard to a given natural language search query q issued by the user along with the entity s as the subject. It is denoted as $r_{s,q}(f)$.*

However, a fact with a high degree of importance may not be particularly relevant with regard to the query context, while a highly relevant fact may not be as important. Hence, we consider *utility* as the overall ranking criterion. The experiments presented in this chapter follow previous work [58] in adopting a notion of utility that is defined as a weighted sum of importance and relevance.

Definition 4. *(Utility) The overall utility of a fact f with respect to a query q and entity s is defined as a weighted sum of the importance and relevance scores of the fact with respect to query and entity. It is denoted as $u_{s,q}(f)$ and computed as*

$$u_{s,q}(f) = \alpha\, i_s(f) + \beta\, r_{s,q}(f) \tag{5.1}$$

Without loss of generality, we follow Hasibi et al. [58] in assuming that $\alpha = \beta = 1$ for simplicity. However, α, β may be adjusted freely to account for application scenario-specific considerations.

Thus, utility relates the fact to the query in a more comprehensive manner than the importance and relevance scores alone can.

Definition 5. *(Fact Ranking) For a given natural language query q and a given target subject entity s, we consider a set $\mathcal{F} = \{f_1, \ldots, f_n\}$ of n candidate facts to be ranked, each taking the form of an $\langle p, o \rangle$ pair. The goal is to find a ranking function g that, based on inputs of this form, obtains an ordinal ranking of the facts with respect to a normalized score in descending order.*

In practice, we learn a function g that predicts importance, relevance, or utility scores. Like previous work, we consider a supervised setup, and rely on ground truth rankings for example queries and entities to guide the learning process.

5.3 Model

Given the natural language input query Q as well as a candidate fact $f_i = \langle p, o \rangle \in \mathcal{F}$, where $\mathcal{F}$ is the set of all candidate facts for Q, our model accepts the query along with the natural language labels of p and o and invokes BERT [67], a deep neural Transformer encoder, to encode bidirectional contextual information for the given sequence of input tokens. Since we simultaneously supply both the query and the candidate fact to the Transformer, the self-attention layers are able to establish connections between (parts of) these two inputs.

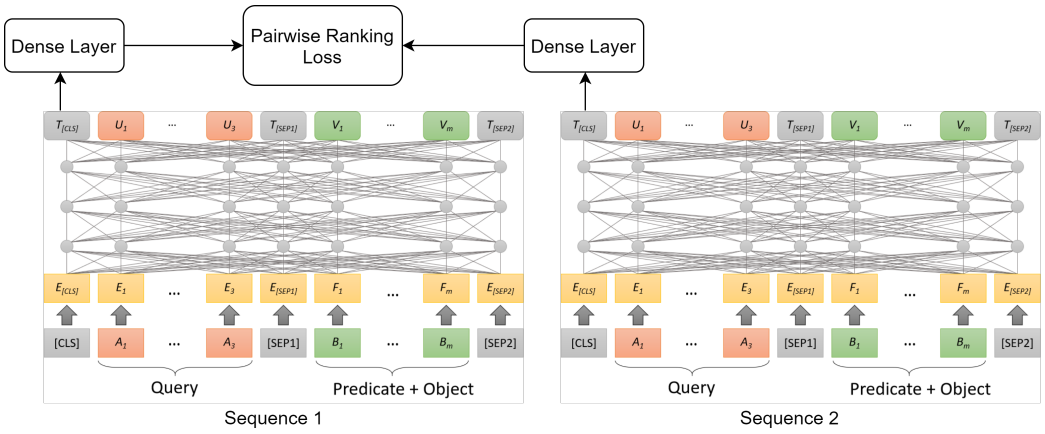


Figure 5.1: A schematic diagram of the model architecture.

Before feeding the tokenized sequences Q, P, O to the model, special tokens `[CLS]` and `[SEP]` are inserted into the input sequence. The `[CLS]` token signifies the start of each sequence, while the `[SEP]` token serves as a demarcation point separating the query segment from the fact segment in the input sequence. The resulting sequence of input token identifiers now becomes:

$$\texttt{[CLS]}, q_1, \ldots, q_l, \texttt{[SEP]}, p_1, \ldots, p_m, o_1, \ldots, o_n, \texttt{[SEP]}$$

The most relevant component for our task lies in the encoded representation of the `[CLS]` token, which serves as a representation of the entire input sequence. This representation is passed through a fully-connected layer followed by a sigmoid activation function to yield a ranking score, which is then compared to the ground truth. Formally,

$g(f_i) = \sigma(\mathbf{W}\mathbf{h_p} + b)$, where $\mathbf{h_p} \in \mathbb{R}^d$ is the `[CLS]` representation from the final hidden layer of the BERT encoder, $\mathbf{W} \in \mathbb{R}^{1\times d}$ and b are trainable parameters, and $\sigma(x) = \frac{1}{1+e^{-x}}$. Note that the predicted ranking scores and the ground truth ranking scores are always in the range of $[0, 1]$.

Pointwise Ranking Many ranking methods, including DynES, use a pointwise ranking approach. A pointwise ranking loss considers only one fact at a time and calculates the training loss using a mean squared error between the ground truth ranking score $r(i)$ and the predicted score $g(f_i)$. Formally,

$$\mathcal{L}(g; \mathcal{F}, \mathcal{R}) = \sum_{i=1}^{n} \big(r(i) - g(f_i)\big)^2 \tag{5.2}$$

Pairwise Ranking A pairwise ranking loss that considers pairs of facts (f_s and f_i) and encourages the model to predict scores for the two involved facts that reflect the correct relative ordering between them. Ideally, the difference between the two predicted scores ($g(f_s) - g(f_i)$) should equal the difference ($r(s) - r(i)$) between the corresponding ground truth ranking scores. These differences are computed as signed values rather than absolute values, so the ordering is crucial. Based on this intuition, we define the loss function as a pairwise mean squared error as follows:

$$\mathcal{L}(g; \mathcal{F}, \mathcal{R}) = \sum_{s=1}^{n-1} \sum_{\substack{i=1 \\ r(i)<r(s)}}^{n} \Big[\big(r(s) - r(i)\big) - \big(g(f_s) - g(f_i)\big)\Big]^2 \tag{5.3}$$

The final ranking is created by ordering the candidate facts $f_i \in \mathcal{F}$ by $g(f_i)$ in descending order, breaking ties arbitrarily. Thus, if $g(f_i) > g(f_j)$, then f_i should be ranked higher than f_j.

5.4 Evaluation

To assess the merits of our proposed model, we conduct a detailed experimental evaluation comparing it against a series of baseline methods.

5.4.1 Dataset

We perform our experiments on the dataset put forth by Hasibi et al. [58] for fact ranking. It was collected by using the CrowdFlower platform. The human annotators were given a set of queries along with the corresponding facts for a given target entity, and were asked to rate the importance and relevance of the facts with respect to the entity as well as the query. The importance of a fact was measured on a 3-point scale (0 – not important, 1 – important, 2 – very important). Similarly, the relevance scores were also measured on a 3 point scale (0 – irrelevant, 1 – relevant, 2 – very relevant). All the entries in the dataset were annotated by 5 different workers. The Fleiss' Kappa inter-annotator agreement was moderate – 0.52 for importance, and 0.41 for relevance. Finally, the utility scores were obtained as a linear combination of the importance and relevance scores as described in Equation 5.1. The utility scores are measured on a 5 point Likert scale ranging from 0 to 4. We perform min-max normalization of the importance, relevance, and utility scores so that the ranking scores are always between $[0, 1]$. Note that while some of the queries take the form of a list query such as *Which European countries have a constitutional monarchy?*, these queries are considered with respect to a single target entity such as *Norway*, and the desired output is not a list of entities but rather a ranking of facts about this single target entity. Thus, the task setting differs substantially from that of Question Answering over Linked Data.

We observe that nearly 42% of the object values are of numerical type, and another 5% consist of a combination of characters and numbers. To improve our model's generalizability, we replace such values in the dataset with placeholders `[NUM]` and `[ALPHANUM]`,

respectively.

We consider two different variants of this dataset, again following Hasibi et al. [58]. For the first variant, *Complete Dataset*, the entire dataset is considered. The second variant, *URI-only Dataset*, keeps only the subset of facts for which the objects are genuine entities denoted by a URI, while facts with literal values are omitted.

Statistics of the two dataset variants are given in Table 5.1.

Variant	#Queries	#Facts	Avg. #Facts per Query
Complete Dataset	100	4,069	41
URI-only Dataset	100	1,309	14

Table 5.1: Dataset Statistics

5.4.2 Baselines

To evaluate our model, we compare it against several baseline models, including DynES [58], a BiLSTM Dual Encoder, as well as two other variants of our proposed model. We describe these baseline models in the following.

Entity Summarization methods As mentioned, the task we consider in this paper is distinct from regular entity summarization, due to the requirement of considering natural language queries. Still, for comparison, we include results of entity summarization methods that can be adapted to our setting. Specifically, we include the RELIN [54] entity summarization method, as well as the *URI-only Dataset* results of the LinkSum [57] and SUMMARUM [55] entity summarizers, as reported by Hasibi et al. [58].

DynES DynES [58] is a feature-based approach that relies on hand-crafted importance and relevance features, and employs Gradient Boosted Regression Trees as the learning model, which is known to be one of the best-performing algorithms for ranking problems. DynES performs pointwise ranking, as the model learns to produce a single ranking score

per fact. The number of trees is set to 100 and the depth is 3 when the model is trained on all features, while the depth is 2 when trained on importance or relevance feature columns.

BiLSTM Dual Encoder Before the advent of BERT, BiLSTMs were considered as the de facto model of choice for sequence encoding. Therefore, for this baseline, we deploy two BiLSTM encoders to encode the query and the fact independently. Formally,

$$\mathbf{h^q} = \text{BiLSTM}(Q) \text{ and } \mathbf{h^f} = \text{BiLSTM}([P; O]) \tag{5.4}$$

where $[;]$ is the concatenation operation. The first and the last hidden states of each BiLSTM are concatenated to obtain representations for the input query and fact.

$$\mathbf{Q} = [\mathbf{h^q}_1; \mathbf{h^q}_l] \text{ and } \mathbf{F} = [\mathbf{h^f}_1; \mathbf{h^f}_{m+n}] \tag{5.5}$$

A dot product between these two vectors yields the pointwise ranking score, i.e., $\psi(Q, F) = \mathbf{Q}^\top \mathbf{F}$. For this experiment, we deploy BiLSTMs with an input embedding and hidden state dimensionality of 512. Note that here the input tokens to the BiLSTM are word-level tokens. The model is trained end-to-end using pointwise Mean Squared Error (MSE) as the loss function (described below).

BERT$_{\text{BASE}}$ Our model fine-tunes a BERT encoder using a custom pairwise ranking loss. However, since BERT is pre-trained on large text corpora, it can readily be used to determine the semantic relatedness between a pair of input sequences. Thus, we conduct additional experiments without fine-tuning BERT to evaluate (1) how well the pre-trained BERT can perform for this specific task, and (2) to what extent we obtain gains after fine-tuning. Note that, for this experiment, we keep the parameters of the pretrained BERT unchanged. Thus, the only trainable parameters are those of the scoring layer.

5.4.3 Evaluation Metric

For evaluation, the data is initially grouped by query. Then for each query, the entire set of candidate facts is passed through the model to obtain the ranking scores for each fact given the query.

To quantitatively measure the predicted ranks in comparison to the ground truth ranks, we consider the standard Normalized Discounted Cumulative Gain (NDCG) metric [120]. NDCG aggregates the relevance of chosen items (facts) at every position in the ranking, using a discounting factor that is logarithmically proportional to the rank, such that higher ranks carry a more substantial weight, and with additional normalization with respect to the ideal ranking, such that the final NDCG scores lie in $[0, 1]$, where 1 is the optimal result. We consider ranked lists of length 5 (NDCG@5) and 10 (NDCG@10).

5.4.4 Experimental Details

For all BERT models in our experiments, the maximum sequence length is set to 128 word-piece tokens. For the BiLSTM Dual Encoder, the maximum sequence length is set to 50 word-level tokens. All BERT models are trained for 10 epochs with a learning rate of $1e^{-5}$. The BiLSTM Dual Encoder model is also trained for 10 epochs with a learning rate of $1e^{-4}$. We use Adam [117] as the optimizer for both BiLSTM and BERT models. For every model, we perform 5-fold cross-validation for both the datasets and report the mean NDCG scores of the best performing epoch of every fold. We choose the best performing epoch for each fold that gives us the highest NDCG@10 score on the held-out set.

5.4.5 Results

We present the results of our experiments for the *Complete Dataset* and *URI-only Dataset* in Table 5.2 and Table 5.3, respectively.

Overall, we observe that our model outperforms all baselines. We analyze and discuss these results in the following.

	Utility		Importance		Relevance	
Model	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
RELIN †	0.4680	0.5322	0.4733	0.5261	0.3514	0.4255
DynES †	0.7547	0.7873	0.7672	0.7792	0.5771	0.6423
Bi-LSTM Dual Encoder	0.4787	0.5572	0.5245	0.5703	0.4708	0.5219
$\text{BERT}_{\text{BASE}}$	0.5959	0.6476	0.5655	0.6162	0.4964	0.5571
Our Model (pointwise)	**0.8137▲**	**0.8432▲**	**0.8742▲**	**0.8735▲**	0.5974▲	0.6568▲
Our Model (pairwise)	0.7753▽	0.8065▽	0.8643▲	0.8577▲	**0.6099▲**	**0.6609▲**

Table 5.2: 5-fold cross-validation results on the *Complete Dataset*. † results are taken from Hasibi et al. [58]. The best results are highlighted in bold face and the second best results are underlined. ▲ indicates statistical significance (p-value < 0.05) w.r.t. DynES and ▽ indicates the results are not statistically significant (p-value $>= 0.05$) w.r.t. DynES. We use Student's paired t-test to determine statistical significance.

