

Enriching Text-Based Human-Machine Interactions with Additional World Knowledge

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von
Debanjan Chaudhuri
aus
Kolkata, India

Bonn 2022

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen
Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Jens Lehmann
2. Gutachter: Prof. Dr. Asja Fischer
Tag der Promotion: 29.04.2022
Erscheinungsjahr: 2022

Abstract

In the last decade, the world has witnessed a massive upheaval in intelligent systems' design and development, especially for systems that can interact with humans. Such systems, capable of real-time interaction with end users can be realized in several ways, ranging from text/speech based to gesture or electroencephalogram (EEG). Speech/text-based human-machine interactive (HMI) systems have become part of human's everyday life in the form of Alexa, Siri, etc. Aforesaid HMI systems can be realized as question-answering systems or Dialogue systems/chatbots; where the former is capable of answering individual queries from the user while the latter has an additional understanding of the conversational context.

Both kinds of systems can potentially benefit from additional world knowledge, which is present across both unstructured and structured sources. Unstructured sources consist of textual knowledge present in the form of definitions or text articles; while, structured knowledge can be present in the form of, e.g., Knowledge Graphs or RDBMS. However, integrating both structured and unstructured knowledge while designing such systems is challenging because the models have to solve multiple tasks of inferencing, coherent response generation, and knowledge integration.

In this thesis, we explore different techniques for integrating additional world knowledge into methods used for implementing question-answering and dialogue systems. Firstly, we examine different techniques to incorporate structural information present in a Knowledge Graph (KG) into Knowledge-Graph-based question answering systems (KGQA). Here, we investigate how to improve KGQA with added structural information of the Knowledge Graph. Most KGQA systems do entity and relation linking separately and have no deeper understanding of the connected KG components and structure of the KG elements. We explore and propose different algorithms for KG-aware question answering, both as modular and end-to-end settings. Secondly, we investigate different techniques for incorporating world knowledge, present in the form of both structured and unstructured sources into multi-turn conversational systems. For unstructured knowledge integration, we probe end-to-end techniques to make retrieval-based chatbots better using textual definitions of in-domain keywords. Henceforth, we explore techniques to incorporate structured information present in the form of Knowledge Graphs into generative dialogue systems. Several novel neural network architectures are proposed which implicitly look into the Knowledge Graph during the dialogue generation process. Some architectures are for handling goal-oriented dialogues while others are designed to handle both goal and non-goal oriented dialogues. For the first model, we incorporate joint Knowledge Graph and word embeddings, projected in the same vector space for task-oriented dialogues. Secondly, we incorporate a copy mechanism into the response generation process for end-to-end generative dialogue systems, where the model can probabilistically copy information from the underlying Knowledge Graph based on the question. Finally, we use a pre-trained transformer (BERT) model which is capable of answering from the Knowledge Graph by jointly performing entity linking, relation identification while generating coherent responses at the same time.

Acknowledgements

First and foremost, I would like to thank Prof. Dr. Jens Lehmann for giving me the precious opportunity to work with him and conduct research under his supervision. I am grateful for the time and supervision Prof. Lehmann has provided throughout this dissertation. Moreover, I would like to thank Prof. Dr. Asja Fischer and Dr. Giulio Napolitano for finding the time to discuss ideas and providing additional supervision that laid the foundation stone for this work. The review provided by Dr. Napolitano for this thesis is invaluable. I would also like to thank the UKP Lab, TU Darmstadt, from where I got my main inspiration, motivation, and experience for conducting research. The main funding to conduct all the research required for this dissertation has been provided by Fraunhofer-Cluster of Excellence “Cognitive Internet Technologies” (CCIT).

Additionally, I am also very thankful to all my colleagues and friends at SDA and EIS who have provided invaluable feedback and contributions during this work. In particular, I would like to thank Md Rashad Al Hasan Rony, Debayan Banerjee, Mohnish Dubey, and Rostislav Nedelchev for the very valuable discussions and important contributions in the form of codes, pair programming that was required for conducting all the experiments for this thesis. I would also like to acknowledge the valuable contributions made by Agustinus Kristiadi and Firas Kassawat for the work they have contributed towards this work during their master studies. I would also like to express my gratitude towards Priyansh Trivedi from whose master’s thesis the preliminaries section of this thesis is adapted. Also a big thanks to my friend Dr. Fathoni A. Musyaffa for helping me during the final phase of this thesis.

Finally, I would like to express my deepest gratitude to all my family members and friends, especially my mother Mrs. Lekha Chaudhuri, and my father Mr. Samir Chaudhuri for believing in me and supporting me through and throughout.

Contents

1	Introduction	1
1.1	Motivation and Challenges	2
1.1.1	Knowledge and Artificial Intelligence	2
1.1.2	Challenges in Knowledge Integration	3
1.2	Research Questions	4
1.3	Thesis Overview	6
1.3.1	Contributions	7
1.3.2	Publications	8
2	Preliminaries	9
2.1	Natural Language Processing	9
2.1.1	Knowledge Sources	10
2.1.2	Structured Knowledge Sources	10
2.2	Neural Networks	11
2.2.1	Deep Learning	13
2.2.2	Deep Learning Models	15
2.2.3	Deep Learning Architectures for NLP Tasks	21
2.3	Representing Language as Dense Vectors	23
2.3.1	Word Embeddings	23
2.3.2	Pre-trained Transformers	24
2.3.3	Knowledge Graph Embeddings	24
3	Related Work	27
3.1	Human Machine Interactive Systems	27
3.1.1	Computer Vision Based	27
3.1.2	Laser Based HMI	27
3.1.3	Bionic Technology Based	27
3.1.4	Speech Based HMI	28
3.2	Question Answering Systems	28
3.2.1	Retrieval Based Question Answering System	28
3.2.2	Reading Comprehension based Question Answering	29
3.2.3	Question Answering over Knowledge Graphs	29
3.3	Dialogue Systems	30
3.3.1	Generative models	30
3.3.2	Retrieval-based models	31

4	Improving Knowledge-Based Question Answering using Structural Information.	33
4.1	EARL	33
4.1.1	Introduction	33
4.1.2	Candidate Generation Steps	34
4.1.3	Using GTSP for Disambiguation	35
4.1.4	Using Connection Density for Disambiguation	37
4.1.5	Adaptive E/R Learning	39
4.1.6	Evaluation	40
4.1.7	Discussion	43
4.2	PNEL	43
4.2.1	Introduction	44
4.2.2	Encoding for Input	45
4.2.3	Training	46
4.2.4	Datasets	47
4.2.5	Evaluation	48
4.2.6	Discussion	51
4.3	ELiDi	52
4.3.1	Introduction	52
4.3.2	Model Description	53
4.3.3	Training & Inference	56
4.3.4	Datasets	58
4.3.5	Evaluation	58
4.3.6	Discussion	59
5	Improving Retrieval-Based Chatbots with added Domain Knowledge.	63
5.1	Introduction	63
5.2	Background	64
5.2.1	Problem definition	64
5.2.2	RNNs, BiRNNs and GRUs	65
5.2.3	Dual Encoder	66
5.3	Model description	66
5.3.1	Attention augmented encoding	66
5.3.2	Incorporating domain keyword descriptions	68
5.4	Experiment	69
5.4.1	Ubuntu multi-turn dialogue corpus	69
5.4.2	Model hyperparameters	70
5.5	Results	70
5.5.1	Comparison against baselines	70
5.5.2	Ablation study	71
5.5.3	Visualizing response attentions	71
5.5.4	Error analysis	72

6	Improving task-oriented, generative Chatbots using Knowledge Graph Embeddings	75
6.1	Introduction	75
6.2	Model Description	76
6.2.1	Attention Based Seq-to-seq Model	76
6.2.2	Predicting Intent	77
6.2.3	Training Joint Text and Knowledge Graph Embeddings	78
6.2.4	Regularizing using additional Entity Loss	79
6.2.5	Final objective Function	79
6.2.6	Key-Value Entity Look-up	80
6.3	Experiments	80
6.3.1	Dataset	80
6.3.2	Pre-processing and Model Hyper-parameters	81
6.4	Results	82
6.4.1	Ablation Study	83
6.4.2	Qualitative Analysis	83
6.4.3	Error Analysis	84
7	Using a Knowledge Copy Mechanism into the response-generation process	87
7.1	Introduction	87
7.2	Soccer Dialogues Dataset	88
7.2.1	Wizard-of-Oz Style Data Collection	88
7.2.2	Ensuring Coherence	90
7.2.3	Soccer Knowledge Graph	90
7.3	KG-Copy Model	91
7.3.1	KG-Copy Encoder	92
7.3.2	KG-Copy Decoder	92
7.3.3	Sentient Gating	92
7.4	Training and Model Hyper-parameters	93
7.4.1	Training Objective	93
7.4.2	Training Details	94
7.5	Evaluation	95
7.6	Discussion	95
7.6.1	Qualitative Analysis	95
7.6.2	Error Analysis	97
8	Grounding Dialogue Systems via Knowledge Graph Aware Decoding with Pre-trained Transformers	99
8.1	Introduction	99
8.2	Model Description	100
8.2.1	Query Encoding	102
8.2.2	Entity Detection	102
8.2.3	Input Query Encoder	102
8.2.4	Intermediate Representation	102
8.2.5	Decoding Process	103

8.2.6	Sub-Graph Encoding	103
8.3	Experimental Setup	104
8.3.1	Datasets	104
8.3.2	Evaluation Metrics	105
8.3.3	Model Settings	105
8.4	Results	106
8.5	Discussion	107
8.5.1	Ablation Study	108
8.5.2	Qualitative Analysis	109
8.5.3	Error Analysis	109
9	Conclusion and Future Directions	111
9.1	Answering the Research Questions	111
9.2	Future Directions	113
9.2.1	Transfer Learning for KGQA	113
9.2.2	Annealing Language Models with Knowledge Graph Information	114
9.2.3	Evaluating Knowledge Groundedness of Generated Dialogues	114
	Bibliography	117
10	List of Publications	137
10.1	Accepted Papers	137
10.2	Papers in Review	137
	List of Figures	139
	List of Tables	141

Introduction

Natural language-based human-machine interaction (HMI) is a sub-category of general Artificial Intelligence (AI). Systems and methods which can communicate with end-users have been studied for a long time [1] [2] [3]. Such systems can be broadly classified into question-answering systems and dialogue systems/chatbots. The former can also be realized as a question-answering system but with an additional understanding of the interaction context. Humans understand and interpret natural language with its knowledge and intelligence [1]. Intuitively, systems developed with integrated additional world knowledge potentially can also benefit human-machine interactive systems.

World knowledge can be present in the form of both structured and unstructured knowledge sources. The majority of data sources are unstructured [4], consisting of mostly free-text-based knowledge such as metadata, definitions, journals, books, e-mails. Structured knowledge on the other hand comprises knowledge organized in the form of a pre-defined data model. Examples of structured knowledge sources are Knowledge Graphs [5], relational databases (RDBMS). Both these types of data sources can potentially benefit question-answering and dialogue systems. For example, let us consider the dialogue happening in 1.1. The first question, *Who is the president of France* can be answered from the Knowledge Graph. However, the second question *what is the name of the incumbent before him*, requires more domain-specific understanding. In that case, integrating the definition of the political domain-specific word *incumbent* is potentially beneficial for the system. The third query is an example where the system could also use knowledge stored in a RDBMS database for responding to the user. It must be noted that in the case of this example, the first question is an example of a question-answering system, but the latter ones require contextual knowledge from the previous questions and answers.

Research objectives: Previous research have focused on integrating structural knowledge for Question Answering [6] [7] [8], textual knowledge for retrieval-based chatbots [9] [8], and additional KG information for generative dialogue systems [10] [11]. However, most of these techniques focus on having modular architectures for integrating additional knowledge into QA and dialogue systems. Potentially, integrating multiple sources in an end-to-end way can improve the performance of question answering and dialogue systems. In this research, we will focus on integrating additional world knowledge, from structured and unstructured sources into question answering and dialogue systems, in an end-to-end manner.

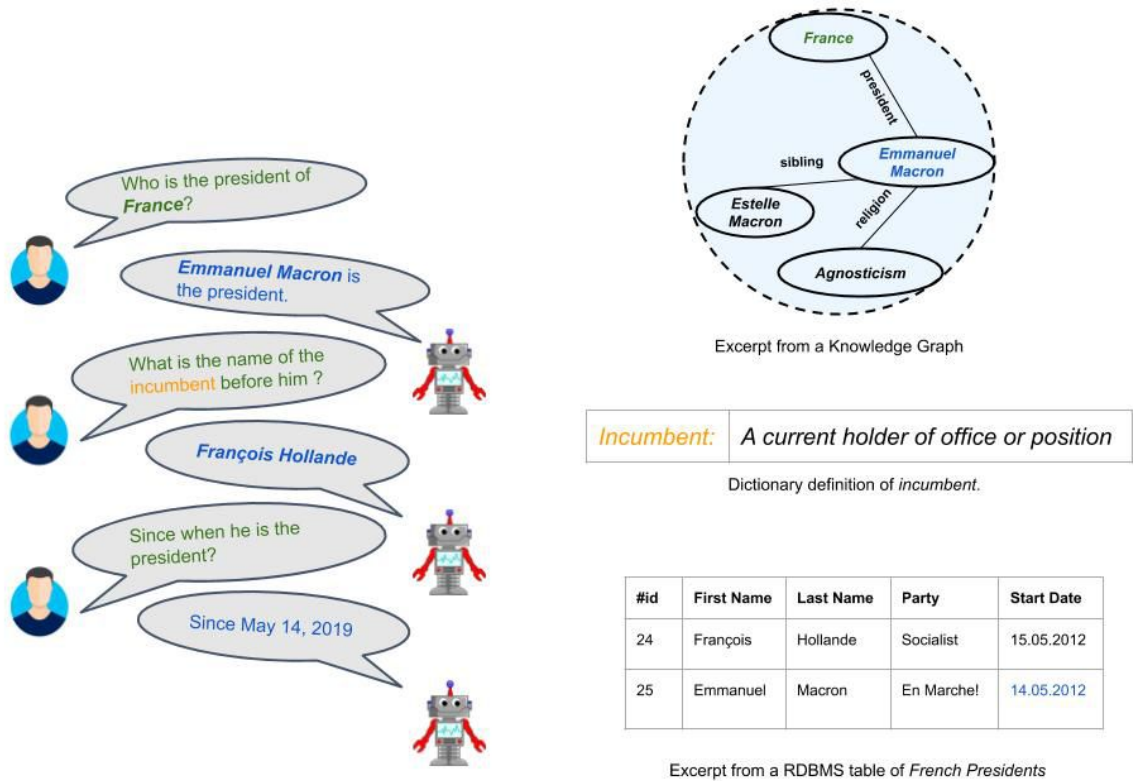


Figure 1.1: Example where a Conversation System can benefit from several knowledge sources in order to have knowledge-grounded conversation with the user.

1.1 Motivation and Challenges

1.1.1 Knowledge and Artificial Intelligence

Over the past century, several studies have been made on understanding human general intelligence. Spearman's [12] theory of human intelligence defines intelligence as a two-factored process, namely general and specific abilities. The general abilities have further been stratified with the Carroll-Horn-Cattell (CHC) model [13]. Their proposed model defines a hierarchical, three-stratum model of general cognitive intelligence as shown in figure 1.2. The second strata of the model comprises of 10 broad factors, among them the most researched factors being *Fluid Intelligence*, *Crystallized Intelligence*, *Broad Visual Perception* and *Cognitive Processing Speed*. *Fluid Intelligence* is interpreted as the property or capacity to solve complex, novel problems using inductive and deductive reasoning skills. *Crystallized Intelligence* on the other hand is the knowledge or information one collects during their lifetime or studies. *Broad Visual Perception*, is interpreted as the ability of an individual to generate, retain, retrieve and transform visual images. On the other hand, *Cognitive Processing Speed* is a broad ability to fluently perform relatively easy or over-learned tasks, particularly which require attention

and focused concentration.

Early advocates of Artificial intelligence focused on solving the general aspect of fluid intelligence, which quintessentially deals with the ability of a machine to reason. [14] proposed a general problem solver (GPS) which could solve mathematical problems that require deductive logic. Many more AI approaches also followed similar lines which focused on having artificial agents possessing reasoning skills. However, most of these approaches could only solve basic reasoning tasks and not real-world problems. Later onwards, the focus of AI research shifted from systems able to handle general problem-solving tasks to systems that firstly learn from the knowledge of human experts and then aid in automating the task of those experts to some extent [15]. Also as stated by later researchers of cognition, "the bottom line is that intelligence is a function of knowledge. One may have the potentiality of intelligence, but without knowledge, nothing will become of that intelligence" [16].