	Utility		Importance		Relevance	
Model	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
RELIN †	0.6300	0.7066	0.6368	0.7130	N/A	N/A
LinkSum †	0.6504	0.6648	0.7018	0.7031	N/A	N/A
SUMMARUM †	0.6719	0.7111	0.7181	0.7412	N/A	N/A
DynES †	0.8164	0.8569	0.8291	0.8652	N/A	N/A
Bi-LSTM Dual Encoder	0.6594	0.7398	0.6912	0.7589	N/A	N/A
$\text{BERT}_{\text{BASE}}$	0.7103	0.7761	0.7246	0.7832	0.5563	0.6372
Our Model (pointwise)	**0.8543▲**	**0.8777**▽	**0.8734▲**	**0.8948▲**	**0.6693**	**0.7264**
Our Model (pairwise)	0.8459▽	0.8719▽	0.8635▽	0.8826▽	0.6422	0.7042

Table 5.3: 5-fold cross-validation results on the *URI-only Dataset*. † results are taken from Hasibi et al. [58], who did not report separate relevance prediction results apart from the overall utility prediction results. The best results are highlighted in bold face and the second best results are underlined. ▲ indicates statistical significance (p-value < 0.05) w.r.t. DynES and ▽ indicates the results are not statistically significant (p-value $>= 0.05$) w.r.t. DynES. We use Student's paired t-test to determine statistical significance.

Comparison with DynES Approach Our model with a pointwise ranking loss outperforms the DynES model with 7.1% and 12.1% absolute gain in terms of the NDCG@10 metric on the *Complete Dataset* for the utility and importance-based rankings, respectively. We observe a similar trend for the *URI-only Dataset*, where our model with a pointwise ranking loss consistently outperforms the DynES model, with respective absolute gains of 4.6% and 5.3% in the NDCG@5 metric for the utility and importance-based rankings.

It is important to note that our model learns to produce the ranking based only on the actual training data, without any access to extrinsic information from the knowledge graph. DynES, in contrast, draws extensively on various kinds of statistics computed from the knowledge graph, such as the frequency of a predicate with subjects with the same

type as the current entity. Our method achieves superior results without consulting the knowledge graph at all, considering only the current set of candidate facts. Thus, one could likely obtain even higher results by augmenting our scoring layer to consider such extrinsic signals as additional features.

Importance of BERT Transformer The BiLSTM Dual Encoder model performs much worse than our model across all metrics and datasets. This suggests that cross-attention between the query and the facts is important to capture their semantic connections. LSTMs, in contrast, proceed in a unidirectional manner. Even if the left-to-right and right-to-left context layers are concatenated in a BiLSTM, this still results in a weaker form of context representation than that of the self-attention in BERT. Another likely reason for the weaker results of the BiLSTM model is that it does not benefit from any linguistic pre-training.

The moderate performance of the pre-trained $\text{BERT}_{\text{BASE}}$ model suggests that $\text{BERT}_{\text{BASE}}$ already has sufficient linguistic information embedded in it to be able to rank the facts to a certain degree. In fact, without any fine-tuning, $\text{BERT}_{\text{BASE}}$ outperforms the BiLSTM Dual Encoder baseline in most of the cases.

Comparison of Pointwise vs. Pairwise Ranking While the pairwise ranking variant of our model yields slightly better results in relevance-based ranking of the complete dataset, we observe that, in general, the performance of our model with a pointwise ranking loss is better than the pairwise variant. Moreover, the pointwise variant of our model consistently produces statistically significant improvements over DynES as shown in Table 5.2 and Table 5.3.

Prior work [121] on learning-to-rank problem has shown the advantage of pairwise ranking loss over pointwise ranking loss. However, [122] has shown that the loss functions of these methods are upper bounds of the measure-based ranking errors. In other words, minimizing these loss functions will lead to the maximization of the ranking measures. In our experiments, pairwise ranking loss reaches a local minima that is worse than the

pointwise ranking loss. We conjecture that this phenomenon is specific to the datasets that we considered for our experiments.

5.5 Related Work

Entity Retrieval In traditional information retrieval, given a natural language query, a corresponding ranked list of relevant natural language documents is sought [123]. In recent decades, a number of different forms of knowledge-driven information retrieval have been studied [124]. The well-known task of Question Answering over Linked Data [125] considers full-length natural language questions for which one or more entities may serve as answers. Typically, this involves transforming the natural language query into a complex structured query that can be executed to retrieve relevant answer entities from the knowledge store. Another line of work considers simpler keyword searches over structured data [126, 127], seeking to automatically infer suitable structured queries. In this work, we consider arbitrary natural language queries such as *tango dance history* that cannot necessarily be converted into a specific structured query. Rather, we assume that the query is entity-centric and that we need to select among the facts that the knowledge base provides for the target entity, considering the natural language query to determine which of those facts are most relevant.

Entity Summarization The task of fact-based entity summarization involves selecting a subset of facts about a real-world entity that can serve as a summary about that entity. The cardinality of this subset is restricted so as to obtain a concise summary. Numerous different approaches have been proposed for entity summarization. These include but are not limited to RELIN [54], SUMMARUM [55], the Langer et al. method [56], and LinkSUM [57]. However, as mentioned earlier, this task setting differs from the task that we consider in this chapter in that the ranking is based on a generic notion of importance. For example, several of the aforementioned approaches apply the PageRank algorithm to obtain generic

importance scores based on the structure of the knowledge graph. In contrast, we consider natural language queries such as "*Niagara falls origin lake*" and seek a ranking that also accounts for the relevance with regard to such queries.

Fact Rankings for Entity-Centric Queries The task we consider is the one proposed by Hasibi et al. [58]. Given a natural language query and an entity along with the corresponding set of candidate facts, the goal is to rank these candidate facts with respect to the query. Their proposed method applies Gradient Boosted Regression Trees, i.e., an ensemble of weak prediction models to learn a simple pointwise ranking score prediction function. Yet, this model is able to outperform previous entity summarization approaches. Thus, we consider DynES as our primary baseline model in our experiments.

Learning to Rank There is a long history of research on machine learning methods to learn rankings. Pointwise approaches separately assign each item a score. For instance, McRank [128] uses gradient boosting trees to learn the ranking scores. [129] consider SVM hyperplanes to draw the boundary between different prediction labels. PRank [130] uses a perceptron-based approach with stochastic gradient descent. Subset ranking [131] introduces the idea of solving the problem of ranking as a regression based one by substituting a fill-in loss function that serves as an upper bound on the DCG based non-convex optimization problem.

Compared to pointwise approaches, which consider each candidate item in isolation, pairwise approaches explicitly learn a comparative notion of ranking by comparing different items and predicting the relative ranks.

To this end, RankNet [121] performs pairwise ranking by training a neural ranking function. In another approach, RankBoost [132] performs pairwise ranking using boosting as the learning method. Our work proposes a deep neural architecture that is specifically tailored to our task and learns to draw connections between the query and the candidate facts.

Another line of approach is to learn a list-wise ranking by considering entire lists of ranked items as training instances. However, it is non-trivial to collect large amounts of ground truth ranked lists.

5.6 Discussion

In Table 5.4, we provide examples of rankings predicted by our method (with pairwise ranking loss), along with the corresponding ground truth rankings and utility scores. We observe that, in most of the cases, our model correctly predicts the fact with the highest utility scores. Additionally, we observe that even in cases when our model fails to correctly identify the top fact, there is no severe rank inversion, and the top fact is often present in the top-5 predicted ranks, as seen for the query *directed bela glen glenda bride monster plan 9 outer space* in the table.

Dynamic fact retrieval for entity-centric search queries using a supervised pairwise ranking methods yields us the state-of-the-art performance. However, it is non-trivial to obtain large-scale annotated training data that covers a wide variety of user queries. Unavailabiliy of large-scale training data is indeed a practical limitation for the proposed model. Future research in this direction should focus on obtaining human annotated data at scale through crowd sourcing.

In our work, we assumed that gold-standard subject entity is known beforehand. For practical use, one must use an entity disambiguation method to identify the subject entity mentioned in a search query. To this end, the end-to-end entity linking method proposed in Chapter 3 can be useful.

Search Query	Subject Entity	Predicted Ranking Order		Ground Truth Ranking Order		Utility
1998 United States embassy bombings	american embassy nairobi	location	nairobi	location	nairobi	3
		location	tanzania	location	kenya	3
		location	kenya	location	tanzania	2
		location	dar es salaam	location	dar es salaam	2
		weapons	stun grenade	weapons	stun grenade	1
directed bela glen glenda bride monster plan 9 outer space	Bride of the Monster	writer	ed wood	starring	loretta king hadler	4
		writer	alex gordon writer producer	writer	ed wood	2
		starring	tor johnson	writer	alex gordon writer producer	2
		starring	loretta king hadler	starring	tor johnson	2
		starring	bela lugosi	starring	bela lugosi	2
who produced the most films	Hal Roach	spouse	marguerite nichols	spouse	marguerite nichols	2
		child	hal roach jr	child	hal roach jr	2
		birth place	elmira new york	birth place	elmira new york	2
		resting place	woodlawn cemetery elmira new york	resting place	woodlawn cemetery elmira new york	1
		occupation	film producer	occupation	film producer	1
bond girls	Ursula Andress	title	bond girl	title	bond girl	3
		place of birth	switzerland	state of origin	swiss people	3
		place of birth	ostermundigen	place of birth	switzerland	3
		birth place	switzerland	place of birth	ostermundigen	3
		spouse	john derek	birth place	switzerland	3
most beautiful railway stations world cities located	Parappanangadi	subdivision name	kerala	is part of	malappuram	3
		is part of	kerala	country	india	3
		country	india	is part of	kerala	2
		time zone	indian standard time	time zone	indian standard time	0
		postal code type	postal index number	subdivision type	states and union territories of india	0

Table 5.4: Some example queries (along with the corresponding subject entities) and the resulting top-5 utility rankings of predicted and ground truth facts from a held-out dataset. The final column provides the ground truth utility score.

CHAPTER 6
ENTITY DESCRIPTION GENERATION

6.1 Overview

A substantial percentage of online search requests involve named entities such as people, locations, businesses, etc. Search engines and digital personal assistants (e.g., Google Assistant, Alexa, Cortana, Siri) alike now extensively draw on *knowledge graphs* as vast databases of entities and their properties.

Most large cross-domain factual knowledge graphs, such as Wikidata [5], the Google Knowledge Graph, and YAGO [4] include short textual descriptions of entities. These can be provided in response to entity queries (such as *Where is Fogo Island?*), e.g., as an informative direct answer or to enrich a knowledge panel given by a Web search engine such as Google, as shown in Figure 6.1 for Fogo Island. Such textual descriptions effectively provide for a near-instantaneous human understanding of an entity. They can also be helpful in a number of linguistic tasks, including named entity disambiguation, and can serve as fine-grained ontological types in question answering and reasoning-driven applications.

Despite their eminent importance for information retrieval and other applications, these descriptions are only sparsely available, typically for the more well-known entities, leaving large numbers of entities with no such descriptions. For example, in Wikidata, which is used by Apple's Siri, as many as 6.8 million entities do not have any description in any language as of March, 2021.[1]

Moreover, new entities are added frequently to the existing knowledge graphs. For example, the number of entities in Wikidata almost doubled in the last couple of years.

Given the large number of entities lacking textual descriptions, a promising solution is

[1] https://tools.wmflabs.org/wikidata-todo/stats.php

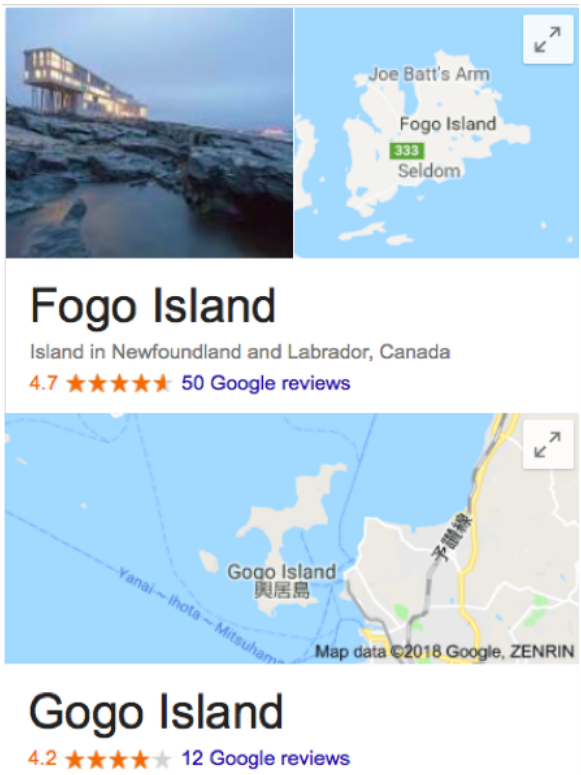


Figure 6.1: An example of a missing description in Google's Knowledge Graph. Similar to *Fogo Island*, a synoptic description for *Gogo Island* could be *Island in Ehime, Japan*.

to fully automatically generate synoptic textual descriptions from the existing factual data. Ideally, this automatic generation process should be able to synthesize very concise descriptions, retaining only a very small number of particularly discernible pieces of information about an entity. For example, in Wikidata and Google's Knowledge Graph, *occupation* and *nationality* are among the preferred attributes invoked in describing a person (e.g., *Magnus Carlsen* as a *Norwegian chess Grandmaster*). At the same time, however, an *open-domain* solution is needed, in order to cope with the various different kinds of entities users may search for in a knowledge graph, and the various kinds of names that might appear in such descriptions. For instance, for Gogo Island in Figure 6.1, an appropriate description would perhaps refer to *Ehime, Japan* as the prefecture in which the island is located. Moreover, apart from people and locations, users could likewise also search for paintings, films, or any other types of entities. Additionally, the generated descriptions should also be precise,

coherent, and non-redundant.

In this chapter, we present two novel neural models for generating succinct textual descriptions from knowledge triples.

- A dynamic memory-based generative network that can generate short textual descriptions from the available factual information about the entities [133].

- A fact-to-sequence encoder–decoder model with explicit copy mechanism to copy fact-specific tokens directly to the output description [109]. Since many names and words in the desired output descriptions are also present in the underlying facts, it is an intuitive design choice to selectively copy words from the facts to the description. The copy mechanism is important, as it is difficult to predict rare or unseen words based on mere statistical co-occurrences of words.

These approaches are substantially different from previous work on this topic. In contrast to some related works that focus on generating Wikipedia-style summary descriptions from factual data [134, 135, 136], or textual verbalization of RDF triples [137, 138, 139], these models focus on synthesizing quickly graspable synoptic textual descriptions from factual knowledge. In most of the cases, these are single-phrase descriptions as compared to multi-sentence Wikipedia-style summaries. Previous work has suggested generating short descriptions using pre-defined templates. However, this approach severely restricts the expressivity of the model and hence such templates are typically only applied to very narrow classes of entities. In contrast, our goal is to design a broad-coverage open domain description generation architecture. Note that we restrict our work only to English description generation, as English is the most prevalent language in which descriptions are available in knowledge graphs. Typically, the first few sentences of Wikipedia articles contain a detailed description of the entity, and one can attempt to generate brief textual descriptions by learning to reproduce the first few sentences based on the remaining article text. However, our goal is to make our method open-domain, i.e., applicable to any type

of entity with factual information that may not have a corresponding Wikipedia-like article available. Indeed, Wikidata currently has more than 90 million items, whereas the English Wikipedia has only 6.5 million articles. Hence, for the vast majority of items in Wikidata, no corresponding Wikipedia article is available. In such cases, a summarization will not be effective.

6.1.1 Preliminaries

Property Each subject entity $s \in \mathcal{E}$ has a number of properties $P \in \mathcal{P}$ associated with it. Each property name is described as a sequence of tokens $P = (w_1^p, w_2^p, ..., w_{|P|}^p)$. Note that these properties are predefined in a schema-based knowledge graph.

Property Value Each property P of entity s has a corresponding value O_P, which is also a sequence of tokens $O_P = (w_1^o, w_2^o, ..., w_{|O_P|}^o)$. The property values could be another entity $e \in \mathcal{E}$, a date–time value, a string, or a numeric value, etc.

Fact Each of the property–value pairs is deemed a *fact*. Each subject entity $s \in \mathcal{E}$ has a number of different facts $\mathcal{F} = \{f_1, f_2, \ldots f_N\}$ that characterize it.

Description A short textual description of entity s is a sequence of tokens $\mathcal{D} = (w_1, w_2, \ldots, w_n)$. Each word in the description D can be a *factual word*, or a *vocabulary word*.

Factual Words The factual words for each fact $f \in \mathcal{F}$, denoted as $\mathcal{V}_f$, are the words appearing in the property value. Stop words (e.g., *for*, *of*, *in* etc.) are excluded from the factual words vocabulary. Note that the sets of factual words are instance-specific, i.e., $\bigcup_f \mathcal{V}_f$ is different for different entities $s \in \mathcal{E}$.

Vocabulary Words The vocabulary words, denoted as $\mathcal{V}$, could be a list of frequent words in the English dictionary. For our experiments, we rely on the 1,000 most frequent words appearing in our corpus as vocabulary words. Typically, the words that appear in

the property names constitute this vocabulary. This is aligned with the expectation for a schema-based knowledge graph with fixed number of properties. Note that $\mathcal{V}_f \cap \mathcal{V} \neq \emptyset$, which indicates that there could be words in the property value that are also vocabulary words.

6.2 A Dynamic Memory-based Generative Network Architecture

The dynamic memory-based generative network consists of three key components: an input module, a dynamic memory module, and an output module. A schematic diagram of these are given in Figure 6.2.

6.2.1 Input Module

The input to the input module is a set of N facts $\mathcal{F} = \{f_1, f_2, \ldots, f_N\}$ pertaining to an entity. Each of these input facts are essentially (s, p, o) triples, for subjects s, predicates p, and objects o. Upon being encoded into a distributed vector representation, we refer to them as *fact embeddings*.

Although many different encoding schemes can be adopted to obtain such fact embeddings, we opt for a positional encoding as described by [140], motivated in part by the considerations given by [141]. For completeness, we describe the positional encoding scheme here.

We encode each fact f_i as a vector $\mathbf{f_i} = \sum_{j=1}^{J} \mathbf{l_j} \circ \mathbf{w_j^i}$, where $\circ$ is an element-wise multiplication, and $\mathbf{l_j}$ is a column vector with the structure $l_{kj} = (1 - \frac{j}{J}) - (k/d)(1 - 2\frac{j}{J})$, with J being the number of words in the factual phrase, $\mathbf{w_j^i}$ as the embedding of the j-th word, and d as the dimensionality of the embedding.

Thus, the output of this module is a concatenation of N fact embeddings $\mathbf{F} = [\mathbf{f_1}; \mathbf{f_2}; \ldots; \mathbf{f_N}]$.

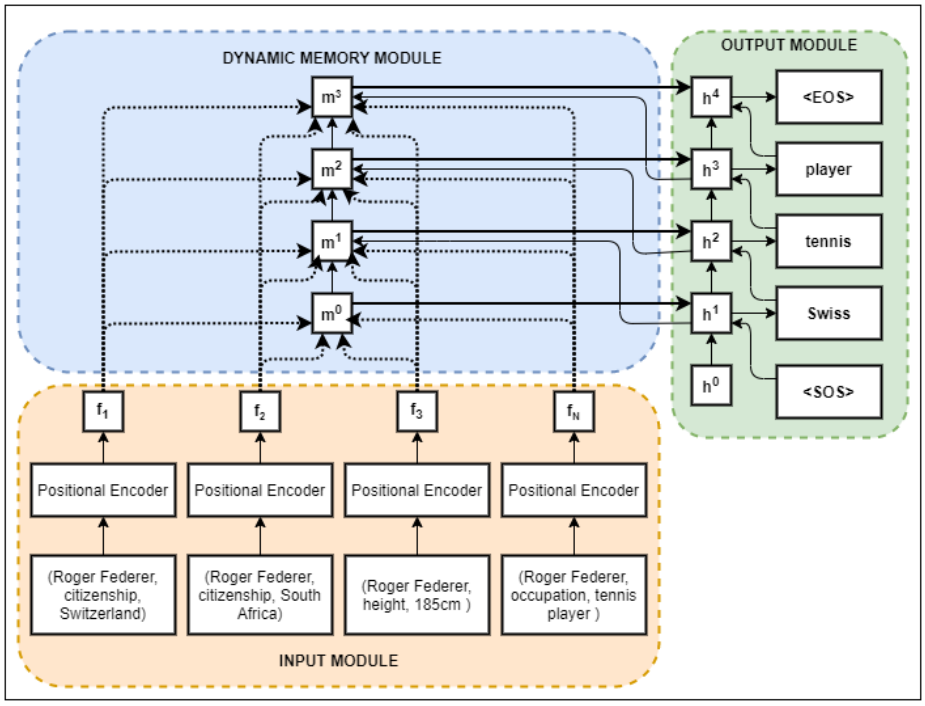


Figure 6.2: Dynamic memory-based generative network architecture.

6.2.2 Dynamic Memory Module

The dynamic memory module is responsible for memorizing specific facts about an entity that will be useful for generating the next word in the output description sequence. Intuitively, such a memory should be able to update itself dynamically by accounting not only for the fact embeddings but also for the current context of the generated sequence of words.

To begin with, the memory is initialized as $\mathbf{m}^{(0)} = \max(\mathbf{0}, \mathbf{W}_\mathbf{m}\mathbf{F} + \mathbf{b}_\mathbf{m})$. At each time step t, the memory module attempts to gather pertinent contextual information by attending to and summing over the fact embeddings in a weighted manner. These attention weights are scalar values informed by two factors: (1) how much information from a particular fact is used by the previous memory state $\mathbf{m}^{(t-1)}$, and (2) how much information of a particular

fact is invoked in the current context of the output sequence $\mathbf{h}^{(t-1)}$. Formally,

$$\mathbf{x_i}^{(t)} = [|\mathbf{f_i} - \mathbf{h}^{(t-1)}|; |\mathbf{f_i} - \mathbf{m}^{(t-1)}|], \tag{6.1}$$

$$\mathbf{z_i}^{(t)} = \mathbf{W_2} \tanh(\mathbf{W_1}\mathbf{x_i}^{(t)} + \mathbf{b_1}) + \mathbf{b_2}, \tag{6.2}$$

$$a_i^{(t)} = \frac{\exp(\mathbf{z_i}^{(t)})}{\sum_{k=1}^{N} \exp(\mathbf{z_k}^{(t)})}, \tag{6.3}$$

where $|.|$ is the element-wise absolute difference and $[;]$ denotes the concatenation of vectors.

Having obtained the attention weights, we apply a soft attention mechanism to extract the current context vector at time t as

$$\mathbf{c}^{(t)} = \sum_{i=1}^{N} a_i^{(t)} \mathbf{f_i}. \tag{6.4}$$

This newly obtained context information is then used along with the previous memory state to update the memory state as follows:

$$\mathbf{C}^{(t)} = [\mathbf{m}^{(t-1)}; \mathbf{c}^{(t)}; \mathbf{h}^{(t-1)}] \tag{6.5}$$

$$\mathbf{m}^{(t)} = \max(\mathbf{0}, \mathbf{W_m}\mathbf{C}^{(t)} + \mathbf{b_m}) \tag{6.6}$$

Such updated memory states serve as the input to the decoder sequence of the output module at each time step.

6.2.3 Output Module

The output module governs the process of repeatedly decoding the current memory state so as to emit the next word in an ordered sequence of output words. We rely on GRUs for this.

At each time step, the decoder GRU is presented as input the current memory state $\mathbf{m}^{(t)}$ as well as the previous context of the output sequence, i.e., the previous hidden state

of the decoder $\mathbf{h}^{(t-1)}$. At each step, the resulting output of the GRU is concatenated with the context vector $\mathbf{c}^{(t)}$ and is passed through a fully connected layer and finally through a softmax layer. During training, we deploy *teacher forcing* at each step by providing the vector embedding of the previous correct word in the sequence as an additional input. During testing, when such a signal is not available, we use the embedding of the predicted word in the previous step as an additional input to the current step. Formally,

$$\mathbf{h}^{(t)} = \text{GRU}([\mathbf{m}^{(t)}; \mathbf{w}^{(t-1)}], \mathbf{h}^{(t-1)}), \tag{6.7}$$

$$\tilde{\mathbf{h}}^{(t)} = \tanh(\mathbf{W_d}[\mathbf{h}^{(t)}; \mathbf{c}^{(t)}] + \mathbf{b_d}), \tag{6.8}$$

$$\hat{\mathbf{y}}^{(t)} = \text{Softmax}(\mathbf{W_o}\tilde{\mathbf{h}}^{(t)} + \mathbf{b_o}), \tag{6.9}$$

where $[;]$ is the concatenation operator, $\mathbf{w}^{(t-1)}$ is vector embedding of the previous word in the sequence, and $\hat{\mathbf{y}}^{(t)}$ is the probability distribution for the predicted word over the vocabulary at the current step.

6.2.4 Loss Function and Training

Training this model amounts optimizing the model parameters θ, which include the matrices $\mathbf{W_1}$, $\mathbf{W_2}$, $\mathbf{W_m}$, $\mathbf{W_d}$, $\mathbf{W_o}$ and the corresponding bias terms $\mathbf{b_1}$, $\mathbf{b_2}$, $\mathbf{b_m}$, $\mathbf{b_d}$, and $\mathbf{b_o}$ as well as the various transition and output matrices of the GRU.

To this end, if each of the training instances has a description with a maximum of M words, we can rely on the categorical cross-entropy over the entire output sequence as the loss function:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{M}\sum_{j=1}^{|\mathcal{V}|} y_j^{(t)} \log(\hat{y}_j^{(t)}). \tag{6.10}$$

where $y_j^{(t)} \in \{0, 1\}$ and $|\mathcal{V}|$ is the vocabulary size.

We train our model end-to-end using Adam as the optimization technique with a learning rate of 1e-03.

6.3 Fact-to-Sequence Encoder-Decoder Model

In order to enable concise and precise open-domain entity descriptions, we propose a generative model consisting of a fact-to-sequence encoder–decoder architecture, aided by a pointer network based copy mechanism. Our model comprises a positional fact encoder, a GRU-based sequence decoder, and a copy function to directly transfer factual words to the generated descriptions.