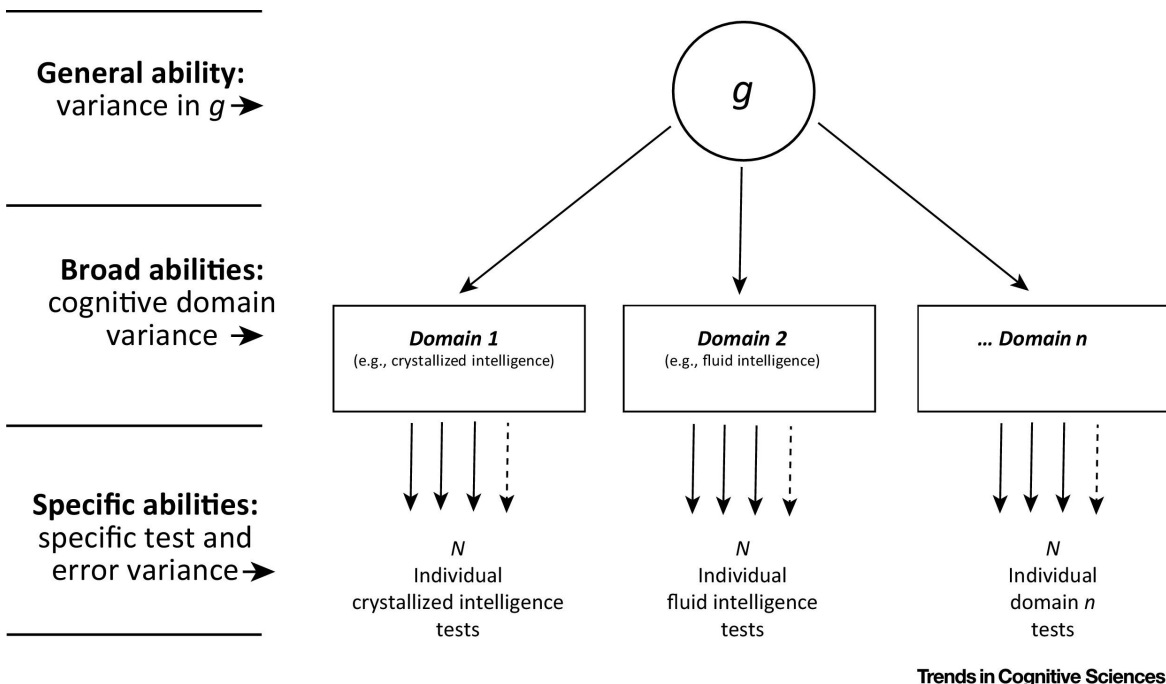


Figure 1.2: Hierarchical Structure of General Intelligence as defined by the CHC model. The last strata representing general intelligence as defined by [12]. Diagram adapted from [17]

Drawing inspirations from the studies in human cognition and psychology, in this thesis, we focus on designing AI agents which are able to implicitly integrate different knowledge sources while performing several downstream tasks for human-machine interaction.

1.1.2 Challenges in Knowledge Integration

World knowledge is available across multiple sources; although integrating additional knowledge is potentially beneficial for an AI system, however integrating knowledge into a chatbot or QA system is challenging.

Knowledge Integration

As mentioned previously, knowledge as an additional source is distributed across several sources, both structured and unstructured. Integrating such sources into a neural network model will depend on the downstream task, hence separate neural network architectures are required depending on the downstream task addressed. For example, a network architecture for an unstructured knowledge source needs to infer important concept(s) in the source which will affect the end-task such as response selection. While, for structured sources, the network architecture must be designed in a way where apart from determining importance, the model should also be able to infer and respond using the source.

Inferencing Capabilities

As mentioned previously, for the case of structured knowledge source integration into a neural network model, a specific downstream task would require the model to infer when the model should use the knowledge from the knowledge source and when to learn the task from the training data. For example, in the case of a dialogue system, if the task of the model is to have both knowledge-grounded and also chit-chat/small talks. The model, when integrated with additional knowledge should infer when to use the knowledge for the query being posed by the user and when to use the knowledge learned from the corpus, implicitly.

End-to-end Integration

Several previous works have worked on integrating additional knowledge into interactive systems, both for question answering [7] [18] and dialogue systems [9] [19] [20]. However, most of these approaches are modular, where the additional world knowledge is incorporated in a modular fashion, with individual task-specific modules designed to perform the final end-task of human-machine interaction. The problem with modular architecture is, several errors in a module would propagate and hence affect the performance of other downstream modules. On the other hand, modular architectures are easier to interpret and infer the outputs for every module and make more informed decisions. Hence, in this thesis, we focus on integrating additional knowledge using both modular and end-to-end, differentiable neural network architectures. However, empirical evidence suggests that end-to-end models perform better than modular approaches for the downstream tasks.

1.2 Research Questions

After discussing the motivation and challenges, we will head on to discuss the research questions. The sub research questions are described in 1.3

* Research Question 1

How the Knowledge Graph Structure can be incorporated for Knowledge Graph Question Answering?

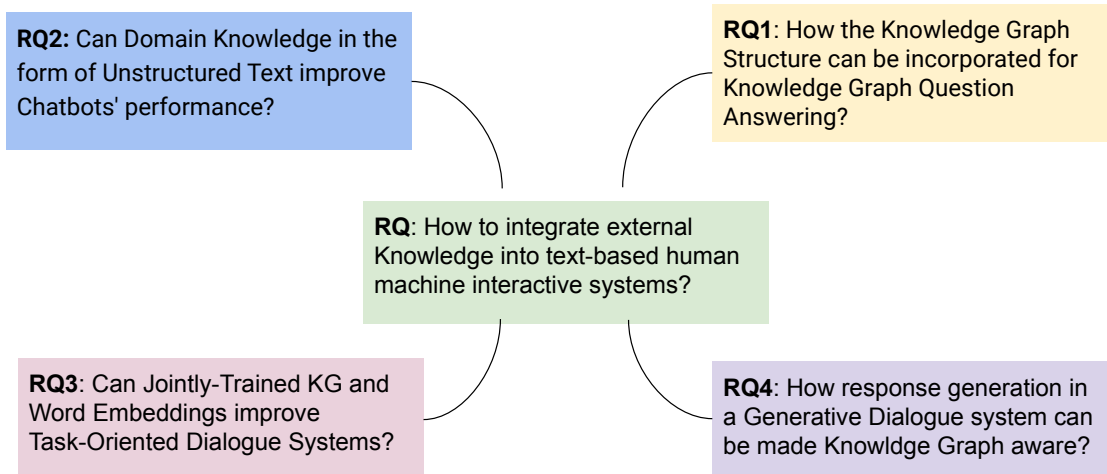


Figure 1.3: The sub-research questions contributing to the main Research Question.

Incorporating additional information of Knowledge Graph Structure can potentially improve the performance of Knowledge Graph-based Question Answering systems. For Knowledge Graph-based question answering systems, the main tasks are entity linking and relation detection. Both these tasks can benefit from the integration of additional knowledge in terms of Knowledge Graph structure, more specifically the knowledge about neighboring entities and relations connected at 1 or 2-hops. We would like to investigate what are the different strategies to incorporate this additional structural information into KGQA models. We would explore both end-to-end and modular approaches and how much they affect the performances of such systems. The models are evaluated in terms of f1 scores for the task of entity linking.

- **Research Question 2**

Can Domain Knowledge in the form of Unstructured Text improve Chatbots' performance?

For the second research question, we would like to investigate the task of adding unstructured domain knowledge for retrieval-based chatbots. Such knowledge, present in the form of textual

descriptions of domain-specific keywords can potentially improve the performance of the bots. The main objective here is to study end-to-end neural network models that can integrate this unstructured knowledge for the downstream task of response selection. We would empirically evaluate whether the proposed models integrated with additional domain knowledge can respond with better answers measured in terms of Recall@top-K.

- **Research Question 3**

Can Jointly-Trained KG and Word Embeddings improve Task-Oriented Dialogue Systems?

Both word and Knowledge graph embeddings can capture the semantic similarity between corresponding elements in separate vector spaces. Projecting them in the same vector space can capture the semantics between Knowledge Graph elements and words. These joint embeddings can be used as input for modeling dialogue generation and can potentially benefit the end task of knowledge grounded dialogue generation. For the third research question, we would like to study neural network architectures for integrating joint Knowledge Graph and word embeddings in the downstream task of knowledge grounded dialogue generation, for task-oriented chatbots. The proposed model's knowledge groundedness is evaluated in terms of BLEU scores.

- **Research Question 4**

How response generation in a Generative Dialogue system can be made Knowledge Graph aware?

Task-oriented dialogues are relatively easy to model because of their limited vocabulary sizes. Studying dialogue generation for task-oriented bots in the last research question, we would like to extend the research to incorporate knowledge groundedness for both task and non-task-oriented settings. Specifically, different neural network architectures that can implicitly integrate Knowledge Graphs during the response generation process are studied here. The main challenge for this research objective is to have systems that can produce knowledge aware, as well as grammatically correct responses. The improvements of the proposed models in terms of knowledge-groundedness are measured using entity f1 scores, while grammatical correctness and articulation are measured using BLEU scores.

1.3 Thesis Overview

This section narrates an overall, high-level overview of the thesis mentioning the scientific contributions made to answer the research questions defined in the previous section. Also, a list of publications and the thesis structure is defined here.

1.3.1 Contributions

Answering Research Question 1

For integrating additional structural knowledge into a Question Answering system, based on Knowledge Graphs, the system must be able to infer from additional structural connectivity information for the entities and relations in a query posed by the user. In order to answer this, we firstly investigate several modular approaches. We propose two modular approaches EARL and PNEL which use structural connectivity information as features from the probable set of entities and relations and finally using a supervised method to pick the most probable pair of entities and relations required to answer the question. Finally, we propose a novel neural network architecture, which uses the entity span and probable relation information in a single neural network model to predict the final entity-relation pair. This model is trained in an end-to-end, differentiable manner and achieves better performance than other similar models.

Answering Research Question 2

From question-answering systems, we focus our attention on dialogue system. For addressing the second research question, we investigate methods to incorporate domain knowledge present in the form of unstructured text. We propose a novel neural network model, which adds the additional textual knowledge using a separate LSTM focused on learning the importance of a domain-specific keyword. This model is trained end-to-end and achieves state-of-the-art performance on retrieval-based chatbots.

Answering Research Question 3

Devising methods in order to have a low-dimensional representation for entities and relations in a Knowledge Graph, is a well-researched topic. These dense embeddings are later used for downstream tasks such as relation linking, triple classification, or missing relation type. Word-embeddings on the other hand also is a well-researched topic and can successfully capture semantic relations between words. In order to answer research question 3, we propose an end-to-end model for task-oriented dialogue systems which is based on jointly embedding word and Knowledge Graph elements in the same vector space and use it in the downstream task of task-oriented dialogue generation with several novel loss functions. Empirical results suggest that using the proposed technique generates better task-oriented dialogues measured in terms of BLEU scores.

Answering Research Question 4

From task-oriented, knowledge grounded dialogue systems, in order to make the systems extendable to non-task oriented dialogues as well. In order to tackle that, we firstly create a dataset for Knowledge Graph-based, non-task oriented dialogues in the domain of football. Additionally, we also propose 2 end-to-end neural network-based models for the task of task and non-task oriented, generative dialogue systems. Firstly, we model the dialogue generation process using a knowledge copy method, where the model learns to copy facts from an in-memory knowledge structure conditioned on the question posed to the model. Henceforth, we propose another novel neural network architecture, where the model learns to answer by conditionally generating tokens from the corpus or using Knowledge Graph

relation information. Additionally, we also use a state-of-the-art pre-trained transformer model as input which aids in the final performance of the proposed architecture.

1.3.2 Publications

This section summarizes the scientific contributions made during the course of this thesis work.

Accepted Papers

- Mohnish Dubey, Debayan Banerjee, **Debanjan Chaudhuri**, and Jens Lehmann, EARL: joint entity and relation linking for question answering over Knowledge Graphs, International Semantic Web Conference(ISWC), 2018.
- Debayan Banerjee, **Debanjan Chaudhuri**, Mohnish Dubey, and Jens Lehmann, PNEL: Pointer Network based End-To-End Entity Linking over Knowledge Graphs, International Semantic Web Conference(ISWC), 2020.
- **Debanjan Chaudhuri**, Augustinus Kristiadi, Jens Lehmann, and Asja Fischer, Improving response selection in multi-turn dialogue systems by incorporating domain knowledge, Conference on Natural Language Learning (CoNLL), 2018.
- Firas Kassawat, **Debanjan Chaudhuri**, and Jens Lehmann, Incorporating joint embeddings into goal-oriented dialogues with multi-task learning, Extended Semantic Web Conference (ESWC), 2019.
- **Debanjan Chaudhuri**, Md Rashad Al Hasan Rony, Simon Jordan, and Jens Lehmann, Using a KG-Copy Network for Non-Goal Oriented Dialogues, International Semantic Web Conference (ISWC), 2019.
- **Debanjan Chaudhuri**, Md Rashad Al Hasan Rony, and Jens Lehmann, Grounding Dialogue Systems via Knowledge Graph Aware Decoding with Pre-trained Transformers, 2021.

Papers in Review

- Md Rashad Al Hasan Rony, **Debanjan Chaudhuri**, Rostislav Nedelchev, Asja Fischer, and Jens Lehmann, End-to-End Entity Linking and Disambiguation leveraging Word and Knowledge Graph Embeddings, 2021.

The rest of the thesis is organized as follows: In chapter 2, all the related concepts required to follow this thesis are mentioned. Chapter 3 summarizes all the previous works relating to human-machine interaction, especially using speech and text. Chapter 4.3.6 summarizes the methods proposed in this work to answer research question 1. The subsequent chapters, namely chapter 5.5.4, 6.4.3 describe novel model and methods proposed to answer research question 2 and 3, respectively. The final research question is studied in 7.6.2 and 8.5.3 and the different approaches and methods are described to answer it. The research is concluded in chapter 9.2.2.

Preliminaries

In this chapter, we summarize the related concepts required to follow the rest of this thesis. We firstly start with the basic concepts of natural language processing and different knowledge source descriptions. Eventually we will focus on more advanced concepts in natural language processing and deep learning

2.1 Natural Language Processing

Natural language processing (NLP) is a sub-field of linguistics and computer science concerned with the interactions between computers and human languages, in particular how to devise algorithms which can interpret and process natural, human language.

Language processing, although has deep significance in the field of linguistics; however, recent research in NLP focus mainly on several individual tasks relating to language processing. Text classification is one of such task, where the task of a model is to classify a piece of text, the latter could be short sentence(s) or larger documents. Text classification can be used for sentiment analysis, document classification and many other business cases. Another NLP task is sequential classification, where the model is suppose to tag or classify each token of a text piece. Such classification task is generally used for the task of entity span detection, part-of-speech identification, et cetera. Probabilistic language generation also comes under the umbrella of NLP as well. Here, a model generates words based on a context of words such as in language modelling or dialogue generation. These are mainly supervised tasks, however there can also be unsupervised tasks such as text clustering etcetera.

The standard process to solve a NLP task is to calculate the conditional distribution $p_{\text{data}}(Y|X)$ by learning from labelled dataset(s) which consists of pairs $x_i \in X$ and $y_i \in \mathcal{Y}$ (supervised learning); or the data distribution $p_{\text{data}}(X)$ by learning from unlabelled datasets consisting of samples $x_i \in X$ (unsupervised learning).

NLP tasks can be modelled using two methodologies 1) Rule based NLP and 2) Statistical NLP. Rule based NLP was based on the insights of language understanding as in[21]. Such approaches made use of a set of rules defined by formal grammar and a finite set of compositional rules and make predictions based on them. However, designing such rules is very time consuming, requires domain experts to design such rules and cannot be easily extended to new or real-world datasets.

Statistical approaches for understanding language and solving related tasks emerged, premised on the distributional hypothesis[22]. Given a dataset, domain experts devise features and its corresponding

extractors, for a particular task. Using these features and a suitable mathematical model the necessary decision planes can be learned automatically from the data. In spite of coming a step forward from rule-based systems, feature engineering is still limited in scope to the kind of features humans can come up with to extract the most out of data.

With recent advancements in neural network based machine learning models, most of NLP research is now focused on using neural networks for learning latent features from the dataset. With the introduction in embedding words into distributed vector spaces has seen rapid advancement in NLP research. More recently, with pre-trained transformer based models, NLP methodologies has reached new heights.

2.1.1 Knowledge Sources

As mentioned previously, world knowledge is present across several sources.

Unstructured Knowledge Sources

The whole web can be considered both structured and unstructured, but most of it is unstructured. As defined by [23], unstructured knowledge is a knowledge source which has no identifiable structure. From [24] Unstructured data is also described as data, that cannot be stored in rows and columns in a relational database. Storing data in an unstructured form without any defined data schema is a common way of filing in information. An example for unstructured data is a document that is archived in a file folder. Other examples are videos and images. The advantage of unstructured data is, that no additional effort on its classification is necessary. A limitation of this kind of data is, that no controlled navigation within unstructured content is possible.

2.1.2 Structured Knowledge Sources

Fully structured data follows a predefined schema. "An instance of such a schema is some data that conforms to this specification" [25]. Typical structured data sources are relational databases or Knowledge Graphs. Typical phase of designing a structural data source is firstly to design the schema and subsequently the content is created, populating the database. Knowledge graphs are also created and populated in the same way. Additionally Knowledge Graphs are also created semi-automatically using techniques such as relation linking and discovery.

Relational Databases

A relational database is a type of structured data source that stores and provides access to data points that are related to one another. As mentioned previously, relational databases are based on a relational model, an intuitive, straightforward way of representing data in tables. In such databases, each row in the table is a record with a unique ID called the key (primary or foreign). The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

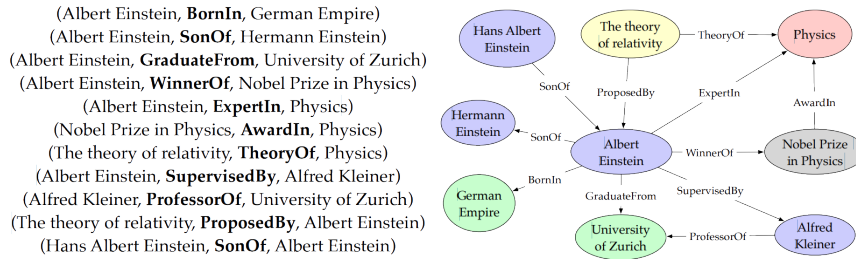


Figure 2.1: Knowledge Base and corresponding Knowledge Graph

Knowledge Graph

Definition 1 (Knowledge Graph) *Within the scope of this work, we define a Knowledge Graph as a labelled directed multi-graph. A labelled directed multi-graph is a tuple $KG = (V, E, L)$ where V is a set called vertices, L is a set of edge labels and $E \subseteq V \times L \times V$ is a set of ordered triples. The Knowledge Graph vertices represent entities and the edges represent relationships between those entities.*

A Knowledge Graph is a form of structured human knowledge [26]. It consists of facts in the form of entities, relationships and semantic descriptions. . Entities can be real-world objects and abstract concepts, relationships represent the relation between entities, and semantic descriptions of entities and their relationships contain types and properties with a well-defined meaning.

While Knowledge Graph and Knowledge Base are two terms used synonymously, there is a minor difference between the two. A knowledge graph can be viewed as a graph when considering its graph structure. When it involves formal semantics, it can be taken as a knowledge base for interpretation and inference over facts. Examples of knowledge base and Knowledge Graph are illustrated in 2.1. Knowledge can be expressed as a fact or triple, used interchangeably in the form of (**head**, *relation*, **tail**) or (**subject**, *predicate*, **object**) under the resource description framework (RDF), for example, (**Albert Einstein**, *WinnerOf*, **Nobel Prize**). It can also be represented as a directed graph with nodes as entities and edges as relations.

2.2 Neural Networks

From knowledge sources we would now shift our focus to explaining the machine learning methods used in this thesis. Neural networks(NN) are the most extensively used in recent machine learning models and specifically in this thesis. Different models of NNs have achieved state-of-the-art performance across various computer vision and NLP related tasks. In this section, we will give a brief overview of the fundamental building blocks of NN and then describe specific architectures and modules related to the task of NLP.

The simplest Neural Network model is a **feed-forward neural network** or **multi-layer perceptron** (MLP) which is a composition of multiple layers of perceptron with non-linear activation functions. Amongst them, the most simple one would be a single layer feed forward network defined as follows:

$$NN_{MLP1}(\mathbf{x}) = g(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}^2$$
$$\mathbf{x} \in \mathbb{R}^{d_{in}}, \mathbf{W}^1 \in \mathbb{R}^{d_{in} \times d_1}, \mathbf{b}^1 \in \mathbb{R}^{d_1}, \mathbf{W}^2 \in \mathbb{R}^{d_1 \times d_2}, \mathbf{b}^2 \in \mathbb{R}^{d_2}$$
 (2.1)

where $\mathbf{W}^1, \mathbf{b}^1$ and $\mathbf{W}^2, \mathbf{b}^2$ are the weight matrix and the bias for the first and second linear transformation, respectively. These are the parameters which are trained using techniques such as back-propagation, and g represents the nonlinear activation function. A simplified representation for a two-layer MLP, which then can be generalized to N layers would be:

$$NN_{MLP2}(\mathbf{x}) = \hat{y}$$
$$\mathbf{h}^1 = g^1(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)$$
$$\mathbf{h}^2 = g^2(\mathbf{h}^1\mathbf{W}^2 + \mathbf{b}^2)$$
$$\hat{y} = \mathbf{h}^2\mathbf{W}^3$$
 (2.2)

A layer is defined as the resulting vector from each linear transformation. The final layer, i.e., the outermost layer is called the *output layer* where a transformation is applied in order to get probabilities for several tasks. Generally, a softmax or sigmoid transformation is applied at the last layer. All layers apart from the output layer are called *hidden layers*. The output of one layer acts as an input to the next layer, which eventually produces the output Y . This process is called *forward propagation* and the non-linearity at each layer makes neural networks universal approximators. The non-linear function g , also called *activation function*, can be of many type and few of the typically used ones are:

- **Sigmoid:** It transforms the input \mathbf{x} to a value between $[0,1]$ and is defined as:

$$g(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}}} \quad (2.3)$$

- **Rectified linear unit (RELU)**[27]: It clips the negative input value to zero and is defined as:

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

- **Hyperbolic tangent function (Tanh):** It maps the output value to $[-1,1]$ and is defined as:

$$g(\mathbf{x}) = \frac{e^{2\mathbf{x}} - 1}{e^{2\mathbf{x}} + 1} \quad (2.5)$$

Gradients are computed one layer at a time in the reverse direction of the forward propagation by employing the chain rule of differentiation. The method of computing the gradients of the final loss function with respect to each weight, across all the layers is called backpropagation. Backpropagation has become a quintessential algorithm shared across several neural network architectures.

2.2.1 Deep Learning

After defining basic neural networks, we will now define the different neural network architectures that are generally used for NLP and computer vision tasks. Deep learning is the extension of neural network, where multiple hidden layers are used in the architecture in contrast to a single hidden layer for simple perceptron. Large proportions of this section contain material from Deep Learning[28].

Deep learning models, or neural networks are composite, parameterized functions which can be trained to output $\hat{y}_i \simeq y_i \in \mathcal{Y}$ when given $x_i \in X$; or generate samples $\hat{x}_i \simeq x_i \in X$. From a theoretical perspective, they can be seen as a family of probability distributions $p_{\text{model}}(\cdot, \theta)$, indexed by θ , which maps any configurations x to a real number estimating the true probability $p_{\text{data}}(y|x)$ (discriminative models), or in the case of generating samples, $p_{\text{data}}(x)$ (generative models).

A deep learning model can be seen as a function f which is arbitrary and is parameterized with θ which works as follows:

$$\hat{y}_i = f(x_i; \theta) \quad (2.6)$$

$$p_{\text{model}}(y_i|x_i; \theta) = \hat{y}_i \quad (2.7)$$

Here, \hat{y}_i is the output of the model given x_i as the input. We would want a value of the index θ which aligns $p_{\text{model}}(\cdot, \theta)$ as close as possible to p_{data} . That is, for an input vector x_i , we would want the model's output vector \hat{y}_i be a distribution over all possible labels, with y_i having the being assigned the highest probability, where (x_i, y_i) are samples generated from p_{data} . One way to achieve this is to use the principle of (conditional) *maximum likelihood estimation* or conditional MLE, defined as:

$$\theta^{\text{ML}} = \underset{\theta}{\operatorname{argmax}} p_{\text{model}}(Y|X; \theta) \quad (2.8)$$

which can be further resolved to:

$$\theta^{\text{ML}} = \underset{\theta}{\operatorname{argmin}} \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim \hat{p}_{\text{data}}} -\log p_{\text{model}}(\mathbf{y}|\mathbf{x}; \theta) \quad (2.9)$$

Hence we can find the most optimal set of parameters for our model by formulating it as an optimization problem. The **loss function** or objective function of this optimization objective would be the *negative log likelihood* (NLL) computed over samples (x_i, y_i) sampled from \hat{p}_{data} ¹, guided by the MLE principle.

We may also minimize the KL divergence², yields cross-entropy loss between the training data and the model distribution as the objective function.

We may replace Negative Log Loss with an arbitrary loss function $\mathcal{L}(f(x_i; \theta), y_i)$, which takes the model output over an instance, true (real) label of the instance, and model parameters as the input, and returns a score representative of the prediction errors of the model.

¹Note that in Eqn. 2.9, \hat{p}_{data} represents the empirical distribution based on sampled data at hand, since we have no direct access to the true data generating distribution p_{data} .

²<http://benlansdell.github.io/statistics/likelihood/>

Eqn 2.9 can then be generalized as:

$$\theta^* = \operatorname{argmin}_{\theta} \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim \hat{p}_{\text{data}}} \mathcal{L}(f(\mathbf{x}; \theta), \mathbf{y}) \quad (2.10)$$

$$= \operatorname{argmin}_{\theta} \frac{1}{m} \sum_{i=1}^m \mathcal{L}(f(x_i; \theta), y_i) \quad (2.11)$$

where m is the number of labelled samples in a given dataset.

The surface of the loss function may be highly complex and lie in a high dimensional space for many deep learning model (number of dimensions as high as 350 million[29]). This renders grid search or random search based solutions computationally intractable. The intractability is further compounded upon by the fact that the actual value of the loss surface at any point would require computing the loss function over the entire dataset.

Most commonly **gradient descent** is used to optimize the objective function - \mathcal{L} . The gradient of a function $f(x)$ is the vector containing all the partial derivatives, denoted by $\nabla_x f(x)$, and critical points are points where every element of the gradient is equal to zero. To minimize \mathcal{L} , we would like to find the direction in which \mathcal{L} decreases the fastest, at any point in the parameter space θ , which is to say

$$\min_{\mathbf{u}, \mathbf{u}^T \mathbf{u} = 1} \mathbf{u}^T \nabla_{\theta} \mathcal{L}(f(\mathbf{x}; \theta), \mathbf{y}) \quad (2.12)$$

$$\min_{\mathbf{u}, \mathbf{u}^T \mathbf{u} = 1} \|\mathbf{u}\|_2 \|\nabla_{\theta} \mathcal{L}(f(\mathbf{x}; \theta), \mathbf{y})\|_2 \cos \phi \quad (2.13)$$

where \mathbf{u} is a unit vector representing the direction in which we wish to move, and ϕ is the angle between \mathbf{u} and the gradient. Since the magnitude of any unit vector ($\|\mathbf{u}\|_2$) is 1, the gradient term isn't influenced by \mathbf{u} , Eqn. 2.13 can be resolved to $\min_{\mathbf{u}} \cos \phi$. Hence moving in opposite direction of $\nabla_{\theta} \mathcal{L}$ minimizes \mathcal{L} the most. This leads to the following update step:

$$\theta' = \theta - \nabla_{\theta} \mathcal{L} \quad (2.14)$$

where, is the **learning rate**, determining the magnitude of the update. This update step, if computed indefinitely (or multiple number of times) will converge the model to a local minima[30, Chapter 9.3.1].

This optimization scheme also has some problems. Firstly, the loss surface of any neural network may have several local minimas and saddle points [31]. Models trained based on (vanilla) gradient descent can get stuck (converge) on either of them. Furthermore, computing the loss over the entire dataset is computationally expensive.

An alternative learning method - **stochastic gradient descent** (SGD) is thus more commonly used to train neural models. Instead of computing the loss over the entire dataset, training under SGD involves sampling a *mini-batch* of examples $B = x_1, \dots, x_m$ drawn uniformly from the training set. This is done treating the gradient as an expectation, which can be approximately estimated using a small set of samples. Sampling mini batches from the dataset for each update step ensures a much reduced training time. Further, this stochastic nature introduces variations to the loss function and consequentially the gradients, reducing convergence time and possibly converging potentially to a better minima (i.e. with lower loss values).

For some other methods of training a deep neural network one may look at [32–36] and interested

readers may refer [28].

We can now sequentialize steps to building a deep learning algorithm into (i) a labelled dataset denoted by \hat{p}_{data} , (ii) a loss function $\mathcal{L}(f(x_i; \theta), y_i, \theta)$, (iii) an optimisation procedure (e.g. stochastic gradient descent), and (iv) a model. In the rest of this section, we discuss some common model components and architectures relevant to this work.

2.2.2 Deep Learning Models

This section describes different deep learning model architectures or neural networks that are used, and are usually trained using the gradient learning methods as mentioned above.

Given the objective of modelling \hat{p}_{data} , we would want to devise functions which (i) are parameterised, and receptive to change in parameters, (ii) are able to learn complex representations of input data by learning simpler representations and or combining them, and (iii) are non-linear, since multiple linear transformations when combined can be represented by one linear transformation rendering multiple layers superfluous. Being able to train them using gradient descent requires that these functions must be (iv) are end-to-end differentiable.

Let us consider the following linear transformation:

$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{x} + \mathbf{b} \quad (2.15)$$

where $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{W} \in \mathbb{R}^{N \times M}$ and $\hat{\mathbf{y}}, \mathbf{b} \in \mathbb{R}^M$. Treating \mathbf{W} and \mathbf{b} as parameters, we can represent it with the following function:

$$f_{\text{linear}}(\mathbf{x}; \theta) = \mathbf{W}_{\theta}\mathbf{x} + \mathbf{b}_{\theta} \quad (2.16)$$

where $f_{\text{linear}} : \mathbb{R}^N \mapsto \mathbb{R}^M$ is a function parameterized which can compute an M -dimensional vector given N -dimensional vectors as inputs. Present in existing literature as a **linear regressor**, this function can be trained to weigh the N different features of input differently and produce an M dimensional vector. For classification over an M dimensional label space Y , we can interpret f_{linear} 's output as a probability distribution over the M classes, and can choose the class with the highest score as the model's predicted output. We can alternatively normalize, and interpret its output ($\hat{\mathbf{y}} = f_{\text{linear}}(\mathbf{x}; \theta)$) as a distribution $p_{\text{model}}(\mathbf{y}|\mathbf{x})$.

A *linear regressor* thus satisfies most of the requirements except for non-linearity. To incorporate non-linearity, we can simply augment $\hat{\mathbf{y}}$ s.t. $\hat{\mathbf{y}} = g \circ f$, where g is a non-linear function, known as the **activation function**. The most common activation function, namely the *sigmoid function* ($\sigma(z)$), defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2.17)$$

is a bounded, differentiable and monotonic function $\sigma : \mathbb{R} \mapsto (0, 1)$. Its generalisation to multi-dimensional domain and range is known as the **softmax function**.

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^M e^{z_j}} \quad (2.18)$$

Softmax is most often used as the activation function over the final outputs of a model, given that it can interpret an n -dimensional vector as a probability distribution over n classes.

Other common choices of activation functions include the hyperbolic tangent function $\tanh : \mathbb{R} \mapsto (-1, 1)$, and the more commonly used Rectified Linear Unit $g(z) = \max\{0, z\}$ or (ReLU).

This constitutes the building block of neural networks, namely - a **feed forward layer**, composed of a linear function $f_{\text{linear}}(\mathbf{x}; \theta)$, and an activation function. We may sequentially place these layers to constitute a model that can learn a complex non-linear, parameterized function, which can be trained to fit a probability distribution. All but the final layers are called *hidden layers*, and layer whose outputs constitute the model's output is called *output layer*. The output of one layer acts as the input to the next, and the process in which these inputs are passed across layers is called *forward propagation*. Correspondingly, during learning, we compute the gradients of the model one layer at a time. Using the chain rule of differentiation as used in calculus. Only the gradients of the layer directly above the current one affects its gradients, making the process of computing the gradients computationally efficient. The process in which these gradients are computed across each layer (from top to bottom) is called *backpropagation*.

For example, a two layered feed forward network f_{mlp} with two hidden layers can be denoted as follows:

$$\begin{aligned}\mathbf{h}_1 &= g_1(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_{a1}) \\ \mathbf{h}_2 &= g_2(\mathbf{W}_2 \mathbf{h}_1 + \mathbf{b}_2) \\ \hat{\mathbf{y}} &= \mathbf{W}_{\text{out}} \mathbf{h}_2 + \mathbf{b}_2 \\ p_{\text{model}}(y|\mathbf{x}) &= \text{softmax}(\hat{\mathbf{y}})\end{aligned}\tag{2.19}$$

Some commonly used layers are as follows:

Feed forward, fully connected layers are not suitable for all tasks. When making predictions over time-dependent, sequential input of length l and dimensions n with an m dimensional label space a single layer feed forward network would require $l \times n \times m + m$ parameters. However if instead we use a recurrent model, where the hidden layers are time-dependent on each other and hence is suitable for such sequential tasks.

Moreover, simple feed forward networks ignore structure of the input while making predictions, as the input needs to be a *flattened* array of numbers. For an image of dimension m rows and n columns, for example, a linear array of $m \times n$ needs to be input, losing neighbouring pixel information for the network. than not. In the subsequent sections we will give a brief overview of deep learning layers used for different tasks.

Recurrent Layers

Recurrent neural networks or RNNs[37] are a set of neural networks architectures which can handle data sequential in nature $\langle \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)} \rangle \in X$. For example, in case of a sentence, the last word of the sentence is probabilistically dependent on the first and other subsequent words as well, to form a latent representation of the whole sentence, the models must take into account the whole sequence of words.