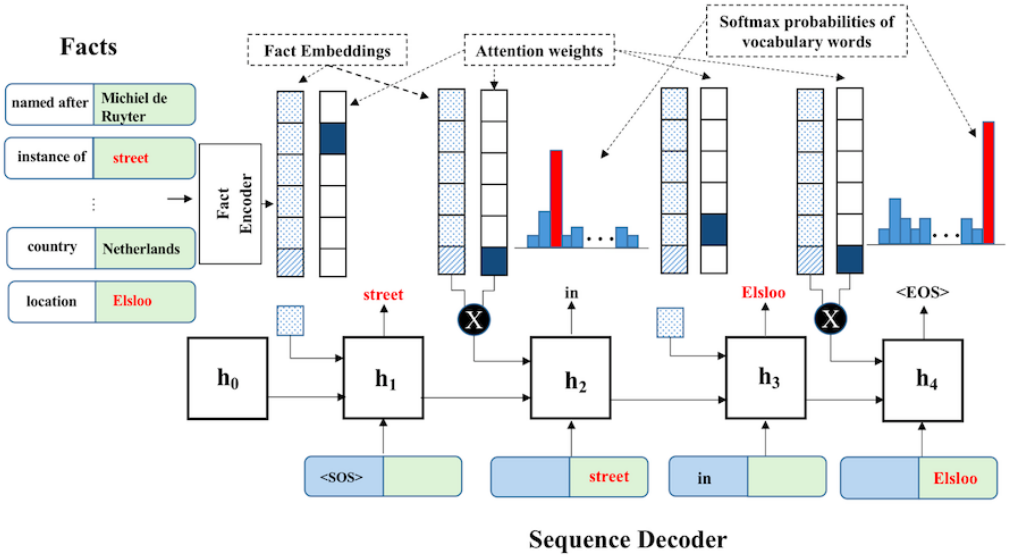


Figure 6.3: Model architecture. For Wikidata item Q19345316 (*Michiel de Ruyterstraat*), factual words such as *street* and *Elsloo* are directly copied from the underlying facts *(Instance of, street)* and *(location, Elsloo)*, respectively, while the general vocabulary words *in* and <EOS> are selected by a softmax classifier.

6.3.1 Fact Encoder

The fact encoder transforms a set of input facts $\mathcal{F} = \{f_1, f_2, \ldots f_N\}$ into *fact embeddings* using the positional encoding technique described in Subsection 6.2.1 If the word w_i^j is not a vocabulary word, we replace it with the special token <UNK>. This arrangement deals with the rare or unseen words that appear in the factual phrase f_i, which is a concatenation of words in the property name and property value.

We append the fact embeddings with a *mean fact* – a special fact that is responsible

for generating vocabulary words. It is derived as the element-wise mean of the fact embeddings, i.e., $\mathbf{f}_{N+1} = \frac{1}{N}\sum \mathbf{f}_i$. We train the model to attend to the mean fact if it has to generate a vocabulary word in the output sequence.

6.3.2 Output Sequence Decoder

At every time step t, the sequence decoder selects a fact. This selection governs the generation of the next word either from the corresponding factual words, or from the vocabulary words.

Fact Selection Given the set of fact embeddings $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{N+1}\}$ as input, the decoder selects a fact at each time step t using an attention mechanism. The attention scores are calculated as a likelihood distribution using the hidden state of the decoder in the previous time step and the fact embeddings. Formally,

$$\mathbf{e}_i = \mathbf{W}_2 \tanh(\mathbf{W}_1[\mathbf{f}_i; \mathbf{h}_{t-1}]) \;\; \forall i \in \{1, N+1\} \tag{6.11}$$

$$P(f = f_i \mid \mathbf{f}_i, \mathbf{h}_{t-1}) = \frac{\exp(\mathbf{e}_i)}{\sum_{i' \in \{1,N+1\}} \exp(\mathbf{e}_{i'})} \tag{6.12}$$

where $\mathbf{e}_i$ denotes the attention energy of the i-th fact, $[;]$ denotes the concatenation operation, and $\mathbf{W}_1 \in \mathrm{R}^{m \times 2d}$, $\mathbf{W}_2 \in \mathrm{R}^m$ are affine transformations of a 2-layer feed-forward network. We select the fact with maximum attention score at time step t, denoted as f_t and its corresponding fact embedding $\mathbf{f}_t$ as the following.

$$f_t = \underset{i \in \{1,\ldots,N+1\}}{\arg\max} \; P(f = f_i \mid \mathbf{f}_i, \mathbf{h}_{t-1}) \tag{6.13}$$

$$\mathbf{f}_t = \mathbf{f}_{f_t} \tag{6.14}$$

GRU Decoder We rely on Gated Recurrent Units (GRU), which is a Recurrent Neural Network (RNN) variant, as our preferred decoder. At each time step t, the decoder input is a concatenation of three vectors: the embedding $\mathbf{f}_t$ of a selected fact in the current time

step, the embedding $\mathbf{w}_{t-1}$ of the vocabulary word at the previous time step, and a one-hot vector $\mathbf{v}_{t-1}$ corresponding to the position of the copied factual word in the previous time step. Note that since the generated word in the previous time step can either be a vocabulary word or a factual word, either $\mathbf{w}_{t-1}$ or $\mathbf{v}_{t-1}$ is set to a zero vector.

The input representation $\mathbf{x}_t = [\mathbf{f}_t; \mathbf{w}_{t-1}; \mathbf{v}_{t-1}]$ is fed to the GRU decoder to update the states.

$$\mathbf{h}_t = \text{GRU}(\mathbf{x}_t, \mathbf{h}_{t-1}) \tag{6.15}$$

Generating Vocabulary Words If the attention mechanism assigns the maximum score to the mean fact f_{N+1}, then the decoder generates a vocabulary word $w_t \in \mathcal{V}$. To generate the vocabulary word, we use the attention-weighted context $\mathbf{c}_t = \sum_i \alpha_i \mathbf{f}_i$ and the GRU output state $\mathbf{h}_t$. A concatenation of these two vectors are fed to a 2-layer feed-forward network with a non-linear ReLU activation applied to the hidden layer. Formally,

$$\mathbf{o}_t = \mathbf{W}_a \text{ReLU}(\mathbf{W}_b[\mathbf{c}_t; \mathbf{h}_t]), \tag{6.16}$$

where $\mathbf{W}_a$ and $\mathbf{W}_b$ are affine transformations and $\text{ReLU}(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x})$. Finally, a probability distribution over the vocabulary words is obtained by a softmax function and the word with the maximum probability is emitted by the decoder. Formally,

$$P(w \mid \mathbf{c}_t, \mathbf{h}_t) = \text{Softmax}(\mathbf{o}_t) \tag{6.17}$$

$$w_t = \arg\max_{w \in \mathcal{V}} P(w \mid \mathbf{c}_t, \mathbf{h}_t) \tag{6.18}$$

Copying Factual Words The decoder copies factual words directly to the output when the fact selection process selects one of the N facts. To facilitate the copy mechanism, the decoder must select the position index of the factual word within the factual phrase. The position index is predicted by a 2-layer feed-forward network that takes a concatenation of the selected fact embedding $\mathbf{f}_t$ and the output state of the GRU $\mathbf{h}_t$ as input. Then, the

position index of the word to copy is determined as follows.

$$\mathbf{r}_t = \mathbf{W}_c \mathrm{ReLU}(\mathbf{W}_d[\mathbf{f}_t; \mathbf{h}_t]) \tag{6.19}$$

$$P(n \mid \mathbf{f}_t, \mathbf{h}_t) = \mathrm{Softmax}(\mathbf{r}_t) \tag{6.20}$$

$$n_t = \underset{n \in \{1,\ldots,|\mathcal{V}_{f_t}|\}}{\arg\max} P(n \mid \mathbf{f}_t, \mathbf{h}_t) \tag{6.21}$$

$$w_t = (\mathcal{V}_{f_t})_{n_t} \tag{6.22}$$

Here, n_t is the position index of the factual word to copy and $\mathcal{V}_{f_t}$ is the sequence of factual words corresponding to fact f_t and $(\mathcal{V}_{f_t})_i$ denotes the i-th item in that sequence.

6.3.3 Training

Our method requires selective copying of factual words to generate a description. In order to obviate the need for ground truth alignments of output description words with facts for training, we introduce an automated method of annotating each token in the description so as to align it to a matching fact. Specifically, we rely on a greedy string matching algorithm as detailed in Algorithm 1 for this purpose. If a token is not aligned with any of the facts, it is annotated as a vocabulary word. However, if the token is neither present in the vocabulary word set nor in the factual word set, it is assigned the special token `<UNK>` to denote that it is neither a factual word nor a vocabulary word. The symbol NaF in Algorithm 1 indicates that such words are not aligned to any fact. Since in our implementation, we consider the mean fact as a source of vocabulary words, we align these words to the mean fact.

Note that the greedy string matching algorithm used for fact alignment is a heuristic process that can be noisy. If a token in the output description appears in more than one fact, the string matching algorithm greedily aligns the token to the first fact it encounters, even if it is not the most relevant one. However, our manual inspection suggests that in most of the cases the alignment is relevant, justifying our choice of such a greedy approach.

Given the input facts $\mathcal{F}$ and the fact-aligned description $\tilde{\mathcal{D}}$, the model maximizes the

Algorithm 1: Algorithm for fact alignment of description

Data: Input: Set of facts $\mathcal{F}$, Description $\mathcal{D} = (w_1, w_2, \ldots, w_n)$
Result: Ordered sequence of fact aligned description $\tilde{\mathcal{D}}$

for $t \in \{1, ..., n\}$ **do**
  factual_flag $\leftarrow$ False
  for $f \in \mathcal{F}$ **do**
    if $w_t \in \mathcal{V}_f$ and factual_flag $=$ False **then**
      $\tilde{d}_t \leftarrow (w_t, f)$
      factual_flag $\leftarrow$ True
    end
  end
  if factual_flag $=$ False **then**
    if $w_t \in \mathcal{V}$ **then**
      $\tilde{d}_t \leftarrow (w_t, \text{NaF})$
    else
      $\tilde{d}_t \leftarrow (\text{UNK}, \text{NaF})$
    end
  end
end
$\tilde{\mathcal{D}} = (\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_n)$

log-likelihood of the observed words in the description with respect to the model parameters θ,

$$\theta^* = \arg\max_{\theta} \log P(\tilde{\mathcal{D}} \mid \mathcal{F}), \tag{6.23}$$

which can be further decomposed as

$$\log P(\tilde{\mathcal{D}} \mid \mathcal{F}) = \sum_{t=1}^{|\tilde{\mathcal{D}}|} \log P(w_t \mid w_{1:t-1}, \mathcal{F}). \tag{6.24}$$

Since the log-likelihood of the word w_t also depends on the underlying fact selection, we can further decompose $P(w_t)$ as

$$P(w_t) = P(w_t \mid f_t, w_{1:t-1})\, P(f_t \mid w_{1:t-1}) \tag{6.25}$$

Therefore, we train our model end-to-end by optimizing the following objective function:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{|\mathcal{D}|} \log P(w_t \mid f_t, w_{1:t-1}) - \sum_{t=1}^{|\mathcal{D}|} \log P(f_t \mid w_{1:t-1}). \tag{6.26}$$

Note that the alignment $\tilde{\mathcal{D}}$ of the description $\mathcal{D}$ to the facts $\mathcal{F}$ provides a ground truth fact f_t as each time step t during training.

6.4 Evaluation

6.4.1 Baselines

We compare the performance of our proposed models against a number of competitive baselines.

Fact-to-Sequence with Attention The comparison with this baseline method can be deemed as an ablation study, in which the decoder has no access to the copy mechanism. This baseline resembles the standard sequence-to-sequence with attention mechanism proposed by Bahdanau et al. [65]. However, unlike in their work, the input to this model consists of positionally encoded discrete facts.

Each word in the output sequence is predicted by providing the attention-weighted fact embeddings and the previous hidden state as an input to the GRU decoder. Then, the output of the decoder is concatenated with the attention-weighted fact embeddings and passed through a 2-layer feed-forward network with a ReLU activation. Finally, the output of the MLP is fed to a softmax classifier that outputs a probability distribution for the combined vocabulary of factual and non-factual words $\mathcal{V} \cup \{\bigcup_{s \in \mathcal{E}} \mathcal{V}_f^s\}$. The word with the maximum probability is emitted by the decoder.

Neural Knowledge Language Model (NKLM) We compare against Ahn et al.'s Neural Knowledge Language Model [135], which is able to generate Wikipedia-style multi-

sentence summary paragraphs for movie actors. Although the descriptions in our case are much shorter, we adopted their model as a representative baseline for methods yielding multi-sentence summaries, as the tasks are similar in nature. NKLM also adopts a copy mechanism but the decision about whether to copy is made by a binary gating variable that is provided as an additional label by augmenting the dataset during training. By predicting whether the word to generate has an underlying fact or not, the model can generate such words by copying from the selected fact. On the contrary, our model decides to copy whenever a fact other than the mean fact is selected. We implemented and trained their model with the benchmark datasets. Their model also requires an alignment of descriptions to the underlying facts. Additionally, NKLM relies on pre-trained TransE [20] embeddings of objects and relations to obtain the fact embeddings that are provided as inputs to the model.