Recurrent models share parameters when processing inputs at different time-steps and learns the parameters of the model θ in a time-dependent manner.

In other words, they process the inputs repeatedly - applying the same parameterised transformation over each vector $(\mathbf{x}^{(t)})$, processing each vector across all time-steps. This enables the model to generalize over sequences of multiple different lengths as well.

Furthermore, to be able to make inferences across different temporal position, recurrent models pass a fixed length vector across time-steps, often referred to as the *hidden state*, or the *memory* updating the parameters at each time-step. We can define them using the following equation:

$$\mathbf{h}^{(t)} = g_{\text{rnn}}(\mathbf{W}\mathbf{x}^{(t)} + \mathbf{U}\mathbf{h}^{(t-1)} + \mathbf{b}) \quad (2.20)$$

Where, g being the activation function (e.g. tanh), $\mathbf{x}^{(t)}$ is the t^{th} input in the sequence, $\mathbf{h}^{(t-1)}$ is the hidden state corresponding to the $t-1^{\text{th}}$ input. The weight matrices \mathbf{W} , \mathbf{U} and the bias \mathbf{b} parameterize this layer (θ for this model), and are shared across time-steps.

One shortcoming of these models is lack of sense of directionality. The flow of information in Eqn. 2.20 is left-to-right which might be sub-optimal when making decisions involving elements across the sequences, or right-to-left scripts (e.g. Hebrew sentences). A **bidirectional RNN** (or BiRNN) aims to partially rectify these drawbacks by using an RTL (right to left) RNN alongwith the conventional LTR (left to right) one, and combining their hidden states at each time to produce the final summary vector, as follows:

$$\begin{aligned} \mathbf{h}_l^{(t)} &= g_{\text{rnn}}(\mathbf{W}_l\mathbf{x}^{(t)} + \mathbf{U}_l\mathbf{h}_l^{(t-1)} + \mathbf{b}_l) \\ \mathbf{h}_r^{(t)} &= g'_{\text{rnn}}(\mathbf{W}_r\mathbf{x}^{(t)} + \mathbf{U}_r\mathbf{h}_r^{(t+1)} + \mathbf{b}_r) \\ \mathbf{h}^{(t)} &= [\mathbf{h}_l^{(t)}, \mathbf{h}_r^{(t)}] \end{aligned} \quad (2.21)$$

$\mathbf{W}_l, \mathbf{W}_r, \mathbf{U}_l, \mathbf{U}_r, \mathbf{b}_l, \mathbf{b}_r$ being the parameters of the layer and shared across time-steps.

There are several other shortcomings of these recurrent architectures namely: vanishing or exploding gradients, information loss across long time-steps which are solved using methods proposed by [38–40]. We discuss them briefly, below.

Long Short-Term Memory (LSTM) Layers

[41] proposed a Long Short-Term Memory (LSTM) cell, an augmented recurrent cell, which can explicitly retain information for even sequences which are long. An LSTM cell transfers not just the last layer's hidden output $\mathbf{h}^{(t)}$ across time-steps, but an internal cell state $\mathbf{c}^{(t)}$ on which the cell's hidden outputs are dependent on. Further, using, removing, and adding information to the cells is managed by the means of three gating mechanisms implemented using: *output*, *forget*, and *input* gates. For the sake of brevity, we omit an in depth explanation of each gate, and refer interested readers to an excellent explanation of the topic -[42], and directly mention the equations which characterise an LSTM cell:

$$\begin{aligned} \mathbf{f}^{(t)} &= g_g(\mathbf{W}_f\mathbf{x}^{(t)} + \mathbf{U}_f\mathbf{h}^{(t-1)} + \mathbf{b}_f) \\ \mathbf{i}^{(t)} &= g_g(\mathbf{W}_i\mathbf{x}^{(t)} + \mathbf{U}_i\mathbf{h}^{(t-1)} + \mathbf{b}_i) \\ \mathbf{o}^{(t)} &= g_g(\mathbf{W}_o\mathbf{x}^{(t)} + \mathbf{U}_o\mathbf{h}^{(t)} + \mathbf{b}_o) \\ \mathbf{c}^{(t)} &= \mathbf{f}^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ g_c(\mathbf{W}_c\mathbf{x}^{(t)} + \mathbf{U}_c\mathbf{h}^{(t-1)} + \mathbf{b}_c) \end{aligned} \quad (2.22)$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \circ g_h(\mathbf{c}^{(t)}) \quad (2.23)$$

where $\mathbf{f}^{(t)}$, $\mathbf{i}^{(t)}$ and $\mathbf{o}^{(t)}$ represent the different gates which dictate the edit operations on the cell state $\mathbf{c}^{(t)}$ as depicted in Eqn. 2.22. Furthermore, g_g are sigmoid activation functions, g_c , g_h are *hyperbolic tangent* activations, \circ represents the Hadamard product, and W_* , U_* , and b_* are parameters, which are time-invariant. Akin to the RNN layer, a layer of LSTM can be also modified for bidirectionality using bi-directional models and concatenating their outputs.

Different variations of these recurrent units have been proposed, including Gated Recurrent Unit (GRU) [43], Minimal Gated Unit (MGU) [44].

Pointer Networks

For the definitions below we borrow heavily from the original work on Pointer Networks [45].

Sequence-to-Sequence Model

Given a training pair, $(\mathcal{P}, C^{\mathcal{P}})$, the sequence-to-sequence [46] model computes the conditional probability $p(C^{\mathcal{P}}|\mathcal{P}; \theta)$ using a parametric model (an RNN with parameters) to estimate the terms of the probability chain rule

$$p(C^{\mathcal{P}}|\mathcal{P}; \theta) = \prod_{i=1}^{m(\mathcal{P})} p(C_i|C_1, \dots, C_{i-1}, \mathcal{P}; \theta) \quad (2.24)$$

Here $P = \{P_1, \dots, P_n\}$ is a sequence of n vectors and $C^{\mathcal{P}} = \{C_1, \dots, C_{m(\mathcal{P})}\}$ is a sequence of $m(\mathcal{P})$ indices, each between 1 and n . The parameters of the model are learnt by maximizing the conditional probabilities for the training set, i.e.

$$\theta^* = \operatorname{argmax}_{\theta} \sum_{\mathcal{P}, C^{\mathcal{P}}} \log p(C^{\mathcal{P}}|\mathcal{P}; \theta) \quad (2.25)$$

where the sum is over training examples.

Context Based Input Attention

The normal sequence-to-sequence model produces the output sequence $C^{\mathcal{P}}$ using the fixed dimensional state of the recognition RNN at the end of the input sequence \mathcal{P} . This constrains the amount of information and computation that can flow through to the generative model. The attention model of [47] mitigates this problem by augmenting the encoder and decoder RNNs with an additional neural network that uses an attention mechanism over the entire sequence of encoder RNN states. For notation purposes, let us define the encoder and decoder hidden states as (e_1, \dots, e_n) and $(d_1, \dots, d_{\uparrow(\mathcal{P})})$, respectively. For the LSTM RNNs, we use the state after the output gate has been component-wise multiplied by the cell activations. We compute the attention vector at each output time i as follows:

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \dots, n) \quad (2.26)$$

$$a_j^i = \operatorname{softmax}(u_j^i) \quad j \in (1, \dots, n) \quad (2.27)$$

$$d_i = \sum_{j=1}^n a_j^i e_j \quad (2.28)$$

where softmax normalizes the vector u^i (of length n) to be the "attention" mask over the inputs, and v , W_1 , and W_2 are learnable parameters of the model. In all our experiments, we use the same hidden dimensionality at the encoder and decoder (typically 512), so v is a vector and W_1 and W_2 are square matrices. Lastly, d_i^+ and d_i^- are concatenated and used as the hidden states from which we make predictions and which we feed to the next time step in the recurrent model.

Ptr-Net

A modification of the attention model that allows us to apply the method to solve combinatorial optimization problems where the output dictionary size depends on the number of elements in the input sequence: The sequence-to-sequence model softmax distribution over a fixed sized output dictionary to compute $p(C_i|C_1, \dots, C_{i-1}, \mathcal{P})$. Thus it cannot be used for problems where the size of the output dictionary is equal to the length of the input sequence. To solve this problem one can model $p(C_i|C_1, \dots, C_{i-1}, \mathcal{P})$ using the attention mechanism of as follows:

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i^-) \quad j \in (1, \dots, n) \quad (2.29)$$

$$p(C_i|C_1, \dots, C_{i-1}, \mathcal{P}) = \text{softmax}(u_i^i) \quad (2.30)$$

where softmax normalizes the vector u_i^i (of length n) to be an output distribution over the dictionary of inputs, and v , W_1 , and W_2 are learnable parameters of the output model. Here, we do not blend the encoder state e_j to propagate extra information to the decoder, but instead, use u_j^i as pointers to the input elements. In a similar way, to condition on C_{i-1} we simply copy the corresponding $P_{C_{i-1}}$ as the input.

Transformers

In order to handle the problem with long-term dependency, recently a self-attention based model was proposed for sequence-to-sequence tasks called transformers [48]. In the case of transformers, the encoder and decoder blocks has similar architecture with only difference of having a masked attention module for the decoder. Self-attention is where the final output or representation for a token is created based on all the other tokens in a sequence. It is calculated using three vectors namely: query vector (Q), key vector (K) and value vector (V). The final attention is calculated as:

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.31)$$

Where, d_k is the dimension of the vectors. The model is trained in a similar fashion as other sequence-to-sequence models (RNN based) by predicting the next word at a time by shifting the decoder words. However, in the case of the transformers, the future words in the decoder are masked (with zeros) to avoid the decoder the see the future output.

A lot of further research [49] [50] has successfully trained such transformer architecture in an unsupervised manner by training the transformer as a language model but by replacing 15% of the words in the sequence by a mask (mask language modelling) and also to predict whether the next sentence is really a next sentence in a document or not (next sentence prediction). These pre-trained transformer models has been very beneficial for other downstream tasks such as question-answering, reading comprehension, sentence classification and so on.

Convolutional Layers

Convolutional Neural Networks (CNNs) as proposed by [51] are neural layers specialized in processing grid-like data structures such as images (2-D grid), or even sequential data which can be modeled as a 1-D grid. As the name suggests, these layers involve the convolutional operation using parameterized, multi-dimensional kernels. These kernels are learnable parameters, used to compute a weighted average over subset of input values to produce one value as an output representation: $h_j^{\text{conv}} = W_k \mathbf{x}_{j-m/2:j+m/2}$ where W_k is a kernel of m dimensions, and \mathbf{x} is a 1-D input of n . By sliding this kernel over the length of \mathbf{x} , we can compute an output representations of $n - m + 1$ dimensions. This operation of *sliding* the kernel over an input is called the *convolution operation*, and can be generalised for multi-dimensional inputs and kernels.

When an input is passed through a feed-forward layer, each output value m is influenced by each input value n correspondingly weighted by $W_{m,n}$, a scalar parameter. When processing structured inputs, this is disadvantageous because of the increased number of parameters, as well as particularly difficult for the model to be spatially aware. Convolutional layers instead have *sparse interactions* as the inputs are convolved, or multiplied with kernels that are significantly smaller in size. Furthermore, given that we slide the same kernel across the inputs, as opposed to having different parameters corresponding to each term in the output vector, the kernel parameters are shared. This invariance helps to train the kernels to find a specific feature over the inputs, for instance an edge in an image, or in less abstract layers: a geometric shape, or a written character, implicitly. In practice, we have multiple kernels (with independent parameters in each) produce multiple output representations of the input corresponding to the presence of feature each kernel is trying to find, giving the kernel their other commonly used name - a feature map. Note that we use the term feature not to denote human-interpretable features but rather a latent, high-dimensional representation in a space with little co-relation to a human interpretable concept.

These multiple outputs are then passed through activation functions, and finally through a **pooling layer** which replace the output of the net at a certain location with a summary statistic of the nearby outputs[28, Chapter 9.3], making the outputs become more invariant to slight variations in the input. All the outputs from all the convolution filters are finally *flattened* and passes on to a fully-connected layer and finally all the network learns with traditional backpropagation based method as explained in the previous sections.

A convolution layer, thus, is composed of (i) a convolution operation, (ii) an activation, and (iii) a pooling operation (could be average, or max pooling). Models composed primarily of convolution layers have shown tremendous performance over multiple tasks, primarily, but not restricted to the domain of computer vision [52–54]. Recently, their use for NLP tasks has been explored, and has shown promising results[55, 56] as well.

Time Distributed Feed Forward Layers

A time distributed feed-forward layer is fully connected layer applied on one element of the input sequence at a time. Intuitively, this can be imagined as a recurrent architecture without the hidden states. We can describe them by the following equations:

$$\hat{\mathbf{y}}^{(t)} = g(W\mathbf{x}^{(t)} + \mathbf{t}) \quad (2.32)$$

which we can alternatively represent as:

$$\hat{\mathbf{y}}^{(t)} = g(f_{\text{mlp}}(\mathbf{x}^{(t)}; \theta)) \quad (2.33)$$

In practice, we use this to classify tokens (as explained below), and more commonly to embed input word tokens into a distributed, low-dimensional vector space, traditionally used as the first layer in neural network models involved in natural language processing.

The latter, known as the **embedding layer** has no explicit activation function.

We discuss the use and the process of training embedding layer at length in the next section - Sec. 2.3.1.

These layers conclude the background neural network architectures that will be used in this thesis, in the later sections we will explain how these architectures for different natural language processing tasks.

Since, most of these thesis focus on using recurrent architectures, we will focus mostly using recurrent architectures and also transformers in the next parts of this section.

2.2.3 Deep Learning Architectures for NLP Tasks

A natural language processing task $T(Y, p_{\text{data}}(Y), p_{\text{data}}(Y|X))$, for a particular domain $D(X, p_{\text{data}}(X))$ can be broadly classified into four categories based on the type of textual understanding it requires. In this subsection, we will describe those NLP categories.

Note that, we will use f_{RNN} as an abstract function which may be composed of multiple recurrent layers, with or without bi-directionality, have any activation function, and may use any recurrent cell including RNNs, LSTMs et cetera. Similarly, f_{MLP} may be used to denote one feed-forward layer, or a combination of multiple such layers, with some activation function applied across each layer.

Sentence Level Classification

For the task of sentence level or sequence level classification, given a sequence of word tokens $x_i = x_i^{(1)} \dots x_i^{(T)}$, a model should classify the sequence x_i into one of n classes. Common examples of such tasks include sentiment classification, topic classification, document classification.

In order to classify the sentence or sequence, firstly the sentence is encoded using a word embedding layer (explained later) and then passed on to a RNN or CNN module to get a final representation of the whole sentence. Thereafter, a final classifier is added after this encoded output which creates a distribution over a label space. We cumulatively refer to the layers which create an encoded representation of the text as *encoder*.

In practice the encoder consists of the aforementioned embedding layer (or a time distributed feed forward layer), and a recurrent layer (or convolutional). The following equations provide an abstract summary in case of a RNN based encoder:

$$\mathbf{x}_{\text{emb}}^{(t)} = f_{\text{emb}}(\mathbf{x}^{(t)}; \theta_{\text{emb}}) \quad (2.34)$$

$$\mathbf{h}_{\text{enc}}^{(t)} = f_{\text{rnn}}(\mathbf{x}_{\text{emb}}^{(t)}; \theta_{\text{emb}}) \quad (2.35)$$

$$\hat{\mathbf{y}} = f_{\text{mlp}}(\mathbf{h}_{\text{enc}}^{(T)}; \theta_{\text{mlp}}) \quad (2.36)$$

where f_{emb} and f_{rnn} collectively constitute the *encoder*, and f_{mlp} is called the classifier which takes the *final* hidden state of the RNN as the sequence's summary. For a convolutional based method, the final encoded representation is obtained by concatenating the mean and max pooled (across time) representations of the hidden states $\mathbf{h}_{\text{enc}}^{(1)} \dots \mathbf{h}_{\text{enc}}^{(T)}$ along with the final state.

Token Level Classification

Unlike sentence level, for token level classification a model should assign a label in one of n classes to each token in a sentence rather than the entire sentence. Common examples of which include Part of Speech tagging, named entity recognition.

In order to tackle it, the previously mentioned network can be replaced by using a time distributed feed forward layer (See Sec. 2.2.2), as opposed to a regular MLP. We can thus modify Eqn. 2.36 as follows:

$$\hat{\mathbf{y}}^{(t)} = f_{\text{mlp}}(\mathbf{h}_{\text{enc}}^{(t)}; \theta_{\text{mlp}}) \quad (2.37)$$

This modification allows us to make prediction corresponding to each hidden state (representing a summary of the sequence up-to that point).

Interestingly, by keeping the label space same as the feature space, we can train the model to predict the next word given the sequence so far. This task is referred to as *language modelling*

The neural architecture defined here, and above (2.34-2.36) are collectively referred to as an **encoder-classifier** network.

Sequence Generation

Instead of classifying a given sequence into one of n classes, some task might involve generating text given a particular input. Some examples include generating captions given an image, or generating sentences from data.

Neural approaches to solve this task introduce the concept of a **conditional recurrent layer**. So far we've seen a recurrent layer as a function which encodes an input sequence's elements by the means of the cell state. Instead, we can think of it as a generative model which generates a hidden state vector *conditioned* on the input.

For the case of sequence generation, the conditional RNN is introduced which is conditioned on the encoded output of an encoder RNN and is trained to generate a sequence of words given the input query.

The task of sequence generation follows of step of creating a fixed length *context* vector given an input from an arbitrary feature space, and then using it to condition a recurrent layer to generate the desired sequence. The latter (combination of an RNN and a time distributed feed forward layer) is referred to as the *decoder* of the model. As above, we refer to the part of the model which encodes the given input to a fixed length context vector as the encoder. The resultant model can be described by the following equations:

$$\mathbf{h} = f_{\text{enc}}(\mathbf{x}; \theta_{\text{enc}}) \quad (2.38)$$

$$\mathbf{s}^{(t)} = f_{\text{rnn}}(\mathbf{h}, \mathbf{s}^{(t-1)}; \theta_{\text{rnn}}) \quad (2.39)$$

$$\hat{\mathbf{y}} = f_{\text{mlp}}(\mathbf{s}^{(t)}; \theta_{\text{mlp}}) \quad (2.40)$$

where f_{enc} is an arbitrary encoder, and f_{rnn} is conditioned on the encoder's output \mathbf{h} .

Sequence Transduction

Tasks which fall under this classification involve trying to predict a sequence given another. Typical examples include machine translation, dialogue generation et cetera, wherein we wish for the model to generate a sequence having the same semantic meaning, but in a different language, or paraphrasing wherein we wish for the model to rephrase one or many sentence keeping their semantic meaning intact.

To solve this, we can modify the network above, especially the decoder to generate a token at every time stamp conditioned on the given input query and the previously generated output.

That is to say, that we can replace f_{enc} in Eqn. 2.38 with a combination of an embedding layer (explained later) and a recurrent layer. Here, we can use both, the last hidden state of the encoding RNN, or its pooled summary (across time) as the context vector. These models are called encoder-decoder, or sequence-to-sequence networks.

2.3 Representing Language as Dense Vectors

So far we have discussed about how to handle different neural network architectures that can handle several input types for different tasks. However, one of the biggest challenges for natural language processing is to represent the text or words of the language which the neural network or machine learning model can understand.

Before the neural network based NLP models, words (or tokens) were represented as one-hot encoded vectors or n-gram based methods. However, they suffered from the curse of dimensionality. Later on, dense represented vectors were used to represent the words where were learned in an unsupervised way. More recently, pre-trained transformer based models are used instead of word-embeddings which empirically performed significantly better across all NLP tasks.

In this section we will explain the different embedding methods for encoding natural language. We will furthermore also discuss techniques proposed for encoding Knowledge Graph elements (entity and relations) using similar low-dimensional Knowledge Graph embeddings.

2.3.1 Word Embeddings

As mentioned before, instead of assigning a unique representation of each word using one-hot encoding, and explicitly modeling the relations between them, in a distributed representation scheme, each word is assigned a vector in a shared space. The meaning of the words and their relations to others are captured by the activations in the vector and the similarity between them[57, Ch. 10.4]. This results in words with similar meaning are close to each other in the distributed vector space.

Numerous approaches exist for designing and training word embeddings model. Notably, [58] propose one of the most popular approach for training word embeddings namely, skip-gram models. Skip-gram models are trained to predict the context given a target word, i.e. words surrounding the

target words, using unsupervised techniques. This is done by optimising the following loss function:

$$L_{\text{SGNS}} = -\frac{1}{|C|} \sum_{t=1}^{|C|} \sum_{-C \leq j \leq C, j \neq 0} \log p_{\text{model}}(x^{(t+j)} | x^{(t)}) \quad (2.41)$$

where $x^{(1:T)}$ is a sequence sampled from the training data. In practice $x^{(t)}$ represents the t^{th} word occurring in the sequence, as an index of the vocabulary V (the feature space for the task). Using this, we can then define a skip-gram model as:

$$\mathbf{x}_{\text{emb}}^{(t)} = \mathbf{W}_{\text{emb}} \mathbf{x}^{(t)} \quad (2.42)$$

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_{\text{context}} \mathbf{x}_{\text{emb}}^{(t)}) \quad (2.43)$$

$$p_{\text{model}}(\mathbf{x}^{(t+j)} | \mathbf{x}^{(t)}) = \hat{\mathbf{y}} \quad (2.44)$$

where \mathbf{W}_{emb} is a linear layer (without a bias or an activation) with $|V|$ input dimensions and an arbitrary hidden dimension (typically 100-300), also known as the **embedding layer**, and correspondingly $\mathbf{W}_{\text{context}}$ is another linear layer known as the context layer which maps the transforms the hidden space to the output space.

Here the trained embedding layer (or the embedding matrix) is of particular interest, since it transforms a word index to a low-dimensional, dense representation space and can be used for later downstream tasks. For the neural architectures explained previously, this embedding layer is used as f_{emb} whose main job is to represent the tokens as vectors to be used across several downstream tasks such as sentiment classification, language modelling, dialogue generation, et cetera.

2.3.2 Pre-trained Transformers

As explained previously pre-trained transformer models, trained on different tasks can also be used as embeddings for several other downstream tasks. Such models are BERT [49], Transformer-XL [59], XLM.

Along with the objectives of masked language modelling and next-sentence prediction, another novelty in the case of these pre-trained transformer models is they use a positional encoding added to the traditional word embedding model which denotes the position of a word in a sequence.

These pre-trained model has improved upon almost all NLP downstream tasks compared to using traditional word embeddings. Such pre-trained models outputs an embedding for the whole sentence corresponding to a special [CLS] token and also for each token in a sequence. The former can be used for sentence level classification tasks while the latter is used for token or sequence level classification tasks.

2.3.3 Knowledge Graph Embeddings

Similar to natural language words, many studies have also focused on getting dense vector representations for Knowledge Graph entities and relations as well. Such embeddings can also capture the semantics of KG elements and can be used for downstream tasks such as relation linking, knowledge graph completion.

Representation Space: Most works literature mainly focuses real-valued point-wise space

including vector, matrix and tensor space, eg, TransE [60] while other kinds of space such as complex vector space eg, RotateE [61] Gaussian space eg, TransG [62] and manifold eg, Dihedral [63] are utilized as well.

Scoring Function: The scoring function is used to measure the plausibility of facts in a Knowledge Graph, which is also referred to the energy function in case of energy-based learning framework. Energy-based learning aims to learn the energy function $E_\theta(x)$ parameterized by θ taking x as input, and to make sure true facts (positive samples) have higher scores than false facts (negative samples). In this thesis, the term of scoring function is adopted for unification. There are two typical types of scoring functions, i.e., distance-based [60] and similarity-based [64] functions. Distance-based scoring function measures the plausibility of facts by calculating the distance between entities, where additive translation with relations as $h + r \approx t$ is widely used. Semantic similarity-based scoring measures the plausibility of facts by semantic matching, which usually adopts multiplicative formulation, i.e., $h^T M_r \approx t^T$, to transform head entity near the tail in the representation space.

Encoding Models: Several models are being proposed that encode the interactions of entities and relations through specific model architectures, including linear/bilinear models, factorization models, neural networks. Linear models formulate relations as a linear/bilinear [60] mapping by projecting head entities into a representation space close to tail entities. Factorization [65] aims to decompose relational data into low-rank matrices for representation learning. Neural networks [66] encode relational data with non-linear neural activation and more complex network structures. Within neural networks the use of convolutional neural networks [67], recurrent neural networks [68], Transformers [69] and Graph Neural Networks [70] are common.

TransE

TransE [60] is one of mostly used KG embeddings (for downstream tasks) which represents entities and relations in d-dimension vector space, i.e., $\mathbf{h}, \mathbf{t}, \mathbf{r} \in \mathbb{R}^d$, and makes embeddings follow the translational principle $h + r \approx t$.

In the case of **PNEL**, one of the model proposed in this thesis, we utilise PyTorch-BigGraph [71] translation embeddings with the dot function as a comparator. Since the number of entities in our chosen Knowledge Graph of Wikidata is high (74 millions) a suitable comparison function has to be chosen to be able to discriminate between embeddings in a crowded space. As opposed to cosine, dot product considers both angle and length of overlap and hence is more discriminative in case of larger datasets.

Related Work

In this chapter, we discuss the different previous works relating to human-machine interactive systems, specifically text-based systems for interaction. We will first describe different previous works on human-machine interactive systems and proceed depth-first into interactive systems that can be realized as a question-answering system or as a dialogue system with additional contextual information. Previous methods and systems relating to question answering and dialogue systems are briefed upon here.

3.1 Human Machine Interactive Systems

As mentioned previously, human machine interactive systems can be realized using several methodologies [72] which are summarized below

3.1.1 Computer Vision Based

Vision based HMIs are mostly focused on gesture or motion detection and analysis. These systems has several applications ranging from automobile control to gaming. It is a well-research topics and various models and systems are proposed for gesture and motion detection [73] [74] [75].

3.1.2 Laser Based HMI

Laser based HMI can be used in conjunction with vision based systems or can also be developed as standalone architectures. By connecting multiple Infra Red(IR) LEDs with IR sensors, one can create a LED-based Multi-touch Sensor which are used in a interactive LCD screen [76] which has several potential applications. Laser based models can also be used to build virtual keyboard [77] and recently also has found usage for building augmented reality [78] systems.

3.1.3 Bionic Technology Based

Bionic HMI systems are influenced from medical based applications. Such systems can built using Electroencephalography(EEG) for tracking the electrical activity in the brain [79] [80] [81]. Also, some BCI systems are based on Electrocardiogram (ECG), Electromyography (EMG) [82], or Electrooculography (EOG) based [83]. Some systems are also modeled by combining EMG, EOG and EEG together.

3.1.4 Speech Based HMI

Last, and the most commonly used HMI are speech based, where the machine interacts with the user using speech interfaces [84] [85] [86].

Speech based human-machine interactive systems, which are the most widely used method for HMI has been researched for a long time. Initial speech based systems were restricted to command based systems where the user interacts with the system using simple commands with restricted vocabularies [87]. Later systems were based on interactive voice responses (IVR) [88]. IVR systems are traditionally built using Dual-Tone Multi-Frequency signaling (or DTMF), but recently voice recognition technologies are also in place. More recently, speech-based HMI's are very much available in every smartphones as voice-enabled personal assistants (eg. siri, cortana). Such systems are built using a speech to text module on top and having a natural language understanding built using statistical methods in the background.

The scope of this thesis is the background natural language understanding modules implemented in a traditional speech-based human machine interactive system. Interactive systems based on natural language are quintessentially conversations agents or dialogue systems which are studied in this work. In the next sections, previous works on conversation systems are stated, including different question-answering systems which can be thought of as conversation systems without any understanding of the conversation context, but able to respond to user's request.

3.2 Question Answering Systems

Question answering systems, as defined by [89] wrt. human-computer interaction: "For human-computer interaction, natural language is the best information access mechanism for humans. Hence, Question Answering Systems (QAS) have special significance and advantages over search engines and are considered to be the ultimate goal of semantic Web research for user's information needs". The last few decades have seen many researches on developing question answering systems. Such systems, can be realized using several techniques which will be discussed in the following sections.

3.2.1 Retrieval Based Question Answering System

These kind of systems answers a user query using information retrieval based methods, where an answer is retrieved from a set of documents which could be stored in a database or present in the web. More traditional retrieval based systems are tf-idf based, where they propose additional mechanism like word based matching like in [90], using word embeddings [91] or using language models [92] [93].

Deep Learning based approaches on the other hand firstly also uses shallow term-frequency based models to generate a set of candidate answers, and finally use deep neural network based methods to re-rank the fetched candidates. Such models can be representational (using a dense latent representation for the query and answer candidates) or interaction based (computing interaction between query and answer tokens using DL). Both Convolutional neural network (CNN) and LSTM based methods are proposed for latent representation based QA systems. Some CNN based models are [94] [95], while LSTM based models are [96] [97]. Additionally there are also models proposed using a hybrid CNN and LSTM architecture as proposed in the QA-LSTM model by [98]. For interaction based models, a

matching matrix is created based on the words in the query and answer candidates and learned using attention [99], rnns [100] or more recent pre-trained transformer based (BERT) [49].

3.2.2 Reading Comprehension based Question Answering

Question Answering systems can also be realized using reading comprehension based methods, where a question can be answered using a paragraph. Previous models for reading comprehension were based on LSTM networks, where the question and the paragraph is encoded using a LSTM model, concatenated and henceforth a probability distribution using attention mechanism is created for the probable document tokens which can answer the question. Some of these LSTM models are proposed in the works by [101] [102] [103] [104].

Similar to other NLP tasks, there are also recent systems which uses pre-trained transformers for comprehensive QA. [49] did fine-tuning on a pre-trained transformer BERT model which they proposed. They represent the query and the paragraph using the BERT model and introduce two output vectors which represents the start and end index of the phrase from the paragraph which can answer the query. The final prediction is made by using a softmax over all the words in the paragraph. [105] also reported state-of-the-art accuracy for the task of reading comprehension for QA with auto-regressive pre-trained transformers.

3.2.3 Question Answering over Knowledge Graphs

Next type of question answering systems are over Knowledge Graphs (KGQA); where, given a question, a model firstly tries to predict the entity mentioned in the question (entity linking), then it predicts the relations that are used in the question (relation linking). Both simple and complex questions (based on the number of entities and relations in the questions) can be answered using knowledge graphs.

The SimpleQuestions dataset, as proposed by [106] is the first large scale dataset for simple questions over Freebase. It consists of 108,442 questions split into train(70%), valid(10%), and test(20%). They also proposed an end-to-end architecture using memory networks along with the dataset.

The second end-to-end approach for simple question answering over Freebase was provided by [107]. They proposed a character LSTM based question encoder for encoding the question, a CNN based encoder for encoding the KG entity and relation, and finally an attention-based LSTM decoder for predicting an entity-relation pair given the question. A similar end-to-end approach was suggested by [7]. It employs Gated Recurrent Unit (GRU) based encoders that work on character and word level and in addition encode the hierarchical types of relations and entities to provide further information.

Furthermore, a growing set of modular architectures was proposed on the SimpleQuestions dataset. [95] proposed a character-level CNN for identifying entity mentions and a separate word-level CNN with attentive max-pooling to select Knowledge Graph tuples. [8] utilized a hierarchical residual bidirectional LSTM for predicting a relation, which is then used to re-rank the entity candidates. They replaced the topic entity in the question with a generic token <e> during relation prediction which helps in better distinguishing the relative position of each word compared to the entity.

[108] proposed a conditional probabilistic framework with bidirectional GRUs that takes advantage of Knowledge Graph embeddings. [6] suggested to use a combination of relatively simple, component-based approaches that build on bidirectional GRUs, bidirectional LSTMs (BiLSTMs), and conditional random fields (CRFs) as well as on graph-based heuristics to select the most relevant entity given a

question from a candidate set. The resulting model provides strong baselines for simple question-answering. More recently, [109] proposed an architecture based on KG embeddings, [110] proposed a technique combining LSTM-CRF based entity detection with BiLSTM based relation linking where they also replace the topic entity with generic tokens following [8].

Some other open-domain Knowledge Graphs are Wikidata[111] and DBpedia [112]. In particular, there are two very recent efforts that provided adaptations of the SimpleQuestions dataset [106] to Wikidata [113] and DBpedia [114]. In addition, there is the Question Answering over Linked Data (QALD) [115–117] series of challenges that use DBpedia as a knowledge base for QA.

For complex KGQA, the WebQSP dataset is proposed over freebase which is later mapped to wikidata by [118]. They also additionally propose a graph neural network based model called VCG for the task.

3.3 Dialogue Systems

Along with question answering systems, human-computer conversational systems also have attracted increasing attention in the research community and dialogue systems have become a field of research on its own. The conversation models proposed in early studies [119–121] were designed for catering to specific domains only, e.g. for performing restaurant bookings, and required substantial rule-based strategy building and human efforts in the building process. With the advancements in machine learning, there have been more and more studies on conversational agents which are based on data-driven approaches. Data-driven dialogue systems can chiefly be realized by two types of architectures: (1) pipeline architectures, which follow a modular pattern for modelling the dialogues, where each component is trained/created separately to perform a specific sub-task, and (2) end-to-end architectures, which consist of a single trainable module for modelling the conversations.

Task-oriented dialogue systems, which are designed to assist users in achieving specific goals, were mainly realized by pipeline architectures. Recently however, there have been more and more works on end-to-end dialogue systems because of the limitations of the former modular architectures, namely, the credit assignment problem and inter-component dependency, as for example described by [122]. [123] and [124] proposed encoder-decoder-based neural networks for modeling task oriented dialogues. Moreover, [122] proposed an end-to-end reinforcement learning-based system for jointly learning to perform dialogue state-tracking [86] and policy learning [125].

Since task oriented systems primarily focus on completing a specific task, they usually do not allow free flowing, articulate conversations with the user. Therefore, there has been considerable effort to develop non-goal driven dialogue systems, which are able to converse with humans on an open domain [126]. Such systems can be modeled using either generative architectures, which are able to freely generate responses to user queries, or retrieval-based systems, which pick a response suitable to a context utterance out of a provided set of responses. Retrieval-based systems are therefore more limited in their output while having the advantage of producing more informative, constrained, and grammatically correct responses [127].