6.4.2 Datasets

WikiFacts10k-OpenDomain		WikiFacts10k-Imbalanced	
Instance of	Pct.	Instance of	Pct.
human	11.45%	human	69.17%
painting	8.73%	painting	2.02%
commune of france	6.41%	commune of france	1.43%
film	6.27%	scientific article	1.36%
scientific article	5.56%	film	1.34%
encyclopedic article	3.30%	encyclopedic article	0.89%
asteroid	2.89%	asteroid	0.62%
taxon	2.58%	taxon	0.57%
album	2.19%	road	0.52%
road	2.02%	album	0.46%

Table 6.1: Frequency distribution of the top-10 domains in the two datasets.

We curate two benchmark datasets, each of which consists of 10K entities with at least 5 facts and an English description. The first of these datasets, denoted as *WikiData10K-Imbalanced*. Since the entities in that dataset were randomly sampled from the Wikidata RDF dump, the ontological types of the sampled entities have a long-tail distribution, while an overwhelming 69.17% of entities are instances of *human*. This highly skewed distribu-

tion makes the dataset biased towards a particular domain. To decrease the percentage of such entities in the dataset, we create another dataset, *WikiData10K-OpenDomain*, in which the instances of *human* are downsampled to 11.45% to accommodate more instances of other ontological types as evinced by the frequency distribution in Table 6.1. These datasets, both available online[2], are split into training, dev., and test sets in a 80:10:10 ratio.

6.4.3 Metrics

Following previous work, we use the automatic evaluation metrics BLEU (1 - 4) [142], ROUGE-L [143], METEOR [144], and CIDEr [145] for a quantitative evaluation of the generated descriptions with respect to the ground truth descriptions provided in the benchmark data. These metrics are widely used in the literature for the evaluation of machine translation, text summarization, and image captioning. BLEU is a precision-focused metric, ROUGE-L is a recall-based metric, METEOR uses both precision and recall with more weight on recall than precision, and CIDEr considers TF-IDF-weighted n-gram similarities. Following standard practice, as in the official implementation[3], the raw scores for CIDEr are multiplied by a factor of 10 for better readability, yielding values in $[0, 10]$. For the same reason, following standard practice, the scores for other metrics are multiplied by 100, yielding scores in the range of $[0, 100]$. Note that typically these metrics rely on more than one human-written ground truth output per instance for evaluation. The lack of any alternative descriptions in the benchmark datasets implies that the generated descriptions are each evaluated on a single ground truth description. Additionally, these metrics take a very conservative approach in that they look for overlapping words, word alignment, longest common subsequence, etc.

[2]https://github.com/kingsaint/Wikidata-Descriptions
[3]https://github.com/vrama91/cider

6.4.4 Experimental Setup

Our models and all other baselines are trained for a maximum 25 epochs. We report the results of the best performing models on the dev set. For NKLM, we use the default hyper-parameter settings used by the authors, and we obtain the fact embeddings by concatenating the embeddings of object entity and relation that are obtained by using the TransE [20] embedding model. For our models and the fact-to-sequence with attention baseline, we fix the embedding dimensions of facts and words to 100. The hidden layers of the GRU and the two-layer feed-forward networks are 100-dimensional. We use Adam as our optimizer with a fixed learning rate of 0.001. We fix the maximum number of facts to 60 including the mean fact. For instances with less that 60 facts, we resort to a suitable masking to obtain the attention scores of the relevant facts. The maximum number of factual words for each fact is limited to 60.

We use the $1,000$ most frequent words in the dataset as our default vocabulary. One can also consider using a much larger vocabulary of frequent English words. However, for our experiments we found it unnecessary. Still, because of this restricted vocabulary, our model may occasionally sparsely generate `<UNK>` tokens, which we remove from the generated description.

The datasets and the PyTorch implementations of all our experiments are available online.[4]

6.4.5 Results

Table 6.2 and Table 6.3 provide the evaluation results of our model and the baselines on the WikiFacts10k-Imbalanced and WikiFacts10k-OpenDomain datasets, respectively. Our fact-to-seq encoder–decoder model with copy mechanism outperforms the dynamic memory-based generative model by more than 8 BLEU-4 points on the imbalanced dataset. We also observe 1 to 7 point gains on the scores across all other metrics. Similar trends are

[4] https://github.com/kingsaint/Wikidata-Descriptions

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR	CIDEr
Fact2Seq w. attention	**62.8▲**	56.3	53.0	52.7	63.0	35.0	3.321
NKLM	34.7	27.7	29.1	29.0	44.1	20.1	1.949
Dynamic Memory-based Generative Model	61.1	53.5	48.5	46.1	64.1	35.3	3.295
Fact2Seq w. copy	61.9	**57.3▲**	**55.6▲**	**54.3▲**	**64.9▲**	**36.3▲**	**3.420▲**
– without Positional Encoder	58.9	51.9	46.3	42.5	63.4	33.7	3.126
– without Mean Fact	58.0	52.2	49.5	48.6	64.3	34.8	3.139
– Copy Only	26.8	21.2	16.5	11.8	45.3	20.6	1.766

Table 6.2: Experimental results for the WikiFacts10K-Imbalanced benchmark dataset. The best results are shown in bold face and the second best results are underlined. ▲ indicates the best results are statistically significant w.r.t the second best results. We use approximate randomization (aka. permutation test) as the method for statistical significance test.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR	CIDEr
Fact2Seq w. attention	62.7	56.3	50.0	46.2	67.7	35.8	3.629
NKLM	47.9	42.1	37.5	32.3	57.4	25.9	2.958
Dynamic Memory-based Generative Model	57.8	50.7	43.6	39.7	67.6	34.9	3.556
Fact2Seq w. copy	**68.2▲**	**61.8▲**	**56.6▲**	**51.9▲**	**70.0▲**	**37.3▲**	**4.084▲**
– without Positional Encoder	66.2	58.4	51.8	45.9	67.9	34.7	3.717
– without Mean Fact	62.4	58.0	55.1	51.8	67.7	36.0	3.856
– Copy Only	37.5	26.4	18.4	10.9	55.2	24.8	2.136

Table 6.3: Experimental results on the WikiFacts10K-OpenDomain benchmark dataset. The best results are shown in bold face and the second best results are underlined. ▲ indicates the best results are statistically significant w.r.t the second best results. We use approximate randomization (aka. permutation test) as the method for statistical significance test.

observed for the more challenging WikiFacts10k-OpenDomain dataset, in which fact-to-seq encoder–decoder model with copy mechanism improves upon the dynamic memory-based generative model by 12.2 points in terms of the BLEU-4 metric. For this dataset, we also observe 5 to 8 point gains across all other metrics. A substantial gain in the BLEU scores indicates that the fact-to-seq encoder–decoder model with copy mechanism can consistently generate longer sequences more accurately that the other models.

Furthermore, we observe that our attention-based fact-to-sequence model also outperforms the dynamic memory-based generative model in most of the metrics across both the datasets. This observation shows that one can do without a dynamic memory module for memorizing fact specific information. It turns out that a GRU-based architecture with an intelligently chosen encoding of the input and an extra skip-connection in the decoder has sufficient memory to retain fact-specific information required to generate words in the output.

The NKLM model by Ahn et al. obtains sub-par results on both datasets, particularly the WikiFacts10k-Imbalanced data, although the original paper focused on descriptions of humans. This shows that their method for generating a short text is unsuitable for our demanding task of generating very concise entity descriptions. Additionally, the pre-trained TransE embeddings required to obtain fact embeddings are not available for previously unseen entities in the test set, thus severely restricting the model's ability to cope with unseen entities.

6.4.6 Ablation Study

In addition to comparing the fact-to-seq encoder–decoder model with copy mechanism against a number of competitive baselines, we perform a series of ablation studies to evaluate the effect of different components within our model. The results of these experiments are included in Table 6.2 and Table 6.3.

Without Positional Encoder To understand the effect of the positional encoder, we replace the positional encoder with the average pooling of the constituent word embeddings of the facts to obtain the fact embeddings. The differences between the BLEU scores of our model and this ablated version indicates that the position of the words in the factual phrases indeed plays an important role in encoding a fact that provides a useful signal to the decoder.

Without Mean Fact To understand the effect of mean fact in our model, we replace the mean fact with a fixed embedding vector of dimension d. The elements of this vector are sampled from a uniform distribution $\mathcal{U}(-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}})$ and do not change during training. The result of this experiment suggests that the role of the mean fact in our model is not of just that of a *sentinel* responsible for generating general vocabulary words. Rather, it encodes some useful semantic information that the model can exploit.

Copy Only This experiment is an ablation study that we perform to understand how precisely a model can generate descriptions if it only copies factual words. This baseline is a restricted version of our model, in which the model never generates any general vocabulary word. The much inferior performance of this model shows that merely copying factual words to the output does not yield precise descriptions. This further demonstrates the importance of a model that dynamically decides whether to copy words from the input facts or instead emit general vocabulary words.

6.4.7 Parameter Efficiency

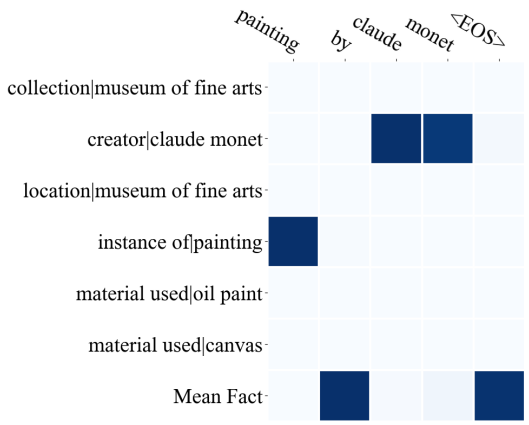
Model	#Parameters
Dynamic memory-based generative model	14,197,741
Fact2Seq w. attention	14,159,561
Neural Knowledge Language Model (NKLM)	20,569,361
Fact2Seq w. copy	979,986

Table 6.4: Comparison of the number of learnable parameters.

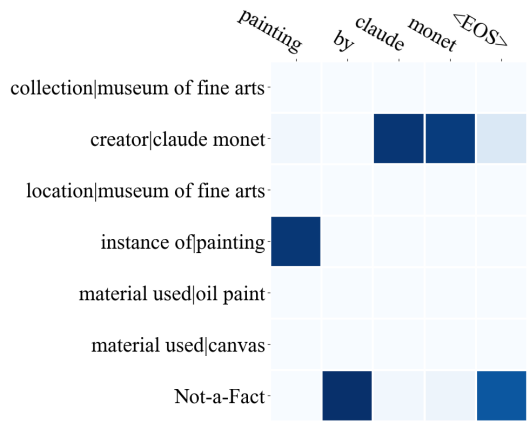
Table 6.4 shows the number of learnable parameters for our model as well as the baselines. Our Fact2Seq with copy model is 14x more parameter-efficient than the competitive baselines. As there are fewer parameters to learn, this drastically improves the average training time of the model as compared to the other baselines. The number of parameters depends on the vocabulary size of the output softmax layer and the input to the word embeddings. Reducing the size of the softmax vocabulary to the most frequent words and making the model copy fact-specific words directly from the input contributes to the parameter efficiency and faster training of the model.

6.4.8 Importance of Fact Alignment of Descriptions

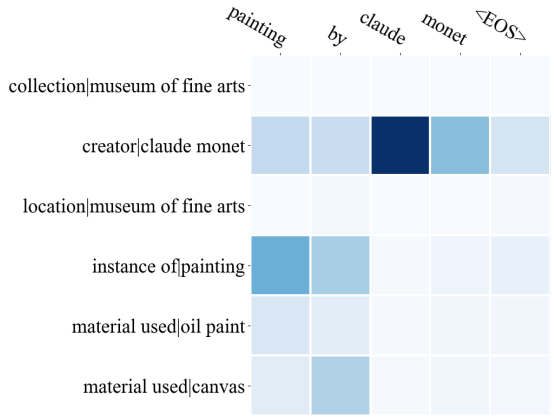
The alignment of facts to the description teaches the model to choose the right fact for generating a factual word. As shown in Figure 6.4, for each word of the generated description, our model precisely selects the relevant underlying fact. The NKLM also shows a simi-



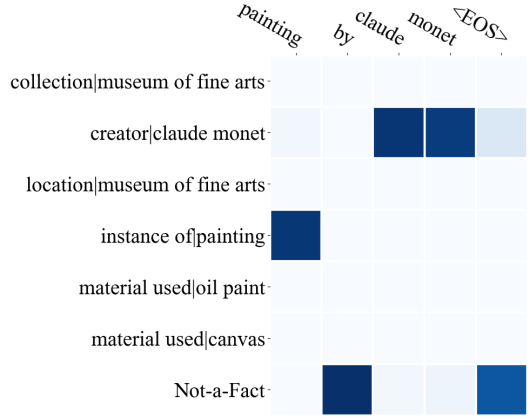
(a) Our Model



(b) NKLM



(c) Fact2Seq



(d) Dynamic Memory-based Generative Model

Figure 6.4: A visualization of attention weights for selecting relevant facts.

lar effect. However, the dynamic memory-based generative model does not always pick the most relevant fact, although it might occasionally generate the right word due to the statistical co-occurrence of the words in the training set.

6.4.9 Significance of the Copy Mechanism

The copy mechanism enables our model to copy fact-specific rare or previously unseen words directly to the output sequence, generating factually correct descriptions. To demonstrate the positive effect of this copy mechanism, we select instances from a subset of 8 ontological types that tend to require explicit copying of factual words. This subset accounts for 19.6% of the test set. Table 6.5 shows some examples from this subset. We also perform an automatic evaluation on this subset and provide the results in Table 6.7. Both the models with copy mechanism significantly outperform the baselines lacking any explicit copy mechanism. These results demonstrate that a suitable copy mechanism generates far more precise open-domain descriptions.