3.3.1 Generative models

[126] were the first to formulate the task of automatic response generation as phrase-based statistical machine translation, which they tackled with n-gram-based language models. Later approaches [128–

[130] applied Recurrent Neural Network (RNN)-based encoder-decoder architectures. However, dialogue generation is considerably more difficult than language translation because of the wide possibility of responses in interactions. Also, for dialogues, in order to generate a suitable response at a certain time-step, knowing only the previous utterance is often not enough and the ability to leverage the context from the sequence of previous utterances is required. To overcome such challenges, a hierarchical RNN encoder-decoder-based system has been proposed by [131] for leveraging contextual information in conversations. [132] proposed a neural network architecture which can leverage contextual information in the previous dialogue utterances to generate a response. Following similar lines, an encoder-decoder based system has been proposed by [131] using hierarchical neural architecture. The model essentially contains two stacked RNNs (Recurrent neural networks) stacked on top of one another. The first RNN encodes each utterance into a fixed length vector, this is called as *sentence-level RNN*. The next *context-level RNN* is responsible for summarizing the conversation at time-step t using all the utterance vectors from the sentence-level RNN from time-steps $0..(t - 1)$.

Leveraging background information for dialogue system improvement is a well-researched topic as well, especially in goal-oriented setting [123, 124, 133]. [20] proposed the in-car dataset which uses a knowledge base for in-car conversation about weather, location and so on. Recently, [10] proposed memory-network based encoder-decoder architecture for integrating knowledge into the dialogue generation process on this dataset. Improved models in this task are proposed by [134, 135]. [136] proposed a soccer dialogue dataset along with a KG-Copy mechanism for non-goal oriented dialogues which are KG-integrated.

More recently, transformer-based [48] pre-trained models have achieved success in solving various downstream tasks in the field of NLP such as question answering [137] [49], machine translation [138], summarization [50]. In a recent work, [139] investigated on transformer-based model for non-goal oriented dialogue generation in single-turn dialogue setting.

In past years, there is increased focus on encoding graph structure using neural networks, a.k.a. Graph Neural Networks (GNNs) [140, 141]. A generalization of CNN to graph domain, Graph Convolutional Networks (GCNs) [142] has become popular in the past years. Such architectures are also adapted for encoding and extracting information from Knowledge Graphs [143].

3.3.2 Retrieval-based models

Earlier works on retrieval-based systems focused on modeling short-text, single-turn dialogues. [144] introduced a data set for this task and proposed a response selection system which is based on information retrieval techniques like the vector space model and semantic matching. [127] suggested to apply a deep neural network for matching contexts and responses, while [145] proposed a topic aware convolutional neural tensor network for answer retrieval in short-text scenarios.

More recently, there has been increased focus on developing retrieval-based models for multi-turn dialogues which is more challenging as the models need to take into account long-term dependencies in the context. [146], introduced the Ubuntu Dialogue Corpus (UDC), which is the largest freely available multi-turn dialogue data set. Moreover, the authors proposed to leverage RNNs, e.g. LSTMs, to encode both the context and the response, before computing the score of the pair based on the similarity of the encodings (w.r.t. a certain measure). This class of methods is referred to as dual encoder architectures. Shortly after, [147] investigated the performance of dual encoders with different kind of encoder networks, such as convolutional neural networks (CNNs) and bi-directional LSTMs. [148] followed a different approach and trained a single CNN to map a context-response pair to the

corresponding matching score.

Later on, various extensions of the dual encoder architecture have been proposed. [149] employed two encoders in parallel, one working on word- the other on utterance-level. wu2017sequential proposed the Sequential Matching Network (SMN), where the candidate response is matched with every utterance in the context separately, based on which a final score is computed. The Cross Convolution Network (CNN) [150] extends the dual encoder with a cross convolution operation. The latter is a dot product between the embeddings of the context and response followed by a max-pooling operation. Both of the outputs are concatenated and fed into a fully-connected layer for similarity matching. Moreover, [150] improve the representation of rare words by learning different embeddings for them from the data.

Handling rare words has also been studied by [151], who proposed to handle Out-of-Vocabulary (OOV) words by using both pre-trained word embeddings and embeddings from task-specific data.

Furthermore, many models targeting response selection along with other sentence pair scoring tasks such as paraphrasing, semantic text scoring, and recognizing textual entailment have been proposed. [152] investigated a stacked RNN-CNN architecture and attention-based models for sentence-pair scoring. Match-SRNN [100] employs a spatial RNN to capture local interactions between sentence pairs. Match-LSTM [153] improves its matching performance by using LSTM-based, attention-weighted sentence representations. QA-LSTM [154] uses a simple attention mechanism and combines the LSTM encoder with a CNN.

Incorporating unstructured domain knowledge into dialogue system has initially been studied by [9] and followed by [19], who incorporated a loosely-structured knowledge base into a neural network using a special gating mechanism. They created the knowledge base from domain-specific data, however their model is not able to leverage any external domain knowledge.

Improving Knowledge-Based Question Answering using Structural Information.

In this section, we will answer research question 1: "How the Knowledge Graph Structure can be incorporated for Knowledge Graph Question Answering?". We firstly investigate a modular approach for entity linking namely EARL 4.1, and finally we propose two end-to-end methods: PNEL 4.2 ELiDi 4.3 for entity linking and compare the methods on simplequestion dataset [106]. It must be mentioned that although we are reporting entity linking scores, we also have used the predicted relations for entity linking. This is synonymous to the task of question answering measured by micro-average accuracy of entity and relation pair linking as proposed in [106].

4.1 EARL

The first modular approach, EARL for entity linking and disambiguation is described in this section. We firstly describe the methodology for EARL and then discuss the evaluation and finally the shortcomings of the model. A paper on this topic is published in the international semantic web conference (ISWC) [155]. The author devised the E/R prediction module, the adaptive E/R learning method, and co-writing the paper.

In this section we first discuss the step of span detection of entity and relation in natural language question and candidate list generation. We perform the disambiguation by two different approaches, which are discussed later in this section.

4.1.1 Introduction

As discussed previously, question answering is a two-step process. Firstly, entity linking is done followed by relation linking. for entity linking, The first step is to identify and spot the span of the entity mentioned in a question (entity span detection). The second step is to disambiguate or link the entity to the Knowledge Graph (linking). For linking, the candidates are generated for the spotted span of the entity and then the best candidate is chosen for the linking. These two steps are similarly followed in standard relation linking approaches. In our approach, we first spot the spans of entities and relations. After that, the (disambiguation) linking task is performed jointly for both entities and relations. However, it must be noted that relations can also be directly predicted from the question

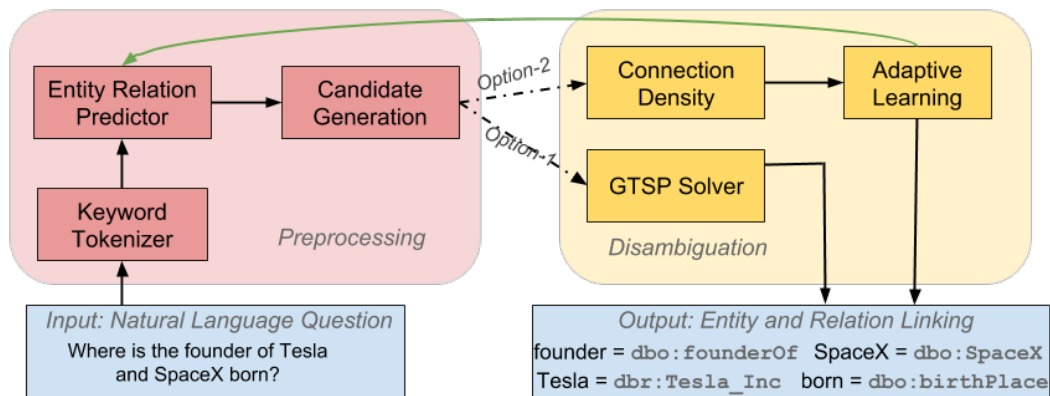


Figure 4.1: EARL Architecture: In the disambiguation phase one may choose either Connection Density or GTSP. In cases where training data is not available beforehand GTSP works better.

itself rather from detecting relation spans.

For EARL, we firstly spot the spans of entities and relations. After that, the (disambiguation) linking task is performed jointly for both entities and relations.

In this section we first discuss the step of span detection of entity and relation in natural language question and candidate list generation. We perform the disambiguation by two different approaches, which are discussed later in this section.

4.1.2 Candidate Generation Steps

Shallow Parsing

Given a question, we firstly perform chunking or shallow parsing. EARL uses SENNA[156] as the keyword extractor. We also remove stop words from the question at this stage. In example question "Where was the founder of Tesla and SpaceX born?" we identify $\langle founder, Tesla, SpaceX, born \rangle$ as our keyword phrases.

E/R Prediction

Once keyword phrases are extracted from the questions, the next step in EARL is to predict whether each of these is an entity or a relation. We use a character embedding based long-short term memory network (LSTM) to do the same. The network is trained using labels for entity and relation in the Knowledge Graph. For handling out of vocabulary words [157], and also to encode the Knowledge Graph structure in the network, we take a multi-task learning approach with hard parameter sharing. Our model is trained on a custom loss given by:

$$\mathcal{E} = (1 - \alpha) * \mathcal{E}_{BCE} + \alpha * \mathcal{E}_{ED} \quad (4.1)$$

Where, \mathcal{E}_{BCE} is the binary cross entropy loss for the learning objective of a phrase being an entity or a relation and \mathcal{E}_{ED} is the squared euclidian distance between the predicted embedding and the correct embedding for that label. The value of α is empirically selected as 0.25. We use pre-trained label embeddings from RDF2Vec [158] which are trained on Knowledge Graphs. RDF2Vec provides

latent representation for entities and relations in RDF graphs. It efficiently captures the semantic relatedness between entities and relations.

We use a hidden layer size of 128 for the LSTM, followed by two dense layers of sizes 512 and 256 respectively. A dropout value of 0.5 is used in the dense layers. The network is trained using Adam optimizer [159] with a learning rate of 0.0001 and a batch size of 128. Going back to the example, this module identifies "founder" and "born" as *relations*, "Tesla" and "SpaceX" as *entities*.

Candidate List Generation

This module retrieves a candidate list for each keyword identified in the natural language question by the shallow parser. To retrieve the top candidates for a keyword we create an Elasticsearch¹ index of URI-label pairs. Since EARL requires an exhaustive list of labels for a URI in the Knowledge Graph, we expand the labels. We used Wikidata labels for entities which are in same-as relation in the knowledge base. For relations we require labels which were semantically equivalent (such as writer, author) for which we took synonyms from the Oxford Dictionary API². To cover grammatical variations of a particular label, we added inflections from fastText³. We avoid any bias held towards or against popular entities and relations.

The output of these pre-processing steps are i) set of keywords from the question, ii) every keyword is identified either as relation or entity, iii) for every keyword there is a set of candidate URIs from the Knowledge Graph.

4.1.3 Using GTSP for Disambiguation

At this point we may use either a GTSP based solution or Connection Density (later explained in 4.3) for disambiguation. We start with the formalisation for GTSP based solution.

The entity and relation linking process can be formalised via spotting and candidate generation functions as follows: Let S be the set of all strings. We assume that there is a function $spot : S \rightarrow 2^S$ which maps a string s (the input question) to a set \mathcal{K} of substrings of s . We call this set \mathcal{K} the *keywords* occurring in our input. Moreover, we assume there is a function $cand_{KG} : \mathcal{K} \rightarrow 2^{V \cup L}$ which maps each keyword to a set of candidate node and edge labels for our Knowledge Graph $G = (V, E, L)$. The goal of joint entity and relation linking is to find combinations of candidates, which are closely related. How closely nodes are related is modelled by a cost function $cost_{KG} : (V \cup L) \times (V \cup L) \rightarrow [0, 1]$. Lower values indicate closer relationships. According to our first postulate, we aim to encode graph distances in the cost function to reward those combinations of entities and relations, which are located close to each other in the input Knowledge Graph. To be able to consider distances between both relations and entities, we transform the Knowledge Graph into its subdivision graph (see Definition ??). This subdivision graph allows us to elegantly define the distance function as illustrated in Figure 4.3.

Given the Knowledge Graph KG and the functions $spot$, $cand$ and $cost$, we can cast the problem of joint entity and relation linking as an instance of the Generalised Travelling Salesman (GTSP) problem: We construct a graph G with $V = \bigcup_{k \in \mathcal{K}} cand(k)$. Each node set $cand(k)$ is called a cluster in this vertex set. The GTSP problem is to find a subset $V' = (v_1, \dots, v_n)$ of V which contains exactly one node from each cluster and the total cost $\sum_{i=1}^{n-1} cost(v_i, v_{i+1})$ is minimal with respect to all such

¹ <https://www.elastic.co/products/elasticsearch>

² <https://developer.oxforddictionaries.com/>

³ <https://fasttext.cc/>

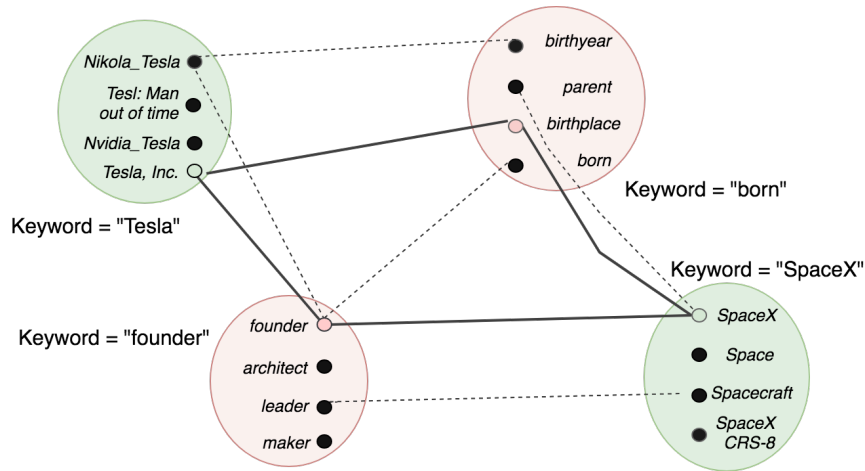


Figure 4.2: Using GTSP for disambiguation : The bold line represents the solution offered by the GTSP solver. Each edge represents an existing connection in the Knowledge Graph. The edge weight is equal to the number of hops between the two nodes in the Knowledge Graph. We also add the index search ranks of the two nodes the edges connect to the edge weight when solving for GTSP.

subsets. Please note that in our formalisation of the GTSP, we do not require V' to be a cycle, i.e. v_1 and v_n can be different. Moreover, we do not require clusters to be disjoint, i.e. different keywords can have overlapping candidate sets.

Figure 4.2 illustrates the problem formulation. Each candidate set for a keyword forms a cluster in the graph. The weight of each edge in this graph is given by the cost function, which includes the distance between the nodes in the subdivision graph of the input Knowledge Graph as well as the confidence scores of the candidates. The GTSP requires the solution to visit one element per cluster and minimises the overall distance.

Approximate GTSP Solvers

In order to solve the joint entity and relation linking problem, the corresponding GTSP instance needs to be solved. Unfortunately, the GTSP is NP-hard [160] and hence it is intractable. However, since GTSP can be reduced to standard TSP, several polynomial approximation algorithms exist to solve GTSP. The state-of-the-art approximate GTSP solver is the Lin–Kernighan–Helsgaun algorithm [161]. Here, a GTSP instance is transformed into standard asymmetric TSP instances using the Noon-Bean transformation. It allows the heuristic TSP solver LKH to be used for solving the initial GTSP. Among LKH’s characteristics, its use of 1-tree approximation for determining a candidate edge set, the extension of the basic search step, and effective rules for directing and pruning the search contribute to its efficiency.

While a GTSP based solution would be suitable for solving the joint entity and relation linking problem, it has the drawback that it can only provide the best candidate for each keyword given the list of candidates. Most approximate GTSP solutions do not explore all possible paths and nodes and hence a comprehensive scoring and re-ranking of nodes is not possible. Ideally, we would like to go beyond this and re-rank all candidates for a given keyword. This would open up new opportunities from a QA perspective, i.e. a user could be presented with a sorted list of multiple possible answers to

GTSP	Connection Density
Requires no training data	Requires data to train the XGBoost classifier
The approximate GSTP LKH solution is only able to return the top result as not all possible paths are explored.	Returns a list of all possible candidates in order of score
Time complexity of LKH is $O(nL^2)$ where n = number of nodes in graph, L = number of clusters in graph of	Time complexity is $O(N^2L^2)$ where N = number of nodes per cluster, L = number of clusters in graph
Relies on identifying the path with minimum cost	Depends on identifying dense and short-hop connections

Table 4.1: Comparison of GTSP based approach and Connection density for Disambiguation

select from.

4.1.4 Using Connection Density for Disambiguation

As discussed earlier, once the candidate list generation is achieved, EARL offers two independent modules for the entity and relation linking. In the previous subsection 4.2 we discussed one approach using GTSP. In this subsection we will discuss the second approach for disambiguation using Connection Density, which works as an alternative to the GTSP approach. We have also compared the two methods in table 2.

Formalisation of Connection Density:

For identified keywords in a question we have the set \mathcal{K} as defined earlier. For each keyword K_i we have list L_i which consists of all the candidate uris generated by text search. We have n such candidate lists for each question given by, $\mathcal{L} = \{L_1, L_2, L_3, \dots, L_n\}$. We consider a probable candidate $c_m^i \in L_i$,

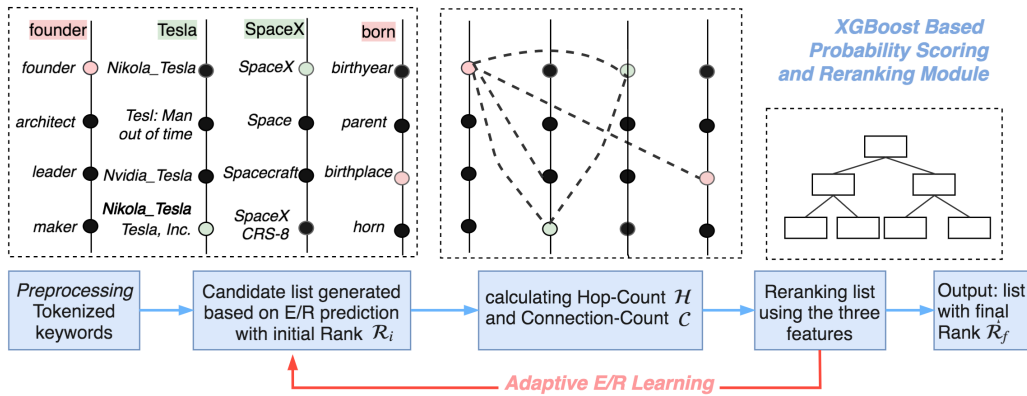


Figure 4.3: Connection Density with example: The dotted lines represent corresponding connections between the nodes in the knowledge base.

where m is the total number of candidates to be considered per keyword, which is the same as the number of items in each list.

The hop distance $dKGhops(c_i^k, c_j^o) \in \mathbb{Z}^+$ is number of hops between c_i^k and c_j^o in the subdivision Knowledge Graph. If the shortest path from c_i^k and c_j^o requires the traversal of h edges then $dKGhops(c_i^k, c_j^o) = h$.

Connection Density is based on the three features: Text similarity based initial Rank of the List item (\mathcal{R}_i) Connection-Count (C) and Hop-Count (\mathcal{H})

Initial Rank of the List (\mathcal{R}_i), is generated by retrieving the candidates from the search index via text search. This is achieved in the preprocessing steps as mentioned in the section 4. Further, to define C we introduce $dConnect$.

$$dConnect(c_i^k, c_j^o) = \begin{cases} 1 & \text{if } dKGhops(c_i^k, c_j^o) \leq 2 \\ 0 & \text{otherwise} \end{cases} \quad (4.2)$$

The Connection-Count C for an candidate c , is the number of connections from c to candidates in all the other lists divided by the total number n of keywords spotted. We consider nodes at hop counts of greater than 2 disconnected because nodes too far away from each other in the knowledge base do not carry meaningful semantic connection to each other.

$$C(c_i^k) = 1/n \sum_{o|o \neq k} \sum_{j=1}^{j=m} dConnect(c_i^k, c_j^o) \quad (4.3)$$

The Hop-Count \mathcal{H} for a candidate c , is the sum of distances from c to all the other candidates in all the other lists divided by the total number of keywords spotted.

$$\mathcal{H}(c_i^k) = 1/n \sum_{o|o \neq k} \sum_{j=1}^{j=m} dKGhops(c_i^k, c_j^o) \quad (4.4)$$

Candidate Re-ranking:

\mathcal{H} , C and \mathcal{R}_i constitute our feature space \mathcal{X} . This feature space is used to find the most relevant candidate given a set of candidates for an identified keyword in the question. We use a machine learning classifier to learn the probability of being the most suitable candidate \bar{c}^i given the set of candidates. The final list \mathcal{R}_f is obtained by re-ranking the candidate lists based on the probability assigned by the classifier. Ideally, \bar{c}^i should be the top-most candidate in \mathcal{R}_f .

The training data consists of the features \mathcal{H} , C and \mathcal{R}_i and a label 1 if the candidate is the correct, 0 otherwise. For the testing, we apply the learned function from the classifier f on \mathcal{X} for every candidate $\in c_i$ and get a probability score for being the most suitable candidate. We perform experiments with three different classifiers, namely extreme gradient boosting(xgboost), SVM(with a linear kernel) and logistic regression to re-rank the candidates. The experiments are done using a 5-fold cross-validation strategy where, for each fold we train the classifier on the training set and observe the mean reciprocal rank (MRR) of \bar{c}^i on the testing set after re-ranking the candidate lists based on the assigned probability. The average MRR on 5-fold cross-validation for the three classifiers are 0.905, 0.704 and 0.794 respectively. Hence, we use xgboost as the final classifier in our subsequent experiments for re-ranking.

Algorithm

We now present a pseudo-code version of the algorithm to calculate the two features: Connection Density algorithm is used for finding hop count and connection count for each candidate node. We then pass these features to a classifier for scoring and ranking This algorithm (Algorithm 1 Connection Density) has a time complexity given by $O(N^2L^2)$ where N is the number of keywords and L is the number of candidates for each keyword.

Algorithmus 1 : Connection Density

```

function :ConnectionDensity()
  input   :  $\mathcal{L}$ , with  $n$  number of keywords // an array of arrays
  output  : Hop-Count  $\mathcal{H}$ , Connection-Count  $C$ 
1 dConnectCounter = { } // Count for connections from and to each node
2 dHopCounter = { } // Similarly hop counts for each node
3 foreach  $L_a \in \mathcal{L}$  do
4   foreach  $c_i^a \in L_a$  do
5      $dConnectCounter[c_i^a] = 0$  // Initialising the dictionary
6      $dHopCounter[c_i^a] = 0$ 
7 foreach  $(L_a, L_b) \in \mathcal{L}$  do
8   foreach  $c_i^a \in L_a$  do
9     foreach  $c_j^b \in L_b$  do
10      if  $dKGhops(c_i^a, c_j^b) \leq 2$  then
11         $dConnectCounter[c_i^a] += 1$ 
12         $dConnectCounter[c_j^b] += 1$ 
13         $dHopCounter[c_i^a] += dKGhops(c_i^a, c_j^b)$ 
14         $dHopCounter[c_j^b] += dKGhops(c_i^a, c_j^b)$ 
15 foreach  $(c_i, score) \in dConnectCounter$  do
16    $C(c_i) = dConnectCounter(c_i)/n$  // Normalisation with respect to
    number of keywords spotted
17 foreach  $(c_i, score) \in dHopCounter$  do
18    $\mathcal{H}(c_i) = dHopCounter(c_i)/n$ 
19 return (Hop-Count  $\mathcal{H}$ , Connection-Count  $C$ )

```

4.1.5 Adaptive E/R Learning

EARL uses a series of sequential modules with little to no feedback across them. Hence, the errors in one module propagate down the line. To trammel this, we implement an adaptive approach especially for curbing the errors made in the pre-processing modules. While conducting experiments, it was observed that most of the errors are in the shallow parsing phase, mainly because of grammatical errors in LC-QuAD which directly affects the consecutive E/R prediction and candidate selection steps. If the E/R prediction is erroneous, it will search in a wrong Elasticsearch index for probable

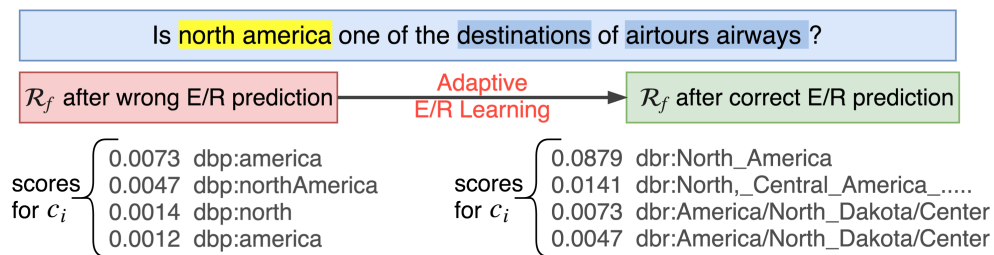


Figure 4.4: Adaptive E/R learning

candidate list generation. In such a case none of the candidates $\in c^i$ for a keyword would contain \bar{c}^i as is reflected by the probabilities assigned to c^i by the re-ranker module. If the maximum probability assigned to c^i is less than a very small threshold value, empirically chosen as 0.01, we re-do the steps from ER prediction after altering the original prediction. If the initial assigned probability is *entity*, we change it to *relation* and vice-versa, example figure 4.4. This module is empirically evaluated in table 4.4.

4.1.6 Evaluation

Data Set: LC-QuAD [162] is the largest complex questions data set available for QA over KGs. We have annotated this data set to create a gold label data set for entity and relation linking, i.e. each question now contains the correct KG entity and relation URIs with their respective text spans in the question. This annotation was done in a semi-automated process and subsequently manually verified. The annotated dataset of 5000 questions is publicly available at <https://figshare.com/projects/EARL/28218>

Experiment 1: Comparison of GTSP, LKH and Connection Density

Aim: We evaluate hypotheses (**H1** and **H2**) that the connection density and GTSP can be used for joint linking task. We also evaluate the LKH approximation solution of GTSP for doing this task. We compare the time complexity of the three different approaches. **Results:** Connection density results in a similar accuracy as that of an exact GTSP solution with a better time complexity (see Table 4.2). Connection density has worse time complexity than approximate GTSP solver LKH if we assume the best case of equal cluster sizes for LKH. However, it provides a better accuracy. Moreover, the average time taken in EARL using connection density (including the candidate generation step) is 0.42 seconds per question. Further observing Table 4.2, we can see that the brute force GTSP solution and Connection Density have similar accuracy, but the brute force GTSP solution has exponential time complexity. The approximate solution LKH has polynomial run time, but its accuracy drops compared to the brute force GTSP solution. Moreover, from a question answering perspective the ranked list offered by the Connection Density approach is useful since it can be presented to the user as a list of possible correct solutions or used by subsequent processing steps of a QA system. Hence, for further experiments in this section we used the connection density approach.

Experiment 2: Evaluating Joint Connectivity and Re-ranker

Aim: Evaluating the performance of Connection Density for predicting the correct entity and relation candidates from a set of possible E-R candidates. Here we evaluate hypothesis **H2**, the correct candidates exhibit relatively dense and short-hop connections.

Metrics: We use the mean reciprocal rank of the correct candidate \bar{c}^i for each entity/relation in the query. From the probable candidate list generation step, we fetch a list of top candidates for each identified phrase in a query with a k value of 10, 30, 50 and 100, where k is the number of results from text search for each keyword spotted. To evaluate the robustness of our classifier and features we perform two tests. i) On the top half of Table 4.3 we re-rank the top k candidates returned from the previous step. ii) On the bottom half of Table 4 we artificially insert the correct candidate into each list to purely test re-ranking abilities of our system (this portion of the table contains k^* as the number of items in each candidate list). We inject the correct uris at the lowest rank (see k^*), if it was not retrieved in the top k results from previous step.

Results: The results in Table 4.3 depict that our algorithm is able to successfully re-rank the correct URIs if the correct ones are already present. In case correct URIs were missing in the candidate list, we inserted URIs artificially as the last candidate. The MRR then increased from 0.568 to 0.905.

Approach	Accuracy (K=30)	Accuracy (K=10)	Time Complexity
Brute Force GTSP	0.61	0.62	$O(n^2 2^n)$
LKH - GTSP	0.59	0.58	$O(nL^2)$
Connection Density	0.61	0.62	$O(N^2 L^2)$

Table 4.2: Empirical comparison of Connection Density and GTSP: n = number of nodes in graph; L = number of clusters in graph; N = number of nodes per cluster; top K results retrieved from ElasticSearch.

Value of k	\mathcal{R}_f based on \mathcal{R}_i	\mathcal{R}_f based on \mathcal{C}, \mathcal{H}	\mathcal{R}_f based on $\mathcal{R}_i, \mathcal{C}, \mathcal{H}$
$k = 10$	0.543	0.689	0.708
$k = 30$	0.544	0.666	0.735
$k = 50$	0.543	0.617	0.737
$k = 100$	0.540	0.534	0.733
$k^* = 10$	0.568	0.864	0.905
$k^* = 30$	0.554	0.779	0.864
$k^* = 50$	0.549	0.708	0.852
$k^* = 50$	0.545	0.603	0.817

Table 4.3: Evaluation of joint linking performance

System	Accuracy LC-QuAD	Accuracy - QALD
FOX [163] + AGDISTIS [164]	0.36	0.30
DBpediaSpotlight [165]	0.40	0.42
TextRazor ⁵	0.52	0.53
Babelfy [166]	0.56	0.56
EARL without adaptive learning	0.61	0.55
EARL with adaptive learning	0.65	0.57

Table 4.4: Evaluating EARL’s Entity Linking performance

System	Accuracy LC-QuAD	Accuracy - QALD
ReMatch [167]	0.12	0.31
RelMatch [168]	0.15	0.29
EARL without adaptive learning	0.32	0.45
EARL with adaptive learning	0.36	0.47

Table 4.5: Evaluating EARL’s Relation Linking performance

Experiment 3: Evaluating Entity Linking

Aim: To evaluate the performance of EARL with other state-of-the-art systems on the entity linking task. This also evaluates our hypothesis **H3**.

Metrics: We are reporting the performance on accuracy. Accuracy is defined by the ratio of the correctly identified entities over the total number of entities present.

Result: EARL performs better entity linking than the other systems (Table 4.4), namely Babelfy, DBpediaSpotlight, TextRazor and AGDISTIS + FOX (limited to entity types - LOC, PER, ORG). We conducted this test on the LC-QuAD and QALD-7 dataset⁴. The value of **k** is set to 30 while re-ranking and fetching the most probable entity.

Experiment 4: Evaluating Relation Linking

Aim: Given a question, the task is to perform relation linking in the question. This also evaluates our hypothesis **H3**.

Metrics: We use the same accuracy metric as in the Experiment 3

Results: As reported in Table 4.5, EARL outperforms other approaches we could run on LC-QuAD and QALD. The large difference in accuracy of relation-linking over LC-QuAD over QALD, is due to the fact that LC-QuAD has 82% questions with more than one relation, thus detecting relation phrases in the question was difficult.

⁴<https://project-hobbit.eu/challenges/qald2017/>

4.1.7 Discussion

Our analysis shows that we have provided two tractable (polynomial with respect to the number of clusters and the elements per cluster) approaches of solving the joint entity and relation linking problem. We experimentally achieve similar accuracy as the exact GTSP solution with both LKH-GTSP and Connection Density with better time complexity, which allows us to use the system in QA engines in practice. It must be noted that one of the salient features of LKH-GTSP is that it requires no training data for the disambiguation module while on the other hand Connection Density performs better given training data for its XGBoost classifier. While the system was tested on DBpedia, it is not restricted to a particular Knowledge Graph.

There are some limitations: The current approach does not tackle questions with hidden relations, such as "How many shows does HBO have?". Here the semantic understanding of the corresponding SPARQL query is to count all TV shows (*dbo:TelevisionShow*) which are owned by (*dbo:company*) the HBO (*dbr:HBO*). Here *dbo:company* is the hidden relation which we do not attempt to link. However, it could be argued that this problem goes beyond the scope of relation linking and could be better handled by the query generation phase of a semantic QA system.

Another limitation is that EARL cannot be used as inference tool for entities as required by some questions. For example Taikonaut is an astronaut with Chinese nationality. The system can only link taikonaut to *dbr:Astronaut*, but additional information can not be captured. It should be noted, however, that EARL can tackle the problem of the "lexical gap" to a great extent as it uses synonyms via the grammar inflection forms.

Our approaches of LKH-GTSP and Connection Density both have polynomial and approximately similar time complexities. EARL with either Connection Density or LKH-GTSP can process a question in a few hundred milliseconds on a standard desktop computer on average. The result logs, experimental setup and source code of our system are publicly available⁶.

4.2 PNEL

The next system we propose is PNEL, which stands for Pointer Network based Entity Linker. As mentioned previously, the entity linking process is two-phased: span detection and linking. These two methods are separately implemented using by different machine learning models which require separate training. Especially for the span detection part, sentences with marked entity spans are a requirement. In practice, such data is not easily available. Moreover, errors introduced by the MD phase cascade on to the ED phase. Hence, a movement towards end-to-end Entity Linkers began [169] [170]. Such systems do not require labelled entity spans during training. In spite of the benefits of end-to-end models some challenges remain: Due to the lack of a span detector at the initial phase, each word of the sentence needs to be considered as an entity candidate for the disambiguation which leads to the generation of a much larger number of entity candidates. To re-rank these candidates a large amount of time is consumed, not just in processing the features of the candidates, but also in compiling their features.