6.5 Related Work

In the following, we review previous research and describe how it differs from the task and approach we consider in this paper.

6.5.1 Text Generation from Structured Data

Lebret et al. [134] take Wikipedia infobox data as input and train a neural language model that, conditioned on occurrences of words in the input table, generates biographical sentences as output. In a similar vein, Ahn et al. [135] infused factual knowledge into an RNN-based language model to generate Wikipedia-style summary paragraphs of film actors. Similar to ours, their approach also uses a copy mechanism to copy specific words from the input facts to the description. However, these approaches are not directly compatible with our problem setting, which focuses on generating synoptic, rather than detailed

Instance Of	Ground Truth Description	Generated Description	
Album	album by hypocrisy	Dynamic Memory	album by michelangelo
		Fact2Seq	album by song
		NKLM	album by hypocrisy hypocrisy
		Our Model	album by hypocrisy
	album by canadian country music group family brown	Dynamic Memory	czech hong kong by rapper laurana
		Fact2Seq	sparta album
		NKLM	album by family brown
		Our Model	album by family brown
Book	science fiction novel by richard k morgan	Dynamic Memory	science fiction novel by
		Fact2Seq	episode of a book by bernard dyer
		NKLM	fiction by richard k
		Our Model	novel by richard k
Painting	painting by hendrick cornelisz van vliet	Dynamic Memory	painting by cornelis de vos
		Fact2Seq	painting by abraham van (ii) switzerland
		NKLM	painting by hendrick cornelisz
		Our Model	painting by hendrick cornelisz van vliet
	painting by eustache le sueur	Dynamic Memory	painting by thomas hovenden
		Fact2Seq	painting by california
		NKLM	painting by eustache le
		Our Model	painting by le sueur sueur
Road	highway in new york	Dynamic Memory	highway in new york
		Fact2Seq	highway in new york
		NKLM	¡UNK¿ highways york
		Our Model	highway in new york
	road in england	Dynamic Memory	area in the london borough of croydon
		Fact2Seq	of in london
		NKLM	road road in the church of england
		Our Model	road in the area london
Sculpture	sculpture by antoine coysevox	Dynamic Memory	sculpture by frederick william pomeroy
		Fact2Seq	sculpture by unknown singer
		NKLM	sculpture by antoine coysevox
		Our Model	artwork by antoine coysevox
	sculpture by donatello	Dynamic Memory	by henry and final by the carducci
		Fact2Seq	sculpture by statue kreis comedy
		NKLM	sculpture by donatello donatello nilo
		Our Model	sculpture by donatello
Single/ Song	1967 gilbert bcaud song	Dynamic Memory	2014 by 2012 of the czech 2014 by czech
		Fact2Seq	song
		NKLM	song by gilbert bcaud
		Our Model	song by gilbert bcaud
	single	Dynamic Memory	song by dutch by 2014 municipality
		Fact2Seq	1980 song by northern song
		NKLM	single by michael jackson
		Our Model	song by michael jackson
Street	street in boelenslaan	Dynamic Memory	street in richard
		Fact2Seq	street in collection
		NKLM	street in boelenslaan
		Our Model	street in achtkarspelen
	street in echt	Dynamic Memory	street in singer
		Fact2Seq	street in one
		NKLM	street in echt
		Our Model	street in echt

Table 6.5: Examples from the subset of ontological types that benefits from copy mechanism.

Item	Instance of	Generated Description
Q11584386	Human	japanese tarento
Q2198428	Human	netherlands businessperson
Q3260917	Human	french military personnel
Q1494733	Painting	painting by august macke
Q16054316	Painting	painting by liselotte schramm-heckmann
Q15880468	Painting	painting by emile wauters
Q10288648	Book	book by izomar camargo guilherme
Q10270545	Book	novel by antonin kratochvil
Q10272202	Book	novel by jose louzeiro
Q1001786	Street	street in budapest
Q10552208	Street	street in orebro
Q10570752	Street	street in malmo municipality

Table 6.6: Examples of generated descriptions for the Wikidata entities with missing descriptions as of December, 2018.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR	CIDEr
Dynamic Memory-based Generative Model	39.5	31.9	22.3	14.8	55.9	23.6	2.319
Fact2Seq w. Attention	52.8	44.5	33.5	24.0	56.9	24.1	2.337
Neural Knowledge Language Model	74.3	68.3	**63.1**	**57.2**	73.9	40.1	5.348
Our Model	**75.4**	**69.1**	62.8	55.5	**76.8**	**41.4**	**5.387**
- Copy Only	42.7	26.3	15.4	0.0	57.9	27.1	2.089

Table 6.7: Experimental results for the subset of ontological types that require explicit copying of factual words.

multi-sentence descriptions. Additionally, in contrast to our setting, which requires dynamically considering a wide variety of domains and entity types, the previous studies consider just human biographies as a single domain. Our experiments show that our method substantially outperforms the approach by Ahn et al. on our task.

The WebNLG Challenge [137] is another task aiming at generating text from RDF triples. However, it differs quite substantially from the task we study in this paper, as it demands a textual verbalization of every single triple. Our task, in contrast, requires synthesizing a short synoptic description by precisely selecting the most relevant and distinctive facts from the set of all available facts about the entity.

6.5.2 Referring Expression Generation

Referring Expression Generation (REG) is a subtask of Natural Language Generation (NLG) that focuses on the creation of noun phrases that identify specific entities. The task comprises two steps. The *content selection* subtask determines which set of properties distinguish the target entity, and the *linguistic realization* part defines how these properties are translated into natural language. There is a long history of research on generating referring expressions. In one of the recent approaches, Kutlak et al. [146] convert property–value pairs to text using a hand-crafted mapping scheme. However, their method requires specific templates for each domain. Applying template-based methods to open-domain knowledge bases is extremely challenging, as this would require too many different templates for different types of entities. Recently, Ferreira et al. [139] proposed an end-to-end neural approach to REG called NeuralREG. They used a delexicalized WebNLG corpus for the training and evaluation of their model. NeuralREG generates a delexicalized template of the referring expression.

6.5.3 Neural Text Summarization

Generating descriptions for entities is related to the task of text summarization in that the salience of information needs to be assessed [147]. Similar to abstractive summarization, our task requires the generation of words not seen in the inputs. At the same time, in a similar vein to extractive summarization, our task also requires a selection of words from input facts for copying to the output sequence. The surge of sequence-to-sequence language modeling via LSTMs naturally extends to the task of abstractive summarization by training a model to accept a longer sequence as input and learning to generate a shorter compressed sequence as a summary. To this end, Rush et al. [148] employed this idea to generate a short headline from the first sentence of a text. Recently, Liu et al. [149] presented a model that generates an entire Wikipedia article via a neural decoder component that performs abstractive summarization of multiple source documents. Our work differs from such previous work in that we do not consider a text sequence as input. Rather, our inputs are a series of property–value pairs deemed as facts.

6.5.4 Pointer Networks and Copy Mechanisms

In order to learn how to solve combinatorial optimization problems that involve output dictionaries of varying sizes, such as the traveling salesman problem, Vinyals et al. [150] proposed a deep neural architecture known as *Pointer Networks*. Such networks rely on an attention mechanism to repeatedly select elements from the input as output. Subsequent works [151, 152, 153] incorporated this idea into hybrid text-based sequence-to-sequence models that occasionally select words from the input and otherwise generate words from a regular output dictionary. This addresses how to cope with rare and unknown words in the desired output sequence, which constitutes one of the key challenges in deploying neural text generation in practice. Since it is not feasible to directly learn from existing examples how to generate all possible vocabulary words that might be needed in the output, oftentimes, it is easier to learn to directly select suitable words from the given input. See et al.

[154] investigated the use of such architectures for the task of abstractive text summarization, so as to better cope with long input texts. As mentioned, Ahn et al. [135] rely on a similar copy mechanism to transform facts into summaries.

While our model adopts similar principles, there is a significant difference in our copy mechanism. In the above-mentioned existing works, the pointing and copying is a one-step process, in which a token within a context window is chosen based on simple attention weights. In our model, the pointing and copying is a two-step process. The model first needs to identify a pertinent fact that is salient enough to merit consideration. Then, within the chosen fact, it selects a suitable word for copying. The context window also varies depending on the number of facts and the number of words within a fact. In our experiments, we show that our approach greatly outperforms the model with copy mechanism proposed by Ahn et al.

6.6 Discussion

This chapter introduced a new task of generating succinct textual descriptions of entities based on available facts. Although the experiments are done on Wikidata, one can extend the methods presented in this chapter to other knowledge graphs. However, the scope of this study is limited to the English language. In principle, the proposed methods are language agnostic, and therefore, they should be applicable to other languages.

Another limitation of this study is that the experiments are done on entities belonging to the most frequent semantic types in Wikidata. Wikidata semantic types have a long-tail distribution where many semantic types have only a few instances. Learning to generate descriptions for such entities requires overcoming two challenges. First, many of these long-tail entities do not have sufficient factual coverage from where our proposed models can draw signals. Secondly, learning from a few instances requires a few-shot learning capabilities. This is indeed a challenging problem for future research in this direction.

Finally, our proposed fact embedding method obviates the disparity in representation

learning methods between texts and knowledge graphs by treating the facts as sequences of tokens. Note that Ahn et al.'s model cannot be extended to newly added entities in a knowledge graph without retraining TransE embeddings. In contrast, our proposed models can deal with any arbitrary entity with sufficient facts.

CHAPTER 7
CONCLUSIONS

This dissertation proposed a series of neural methods to deal with entity-centric knowledge in natural language. The overarching theme of this dissertation was to enhance entity-centric knowledge in knowledge graphs by means of extracting additional facts from text documents and leveraging the existing facts in knowledge graphs. In particular, it introduced an efficient and effective neural method for entity-related knowledge extraction from texts, a novel framework for inductive representation learning and explainable reasoning with factual knowledge present in open-domain knowledge graphs, novel methods to enrich knowledge graphs with concise and precise entity descriptions, and a neural method for better retrieval of factual knowledge for entity-centric search queries.

The key contributions and findings of Chapters 3, 4, 5, and 6 are summarized in the following.

In Chapter 3, a BERT-based dual encoder model is proposed that performs the tasks of entity linking and relation extraction. In contrast to the existing *retrieve and rerank* paradigm for entity linking that uses a BERT-based dual encoder to retrieve candidate entities and a BERT-based cross-encoder model for candidate reranking, this dissertation only uses a dual encoder model. While the BLINK model that follows the *retrieve and rerank* paradigm processes only one entity mention in each pass, the proposed model can process multiple mentions of entities in a single pass. This leads to efficient training and inference in wall-clock time. Additionally, it is observed in the empirical results that processing multiple mentions improves the recall of candidate retrieval and the overall performance over other models that processes one mention at a time. This observation suggests that processing multiple mentions in a context helps learn better representations of the mentions and candidates.

The dual encoder model is extended for end-to-end entity linking and relation extraction tasks. The model extracts spans of entity mentions, disambiguate the mentions by mapping them to their canonical form in a target knowledge graph and extracts possible relations between pairs of entity mentions. In contrast to the pipelined approach, where a different model is responsible for each of the tasks mentioned above, this approach performs all these tasks using a single end-to-end differentiable model. Additionally, the experimental results presented in Chapter 3 show that exhaustively enumerating all possible spans yields better precision/recall in span detection than a sequence tagging model using BIO tags.

In Chapter 4, a model is proposed for explainable link prediction for emerging entities in knowledge graphs. The model consists of two modules: (1) a modified graph transformer module to inductively learn entity representations and (2) a policy gradient-based reinforcement learning module for explainable link prediction, which are trained jointly in an end-to-end differentiable framework. This chapter also introduced three new benchmark datasets for this task. The empirical results demonstrate the advantages of the proposed model compared to other inductive representation learning and explainable reasoning methods for link prediction. Moreover, the proposed method overcomes a key disadvantage of existing state-of-the-art embedding-based models for link prediction. The existing models work only for a static snapshot of a knowledge graph, and therefore, requires retraining of the model when new entities are added in a knowledge graph. In contrast, the proposed model enables link prediction for newly-added entities without being retrained.

A BERT-based ranking model is proposed in Chapter 5 to return a ranked list of facts to enable dynamic fact retrieval for entity-centric search queries. It is a supervised model for query-aware entity summarization that performs ranking based on two human-defined metrics – importance and relevance, and their linear combination. In contrast to prior work, this model does not require various statistical signals pre-computed from the underlying knowledge graph as input. This makes the proposed model capable of handling ad-hoc search queries.

Chapter 6 introduces a novel task of generating succinct textual descriptions of entities in a knowledge graph using the existing facts. Two auto-regressive description generation model is proposed. The first model uses an encoder-decoder model with a dynamic memory network to memorize the factual information required to emit words in the output description. The second model uses an encoder-decoder model with an explicit copy mechanism to copy out-of-vocabulary words from the input facts to the output descriptions. The generated descriptions can be useful in various natural language processing tasks such as fine-grained entity typing, entity disambiguation, enhancing the knowledge panels' informativeness in search engine results, etc. The proposed method generated precise and concise descriptions of several entities belonging to various semantic types in Wikidata that have missing English descriptions. At the time of their introduction, the proposed methods were the first to deal with sparsely available entity descriptions in Wikidata.

7.1 Future Work

While the proposed methods in this dissertation have made much progress in dealing with entity-centric knowledge in the realm of natural language processing, there are some limitations. This section highlights some of these limitations and some future directions.

7.1.1 Unified Representation Learning for Texts and Knowledge Graphs

The disparity in representation learning methods for texts and knowledge graphs is detrimental for seamless conversion of factual knowledge in unstructured texts to structured triples in knowledge graphs and vice-versa. This dissertation attempted to narrow this disparity by treating knowledge graph facts as natural language texts (Chapters 5 and 6). Multi-head attention-based contextual representation learning has shown tremendous success in natural language processing tasks. This dissertation used a similar representation learning method for graph-structured data in Chapter 4, further reducing representation learning disparity between texts and knowledge graphs. However, developing a unified rep-

resentation learning method for structured knowledge graphs and unstructured texts could be a promising future direction. To this end, recently proposed Transformer-based language models [155, 156] have injected factual knowledge into the language model pretraining by augmenting the pre-training objective with entity linking. These methods have shown improved performance in various knowledge-intensive tasks.

7.1.2 Beyond Contextual Representation Learning

A key constraint in automatic knowledge extraction from text is the dependence on domain-specific annotated data to train models. For many domains (e.g., biomedical, legal, etc.), curating such dataset is expensive and rely on expert annotators. Moreover, the success of transfer learning using pre-trained language models is limited as cross-domain transfers often need fine-tuning using domain-specific annotated data. This shows that learning contextual representations of tokens is not sufficient, and the model must learn *meaning representation* to 'understand' relations among entity mentions in texts. Graph-based formalism such as AMR parsing is a promising research direction in this regard. However, it is limited to capturing only sentence-level semantics. Exploring the use of meaning representation as a tool for knowledge extraction from the text - first, at sentence-level and then at document-level is a future work.

Another advantage of graph-based formalism is the similarity in representation learning of entity mentions in text and entities in a knowledge graph. We can leverage graph representation learning methods in both cases. Additionally, logical rule induction on graph-structured AMR representation may help to discover implicit relations among entities.

7.1.3 Efficient Knowledge Graph Reasoning

Learning logic rules from the training sub-graph involves path exploration in an exponentially large space to search for candidate logic rules. It is often hard to learn good quality logic rules through this expensive process. The existing approaches explore the search

space of entities and their relations. In contrast, a semantic type augmented path exploration can operate at the schema level, i.e., the search space consists of semantic types and their relations. Since the number of semantic types is much lesser than the number of entities and each relation can only be associated with a fixed set of semantic types, the search space will reduce, leading to more efficient path exploration. Upon finding a general rule in the semantic type space, a model can instantiate the rule with specific entities of the respective types.

As mentioned at the beginning of this dissertation, understanding entities and their relationships is a key for machines to understand human spoken and written languages. Future research in this direction should explore new frontiers for better representation learning and reasoning methods to deal with the complex interactions between entities in natural language. Additionally, future research should also explore ways of combining the structured and unstructured sources of factual knowledge in a seamless manner.

ACKNOWLEDGMENT OF PREVIOUS PUBLICATIONS

P1 Rajarshi Bhowmik, Karl Stratos, and Gerard de Melo, "Fast and Effective Biomedical Entity Linking Using a Dual Encoder," in Proceedings of the 12th International Workshop on Health Text Mining and Information Analysis (LOUHI), EACL 2021.

P2 Rajarshi Bhowmik and Gerard de Melo, "Explainable link prediction for emerging entities in knowledge graphs," in Proceedings of the 19th International Semantic Web Conference (ISWC) 2020.

P3 Atharva Prabhat Paranjpe, Rajarshi Bhowmik, and Gerard de Melo, "Facts that matter: dynamic fact retrieval for entity-centric search queries," in Proceedings of the 19th International Semantic Web Conference (ISWC) 2020.

P4 Rajarshi Bhowmik and Gerard de Melo, "Be concise and precise: synthesizing open-domain entity descriptions from facts," in The World Wide Web Conference (WWW '19). Association for Computing Machinery, New York, NY, USA, 116–126.

P5 Rajarshi Bhowmik and Gerard de Melo, "Generating fine-grained open vocabulary entity typedescriptions," in Proceedings of ACL 2018, Melbourne, Australia: Association for Computational Linguistics, 2018, 877–888.

REFERENCES

[1] G. Weikum, L. Dong, S. Razniewski, and F. Suchanek, *Machine knowledge: Creation and curation of comprehensive knowledge bases*, 2021. arXiv: 2009.11564 [cs.AI].

[2] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J. E. Labra Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A.-C. Ngonga Ngomo, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and A. Zimmermann, "Knowledge graphs," *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, Jul. 2021.

[3] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic web*, vol. 6, no. 2, pp. 167–195, 2015.

[4] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," ser. WWW '07, Banff, Alberta, Canada: Association for Computing Machinery, 2007, pp. 697–706, ISBN: 9781595936547.

[5] D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[6] O. Bodenreider, "The unified medical language system (umls): Integrating biomedical terminology," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D267–D270, 2004.

[7] N. Kolitsas, O.-E. Ganea, and T. Hofmann, "End-to-end neural entity linking," in *Proceedings of the 22nd Conference on Computational Natural Language Learning*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 519–529.

[8] L. Logeswaran, M.-W. Chang, K. Lee, K. Toutanova, J. Devlin, and H. Lee, "Zero-shot entity linking by reading entity descriptions," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3449–3460.

[9] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer, "Zero-shot entity linking with dense entity retrieval," in *arXiv:1911.03814*, 2019.

[10] D. Q. Nguyen, "An overview of embedding models of entities and relationships for knowledge base completion," *CoRR*, vol. abs/1703.08098, 2017. arXiv: 1703.08098.

[11] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. J. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," *CoRR*, vol. abs/1711.05851, 2017. arXiv: 1711.05851.

[12] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," *CoRR*, vol. abs/1808.10568, 2018. arXiv: 1808.10568.

[13] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 229–256, May 1992.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 5998–6008.

[15] D. B. Lenat, "Cyc: A large-scale investment in knowledge infrastructure," *Communication of ACM*, vol. 38, no. 11, pp. 33–38, Nov. 1995.

[16] G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[17] T. Mitchell and E. Fredkin, "Never-ending language learning," in *2014 IEEE International Conference on Big Data (Big Data)*, 2014, pp. 1–1.

[18] G. de Melo and G. Weikum, "MENTA: Inducing multilingual taxonomies from Wikipedia," in *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010)*, (Oct. 26, 2010), J. Huang, N. Koudas, G. Jones, X. Wu, K. Collins-Thompson, and A. An, Eds., Toronto, Canada: ACM, Oct. 2010, pp. 1099–1108, ISBN: 978-1-4503-0099-5.

[19] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. D. Cao, J. Thorne, Y. Jernite, V. Plachouras, T. Rocktäschel, and S. Riedel, *Kilt: A benchmark for knowledge intensive language tasks*, 2020. arXiv: 2009.02252 [cs.CL].

[20] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26, Curran Associates, Inc., 2013.

[21] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, Feb. 2015.

[22] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, Jun. 2014.

[23] B. Yang, S. W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, May 2015.

[24] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2071–2080.

[25] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, *Rotate: Knowledge graph embedding by relational rotation in complex space*, 2019. arXiv: 1902.10197 [cs.LG].

[26] Z. Zhang, J. Cai, Y. Zhang, and J. Wang, "Learning hierarchy-aware knowledge graph embeddings for link prediction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, pp. 3065–3072, Apr. 2020.

[27] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, AAAI Press, 2018, pp. 1811–1818.

[28] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," in *Proceedings of NAACL 2018*, 2018, pp. 327–333.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[30] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing*, 2017.

[31] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018.

[32] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proceedings of ESWC 2018*, ser. LNCS, vol. 10843, Springer, 2018, pp. 593–607.

[33] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto, "Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach," in *Proceedings of IJCAI*, AAAI Press, 2017, pp. 1802–1808, ISBN: 978-0-9992411-0-3.

[34] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou, "End-to-end structure-aware convolutional networks for knowledge base completion," in *Proceedings of AAAI*, 2019.

[35] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar, "Composition-based multi-relational graph convolutional networks," in *International Conference on Learning Representations*, 2020.

[36] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, *A survey on knowledge graphs: Representation, acquisition and applications*, 2021. arXiv: 2002.00388 [cs.CL].

[37] N. Lao, T. M. Mitchell, and W. W. Cohen, "Random walk inference and learning in A large scale knowledge base," in *Proceedings of EMNLP 2011*, ACL, 2011, pp. 529–539.

[38] M. Gardner, P. P. Talukdar, B. Kisiel, and T. M. Mitchell, "Improving learning and inference in a large knowledge-base using latent syntactic cues," in *Proceedings of EMNLP 2013*, ACL, 2013, pp. 833–838.

[39] M. Gardner, P. P. Talukdar, J. Krishnamurthy, and T. M. Mitchell, "Incorporating vector space similarity in random walk inference over knowledge bases," in *Proceedings of EMNLP 2014*, ACL, 2014, pp. 397–406.

[40] A. Neelakantan, B. Roth, and A. McCallum, "Compositional vector space models for knowledge base completion," in *Proceedings of ACL 2015*, ACL, 2015.

[41] K. Guu, J. Miller, and P. Liang, "Traversing knowledge graphs in vector space," in *Proceedings of EMNLP 2015*, ACL, 2015, pp. 318–327.

[42] W. Xiong, T. Hoang, and W. Y. Wang, "DeepPath: A reinforcement learning method for knowledge graph reasoning," in *Proceedings of EMNLP 2017*, ACL, 2017.

[43] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. J. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," *arXiv*, vol. 1711.05851, 2017. arXiv: 1711.05851.

[44] Y. Shen, J. Chen, P.-S. Huang, Y. Guo, and J. Gao, "M-Walk: Learning to walk over graphs using Monte Carlo Tree Search," in *Advances in Neural Information Processing Systems 31*, Curran Associates, Inc., 2018, pp. 6786–6797.

[45] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek, "Fast rule mining in ontological knowledge bases with AMIE++," *The VLDB Journal*, vol. 24, no. 6, pp. 707–730, 2015.

[46] C. Meilicke, M. W. Chekol, D. Ruffinelli, and H. Stuckenschmidt, "An introduction to AnyBURL," in *KI 2019: Advances in Artificial Intelligence*, Cham: Springer International Publishing, 2019, pp. 244–248, ISBN: 978-3-030-30179-8.

[47] V. T. Ho, D. Stepanova, M. H. Gad-Elrab, E. Kharlamov, and G. Weikum, "Rule learning from knowledge graphs guided by embedding models," in *Proceedings of ISWC 2018*, Cham: Springer International Publishing, 2018, pp. 72–90, ISBN: 978-3-030-00671-6.

[48] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*, 1992.

[49] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Suntec, Singapore: Association for Computational Linguistics, Aug. 2009, pp. 1003–1011.

[50] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open information extraction from the web," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, ser. IJCAI'07, Hyderabad, India: Morgan Kaufmann Publishers Inc., 2007, pp. 2670–2676.

[51] A. Fader, S. Soderland, and O. Etzioni, "Identifying relations for open information extraction," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK.: Association for Computational Linguistics, Jul. 2011, pp. 1535–1545.

[52] L. Del Corro and R. Gemulla, "Clausie: Clause-based open information extraction," in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13, Rio de Janeiro, Brazil: Association for Computing Machinery, 2013, pp. 355–366, ISBN: 9781450320351.

[53] K. Gashteovski, R. Gemulla, and L. del Corro, "MinIE: Minimizing facts in open information extraction," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 2630–2640.

[54] G. Cheng, T. Tran, and Y. Qu, "Relin: Relatedness and informativeness-based centrality for entity summarization," in *The Semantic Web – ISWC 2011*, L. Aroyo,

C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, and E. Blomqvist, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 114–129, ISBN: 978-3-642-25073-6.

[55] A. Thalhammer and A. Rettinger, "Browsing DBpedia entities with summaries," in *ESWC (Satellite Events)*, V. Presutti, E. Blomqvist, R. Troncy, H. Sack, I. Papadakis, and A. Tordai, Eds., ser. Lecture Notes in Computer Science, vol. 8798, Springer, 2014, pp. 511–515, ISBN: 978-3-319-11954-0.

[56] P. Langer, P. Schulze, S. George, M. Kohnen, T. Metzke, Z. Abedjan, and G. Kasneci, "Assigning global relevance scores to DBpedia facts," in *International Workshop on Data Engineering meets the Semantic Web (DESWeb)*, Chicago, IL, 2014.

[57] A. Thalhammer, N. Lasierra, and A. Rettinger, "LinkSUM: Using Link Analysis to Summarize Entity Data," in *Proceedings of the 16th International Conference on Web Engineering (ICWE 2016)*, ser. Lecture Notes in Computer Science, vol. 9671, Springer International Publishing, 2016, pp. 244–261, ISBN: 978-3-319-38791-8.

[58] F. Hasibi, K. Balog, and S. E. Bratsberg, "Dynamic factual summaries for entity cards," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '17, ACM, 2017, pp. 773–782.

[59] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26, Curran Associates, Inc., 2013.

[60] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543.

[61] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. 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)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237.

[62] L. J. Elman, "Back propagation through time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[63] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[64] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. 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 (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734.

[65] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2016. arXiv: 1409.0473 [cs.CL].

[66] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. arXiv: 1607.06450 [stat.ML].

[67] J. Devlin, M.-W. Chang, K. Lee, and K. 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, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[68] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 19–27.

[69] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, J. Wren, Ed., Sep. 2019.

[70] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14, New York, New York, USA: Association for Computing Machinery, 2014, pp. 701–710, ISBN: 9781450329569.

[71] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15, Florence, Italy: International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077, ISBN: 9781450334693.

[72] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 855–864, ISBN: 9781450342322.

[73] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018, accepted as poster.

[74] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[75] S. Vashishth, R. Joshi, R. Dutt, D. Newman-Griffis, and C. Rose, "MedType: Improving Medical Entity Linking with Semantic Type Prediction," *arXiv e-prints*, arXiv:2005.00460, arXiv:2005.00460, May 2020. arXiv: 2005.00460 [cs.CL].

[76] M. Neumann, D. King, I. Beltagy, and W. Ammar, "ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing," in *Proceedings of the 18th BioNLP Workshop and Shared Task*, Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 319–327. eprint: arXiv:1902.07669.

[77] N. Gupta, S. Singh, and D. Roth, "Entity linking via joint encoding of types, descriptions, and context," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2681–2690.

[78] J. Raiman and O. Raiman, "Deeptype: Multilingual entity linking by neural type system evolution," *arXiv preprint arXiv:1802.01021*, 2018.

[79] T. Févry, N. FitzGerald, L. B. Soares, and T. Kwiatkowski, *Empirical evaluation of pretraining strategies for supervised entity linking*, 2020. arXiv: 2005.14253 [cs.CL].

[80] R. Leaman and Z. Lu, "Taggerone: Joint named entity recognition and normalization with semi-markov models," *Bioinformatics*, vol. 32, no. 18, pp. 2839–2846, 2016.

[81] A. R. Aronson, "Metamap: Mapping text to the umls metathesaurus," *Bethesda, MD: NLM, NIH, DHHS*, vol. 1, p. 26, 2006.

[82] G. K. Savova, J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. C. Kipper-Schuler, and C. G. Chute, "Mayo clinical text analysis and knowledge extraction system (ctakes): Architecture, component evaluation and applications," *Journal of the American Medical Informatics Association*, vol. 17, no. 5, pp. 507–513, 2010.

[83] L. Soldaini and N. Goharian, "Quickumls: A fast, unsupervised approach for medical concept extraction," in *MedIR workshop, sigir*, 2016, pp. 1–4.

[84] M. Zhu, B. Celikkaya, P. Bhatia, and C. K. Reddy, *Latte: Latent type modeling for biomedical entity linking*, 2019. arXiv: 1911.09787 [cs.CL].

[85] D. Xu, Z. Zhang, and S. Bethard, "A generate-and-rank framework with semantic type regularization for biomedical concept normalization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 8452–8464.

[86] Z. Ji, Q. Wei, and H. Xu, "Bert-based ranking for biomedical entity normalization," *AMIA Summits on Translational Science Proceedings*, vol. 2020, p. 269, 2020.

[87] Y. Cao, L. Hou, J. Li, and Z. Liu, "Neural collective entity linking," *CoRR*, vol. abs/1811.08603, 2018. arXiv: 1811.08603.

[88] I. Yamada and H. Shindo, "Pre-training of deep contextualized embeddings of words and entities for named entity disambiguation," *arXiv preprint arXiv:1909.00426*, 2019.

[89] Y. Yang, O. Irsoy, and K. S. Rahman, "Collective entity disambiguation with structured gradient tree boosting," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 777–786.

[90] M. Xue, W. Cai, J. Su, L. Song, Y. Ge, Y. Liu, and B. Wang, "Neural collective entity linking based on recurrent random walk network learning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, Jul. 2019, pp. 5327–5333.

[91] J. Giorgi, X. Wang, N. Sahar, W. Y. Shin, G. D. Bader, and B. Wang, *End-to-end named entity recognition and relation extraction using pre-trained language models*, 2019. arXiv: 1912.13415 [cs.CL].

[92] M. Eberts and A. Ulges, *Span-based joint entity and relation extraction with transformer pre-training*, 2019. arXiv: 1909.07755 [cs.CL].

[93] P. Crone, *Deeper task-specificity improves joint entity and relation extraction*, 2020. arXiv: 2002.06424 [cs.CL].

[94] D. Q. Nguyen and K. Verspoor, "End-to-end neural relation extraction using deep biaffine attention," in *Proceedings of the 41st European Conference on Information Retrieval*, 2019.

[95] S. Zhao, M. Hu, Z. Cai, and F. Liu, "Modeling dense cross-modal interactions for joint entity-relation extraction," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed., Main track,

International Joint Conferences on Artificial Intelligence Organization, Jul. 2020, pp. 4032–4038.

[96] T. Bansal, P. Verga, N. Choudhary, and A. McCallum, *Simultaneously linking entities and extracting relations from biomedical text without mention-level supervision*, 2019. arXiv: 1912.01070 [cs.CL].

[97] X. Lin, H. Li, H. Xin, Z. Li, and L. Chen, "Kbpearl: A knowledge base population system supported by joint entity and relation linking," *Proc. VLDB Endow.*, vol. 13, no. 7, pp. 1035–1049, Mar. 2020.

[98] B. D. Trisedya, G. Weikum, J. Qi, and R. Zhang, "Neural relation extraction for knowledge base enrichment," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 229–240.

[99] D. Gillick, S. Kulkarni, L. Lansing, A. Presta, J. Baldridge, E. Ie, and D. Garcia-Olano, "Learning dense representations for entity retrieval," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 528–537.

[100] B. Z. Li, S. Min, S. Iyer, Y. Mehdad, and W.-t. Yih, "Efficient one-pass end-to-end entity linking for questions," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 6433–6441.

[101] S. Mohan and D. Li, "Medmentions: A large biomedical corpus annotated with umls concepts," *ArXiv*, vol. abs/1902.09476, 2019.

[102] J. Li, Y. Sun, R. J. Johnson, D. Sciaky, C.-H. Wei, R. Leaman, A. P. Davis, C. J. Mattingly, T. C. Wiegers, and Z. Lu, "Biocreative v cdr task corpus: A resource for chemical disease relation extraction," *Database*, vol. 2016, 2016.

[103] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv:2004.05150*, 2020.

[104] H. Peng, T. Gao, X. Han, Y. Lin, P. Li, Z. Liu, M. Sun, and J. Zhou, "Learning from Context or Names? An Empirical Study on Neural Relation Extraction," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 3661–3672.

[105] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Proceedings of the 2017 Con-*

ference on Empirical Methods in Natural Language Processing (EMNLP 2017), 2017, pp. 35–45.

[106] R. Xing, J. Luo, and T. Song, "Biorel: Towards large-scale biomedical relation extraction," *BMC bioinformatics*, vol. 21, no. 16, pp. 1–13, 2020.

[107] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest, and E. Simperl, "T-REx: A large scale alignment of natural language with knowledge base triples," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan: European Language Resources Association (ELRA), May 2018.

[108] Y. Yao, D. Ye, P. Li, X. Han, Y. Lin, Z. Liu, Z. Liu, L. Huang, J. Zhou, and M. Sun, "DocRED: A large-scale document-level relation extraction dataset," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 764–777.

[109] R. Bhowmik and G. de Melo, "Be concise and precise: Synthesizing open-domain entity descriptions from facts," in *Proceedings of The Web Conference 2019*, San Francisco, 2019.

[110] S. S. Dasgupta, S. N. Ray, and P. Talukdar, "HyTE: Hyperplane-based temporally aware knowledge graph embedding," in *Proceedings of EMNLP 2018*, 2018.

[111] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "DyRep: Learning representations over dynamic graphs," in *ICLR*, 2019.

[112] R. Koncel-Kedziorski, D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi, "Text Generation from Knowledge Graphs with Graph Transformers," 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)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 2284–2293.

[113] V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

[114] R. Bhowmik and G. de Melo, "Explainable link prediction for emerging entities in knowledge graphs," in *Proceedings of ISWC 2020*, (Nov. 2, 2020), 2020.

[115] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 687–696.

[116] K. Toutanova, V. Lin, W.-t. Yih, H. Poon, and C. Quirk, "Compositional learning of embeddings for relation paths in knowledge base and text," in *Proceedings of ACL 2016*, ACL, 2016.

[117] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[118] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 9, PMLR, 13–15 May 2010, pp. 249–256.

[119] A. P. Paranjpe, R. Bhowmik, and G. de Melo, "Facts that matter: Dynamic fact retrieval for entity-centric search queries," in *Proceedings of ISWC 2020*, 2018.

[120] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002.

[121] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd International Conference on Machine Learning*, ser. ICML '05, Bonn, Germany: Association for Computing Machinery, 2005, pp. 89–96, ISBN: 1595931805.

[122] W. Chen, T.-y. Liu, Y. Lan, Z.-m. Ma, and H. Li, "Ranking measures and loss functions in learning to rank," in *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds., vol. 22, Curran Associates, Inc., 2009.

[123] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. USA: Cambridge University Press, 2008, ISBN: 0521865719.

[124] V. Lopez, V. Uren, M. Sabou, and E. Motta, "Is question answering fit for the semantic web? a survey," *Semant. Web*, vol. 2, no. 2, pp. 125–155, Apr. 2011.

[125] C. Unger, A. Freitas, and P. Cimiano, "An introduction to question answering over linked data," in *Reasoning Web. Reasoning on the Web in the Big Data Era: 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings*, M. Koubarakis, G. Stamou, G. Stoilos, I. Horrocks, P. Kolaitis, G. Lausen, and G. Weikum, Eds. Cham: Springer International Publishing, 2014, pp. 100–140, ISBN: 978-3-319-10587-1.

[126] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword searching and browsing in databases using banks," *Proceedings 18th International Conference on Data Engineering*, pp. 431–440, 2002.

[127] T. Tran, H. Wang, S. Rudolph, and P. Cimiano, "Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data," in *25th IEEE International Conference on Data Engineering*, 2009, pp. 405–416.

[128] P. Li, C. Burges, and Q. Wu, "McRank: learning to rank using multiple classification and gradient boosting," Jan. 2007.

[129] A. Shashua and A. Levin, "Ranking with large margin principle: Two approaches," in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, ser. NIPS'02, Cambridge, MA, USA: MIT Press, 2002, pp. 961–968.

[130] K. Crammer and Y. Singer, "Pranking with ranking," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, ser. NIPS'01, Vancouver, British Columbia, Canada: MIT Press, 2001, pp. 641–647.

[131] D. Cossock and T. Zhang, "Subset ranking using regression," in *Proceedings of the 19th Annual Conference on Learning Theory*, ser. COLT'06, Pittsburgh, PA: Springer-Verlag, 2006, pp. 605–619, ISBN: 3540352945.

[132] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *J. Mach. Learn. Res.*, vol. 4, no. null, pp. 933–969, Dec. 2003.

[133] R. Bhowmik and G. de Melo, "Generating fine-grained open vocabulary entity type descriptions," in *Proceedings of ACL 2018*, Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 877–888.

[134] R. Lebret, D. Grangier, and M. Auli, "Neural text generation from structured data with application to the biography domain," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1203–1213.

[135] S. Ahn, H. Choi, T. Pärnamaa, and Y. Bengio, *A neural knowledge language model*, 2017. arXiv: 1608.00318 [cs.CL].

[136] W. Chen, Y. Su, X. Yan, and W. Y. Wang, "KGPT: Knowledge-grounded pre-training for data-to-text generation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 8635–8648.

[137] C. Gardent, A. Shimorina, S. Narayan, and L. Perez-Beltrachini, "The WebNLG challenge: Generating text from RDF data," in *Proceedings of the 10th Interna-*

tional Conference on Natural Language Generation, Santiago de Compostela, Spain: Association for Computational Linguistics, Sep. 2017, pp. 124–133.

[138] B. D. Trisedya, J. Qi, R. Zhang, and W. Wang, "GTR-LSTM: A triple encoder for sentence generation from RDF data," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1627–1637.

[139] T. Castro Ferreira, D. Moussallem, Á. Kádár, S. Wubben, and E. Krahmer, "NeuralREG: An end-to-end approach to referring expression generation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1959–1969.

[140] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 2440–2448.

[141] C. Xiong, S. Merity, and R. Socher, "Dynamic memory networks for visual and textual question answering," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16, New York, NY, USA: JMLR.org, 2016, pp. 2397–2406.

[142] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002.

[143] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Proc. ACL workshop on Text Summarization Branches Out*, 2004, p. 10.

[144] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," in *Proceedings of the Second Workshop on Statistical Machine Translation*, ser. StatMT '07, Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 228–231.

[145] R. Vedantam, C. L. Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," *CoRR*, vol. abs/1411.5726, 2014. arXiv: 1411.5726.

[146] R. Kutlak, K. van Deemter, and C. Mellish, *Generation of referring expressions in large domains*, Jul. 2013.

[147] Q. Yang, Y. Cheng, S. Wang, and G. de Melo, "HiText: Text reading with dynamic salience marking," in *Proceedings of WWW 2017 (Digital Learning Track)*, (Apr. 3, 2017), Perth, Australia: ACM, 2017.

[148] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *arXiv preprint arXiv:1509.00685*, 2015.

[149] P. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating Wikipedia by summarizing long sequences," *CoRR*, vol. abs/1801.10198, 2018.

[150] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 2692–2700.

[151] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1631–1640.

[152] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany: Association for Computational Linguistics, 2016, pp. 140–149.

[153] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *CoRR*, vol. abs/1609.07843, 2016. arXiv: 1609.07843.

[154] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, 2017, pp. 1073–1083.

[155] M. E. Peters, M. Neumann, R. Logan, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, "Knowledge enhanced contextual word representations," 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)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 43–54.

[156] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "ERNIE: Enhanced language representation with informative entities," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1441–1451.