In this work, we remain cognizant of these challenges and design a system that completely avoids querying the Knowledge Graph during runtime. PNEL (Pointer Network based Entity Linker) instead relies on pre-computed and pre-indexed TransE embeddings and pre-indexed entity label and

⁶<https://github.com/AskNowQA/EARL>

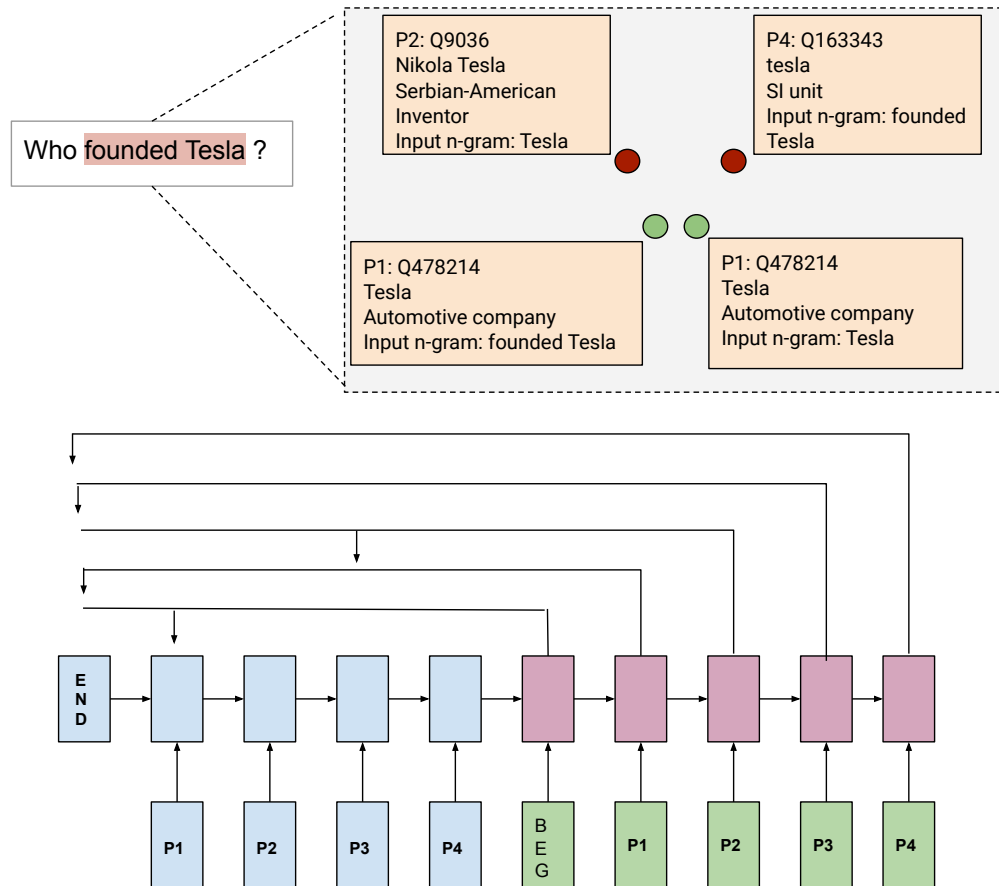


Figure 4.5: The red and green dots represent entity candidate vectors for the given question. The green vectors are the correct entity vectors. Although they belong to the same entity they are not the same dots because they come from different n-grams. At each time step the Pointer Network points to one of the input candidate entities as the linked entity, or to the END symbol to indicate no choice.

description text as the only set of features for a given candidate entity. We demonstrate that this produces competitive performance while maintaining lower response times when compared to another end-to-end entity-linking system, VCG [170].

4.2.1 Introduction

Inspired by the use case of Pointer Networks [45] in solving the convex hull and the generalised travelling salesman problems, this work adapts the approach to solving entity linking. *Conceptually, each candidate entity is a point in an euclidean space, and the pointer network finds the correct set of points for the given problem.* A paper [171] about the proposed method is published in the international semantic web conference (ISWC). The author was responsible on ideating the main proposed algorithm and also co-writing the paper.

4.2.2 Encoding for Input

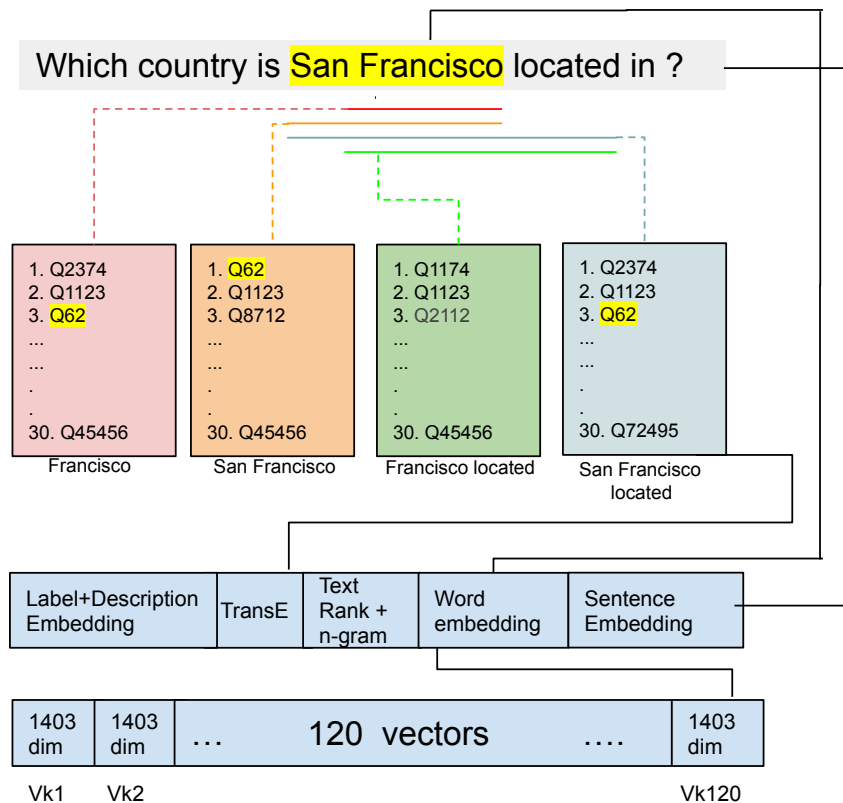


Figure 4.6: The word “Francisco” is vectorised in the following manner: 4 n-grams represented by the underlines are considered and searched against an entity label database. The top 50 search results are depicted for each of the n-grams resulting in 200 candidates. For the entity Q72495, for example, we fetch its TransE embedding, add its text search rank, n-gram length, word index position as features. Additionally we also append the fastText embedding for “Francisco” and the entire fastText embedding for the sentence (average of word vectors) to the feature. We then append the fastText embeddings for the label and description for this entity. Hence we get a 1142 dimensional vector V_{k120} **corresponding to entity candidate Q72495**. For all 200 candidate entities for “Francisco”, we have a sequence of two hundred 1142 dimensional vectors as input to the pointer network. For the sentence above which has 7 words, this results in a final sequence of $7 \times 200 = 1400$ vectors each of length 1142 as input to our pointer network. Any one or more of these vectors could be the correct entities.

The first step is to take the input sentence and vectorise it for feeding into the pointer network. We take varying length of n-grams, also called n-gram tiling and vectorise each such n-gram.

Given an input sentence $S = \{s_1, s_2, \dots, s_n\}$ where s_k is a token (word) in the given sentence, we vectorise s_k to v_k , which is done in the following manner:

1. Take the following 4 n-grams: $[s_k]$, $[s_{k-1}, s_k]$, $[s_k, s_{k+1}]$, $[s_{k-1}, s_k, s_{k+1}]$
2. For each such n-gram find the top L text matches in the entity label database. We use the OKAPI BM25 algorithm for label search.

3. For each such candidate form a candidate vector comprising of the concatenation of the following features
 - a) R_{kl} = Rank of entity candidate in text search (length 1), where $1 \leq l \leq L$
 - b) $ngramlen$ = The number of words in the current n-gram under consideration where $1 \leq ngramlen \leq 3$ (length 1)
 - c) k = The index of the token s_k (length 1)
 - d) pos_k = A one-hot vector of length 36 denoting the PoS tag of the word under consideration. The 36 different tags are as declared in the Penn Treebank Project [172]. (length 36)
 - e) $EntEmbed_{kl}$ = TransE Entity Embedding (length 200)
 - f) $SentFTEEmbed$ = fastText embedding of sentence S (length 300), which is a mean of the embeddings of the tokens of S . In some sense this carries within it the problem statement.
 - g) $TokFTEEmbed_k$ = fastText embedding of token s_k (length 300). Addition of this feature might seem wasteful considering we have already added the sentence vector above, but as shown in the ablation experiment in Experiment 4.11, it results in an improvement.
 - h) $DescriptionEmbed_{kl}$ = fastText embedding of the Wikidata description for entity candidate kl (length 300)
 - i) $TextMatchMetric_{kl}$ = This is a triple of values, each ranging from 0 to 100, that measures the degree of text match between the token under consideration s_k and the label of the entity candidate kl . The three similarity matches are *simple ratio*, *partial ratio*, and *token sort ratio*. In case of *simple ratio* the following pair of text corresponds to perfect match: "Elon Musk" and "Elon Musk". In case of *partial ratio* the following pair of text corresponds to a perfect match: "Elon Musk" and "Musk". In case of *token sort ratio* the following pair of text corresponds to a perfect match: "Elon Musk" and "Musk Elon". (length 3)

For each token s_k we have an expanded sequence of token vectors, comprising of 4 n-grams, upto 50 candidates per n-gram, where each vector is of length 1142. Hence each token s_k is transformed into $4 \times 50 = 200$ vectors, each a 1142 length vector. We may denote this transformation as $s_k \rightarrow \{v_{k1}, v_{k2}, \dots, v_{k200}\}$. Note that there may be less than 50 search results for a given token so there may be less than 200 entity candidates in the final vectorisation. Each of these v_k vectors is an entity candidate.

4.2.3 Training

For the entire sentence, a sequence of such vectors is provided as input to the pointer network. During training the labels for the given input sequence are the index numbers of the correct entities in the input sequence. Note that the same entity appears multiple times because of n-gram tiling and searching. During each decoding time step the decoder produces a softmax distribution over the input sequence, which in our implementation has a maximum sequence length of 3000. Additionally the BEGIN, END, PAD symbols add to a total of 3003 symbols to softmax over. The cross entropy loss function is averaged over the entire output sequence of labels and is considered the final loss for the entire input sequence.

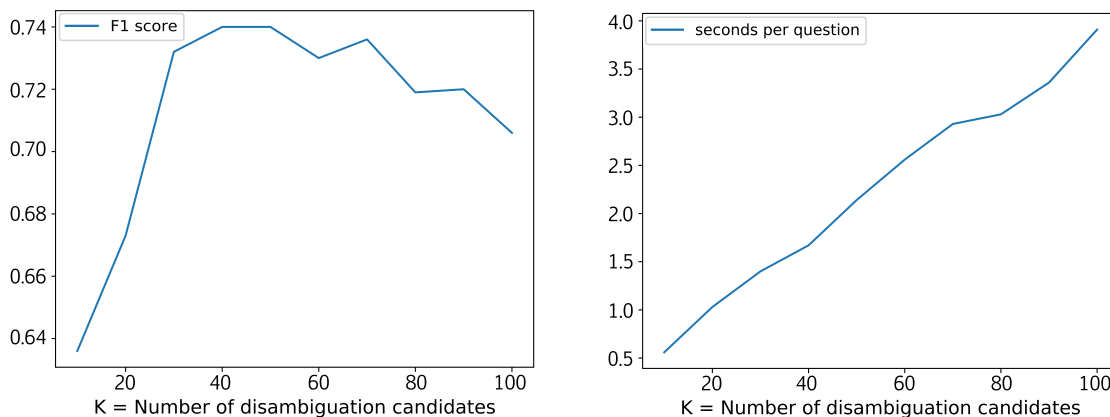


Figure 4.7: K =Number of search candidates per n -gram. On the left: K vs F1 score on a set of 100 WebQSP test questions, with average word length of 6.68. F1 is maximum for $K=40$ and 50. On the right: K vs time taken for PNEL to return a response. The relationship appears to be close to linear.

Network Configuration

We use a single layer bi-LSTM [173] pointer network with 512 hidden units in a layer and an attention size of 128. Addition of an extra layer to the network did not result in an improvement. The Adam optimizer [159] was used with an initial learning rate of 0.001. A maximum input sequence length of 3000 and a maximum output length of 100 were enforced.

4.2.4 Datasets

For reasons explained in section 1 we exclusively evaluate on Wikidata based datasets. We use the following:

- **WebQSP:** We use the dataset released by Sorokin et al [170] where the original WebQSP dataset by Yih et al [174], which was based on Freebase, has been adapted and all Freebase IDs converted to their respective Wikidata IDs. WebQSP contains questions that were originally collected for the WebQuestions dataset from web search logs (Berant et al [175]). WebQSP is a relatively small dataset consisting of 3098 train 1639 test questions which cover 3794 and 2002 entities respectively. The dataset has a mixture of simple and complex questions. We found some questions in the test set that had failed Freebase to Wikidata entity ID conversions. We skipped such questions during PNEL’s evaluation.
- **SimpleQuestions:** To test the performance of PNEL on simple questions, we choose SimpleQuestions [106], which as the name suggests, consists only of Simple Questions. The training set has more than 30,000 questions while the test set has close to 10,000 questions. This dataset was also originally based on Freebase and later the entity IDs were converted to corresponding Wikidata IDs. However out of the 10,000 test questions only about half are answerable on the current Wikidata.
- **LC-QuAD 2.0:** Unlike the first two datasets, LC-QuAD 2.0 [176] is based on Wikidata since its inception and is also the most recent dataset of the three. It carries a mixture of simple and

complex questions which were verbalised by human workers on Amazon Mechanical Turk. It is a large and varied dataset comprising of 24180 train questions and 6046 test questions which cover 33609 and 8417 entities respectively.

4.2.5 Evaluation

In this section we evaluate our proposed model(s) against different state-of-the-art methods for KGQA. As notations, PNEL-L stands for PNEL trained on LC-QuAD 2.0. PNEL-W and PNEL-S stand for PNEL trained on WebQSP and SimpleQuestions respectively.

Experiment 1 : EL over KBPearl split of LC-QuAD 2.0 test set

Objective: The purpose of this experiment is to benchmark PNEL against a large number of EL systems not just over Wikidata but also other KBs.

Method: The results are largely taken from KBPearl. PNEL is trained on the LC-QuAD 2.0 training set. For a fair comparison, the systems are tested on the 1294 questions split of test set provided by KBPearl. We train PNEL for 2 epochs.

Remarks: Results for Falcon 2.0 and OpenTapioca were obtained by accessing their live API. The original Falcon 2.0 paper provides an F1 of 0.69 on 15% of randomly selected questions from a combination of the train and test splits of the dataset. Several systems in the table below do not originally produce Wikidata entity IDs, hence the authors of KBpearl have converted the IDs to corresponding Wikidata IDs.

Analysis: As observed from the results in Table 1, PNEL outperforms all other systems on this particular split of LC-QuAD 2.0 dataset.

Entity Linker	Precision	Recall	F1
Falcon[177]	0.533	0.598	0.564
EARL[178]	0.403	0.498	0.445
Spotlight[165]	0.585	0.657	0.619
TagMe[179]	0.352	0.864	0.500
OpenTapioca[180]	0.237	0.411	0.301
QKBfly[181]	0.518	0.479	0.498
Falcon 2.0	0.395	0.268	0.320
KBPearl-NN	0.561	0.647	0.601
PNEL-L	0.803	0.517	0.629

Table 4.6: Evaluation on KBPearl split of LC-QuAD 2.0 test set

Experiment 2 : EL over full LC-QuAD 2.0 test set

Objective: The objective of this experiment is to compare systems that return Wikidata IDs for the EL task.

Method: We train PNEL on LC-QuAD 2.0 train set and test on all 6046 questions in test set. PNEL

was trained for 2 epochs.

Remarks: Results for competing systems were obtained by accessing their live APIs. We choose systems that return Wikidata IDs.

Analysis: Similar to the previous experiment, PNEL performs the best on the LC-QuAD 2.0 test set.

Entity Linker	Precision	Recall	F1
VCG[170]	0.516	0.432	0.470
OpenTapioca[180]	0.237	0.411	0.301
Falcon 2.0	0.418	0.476	0.445
PNEL-L	0.688	0.516	0.589

Table 4.7: Evaluation on LC-QuAD 2.0 test set

Experiment 3 : EL over WebQSP test set

Objective: Benchmark against an end-to-end model that returns Wikidata IDs.

Method: Train and test PNEL on WebQSP train and test sets respectively. PNEL is trained for 10 epochs.

Remarks: Results for the competing systems were taken from Sorokin et al [170].

Entity Linker	Precision	Recall	F1
Spotlight	0.704	0.514	0.595
S-MART[182]	0.666	0.772	0.715
VCG[170]	0.826	0.653	0.730
PNEL-L	0.636	0.480	0.547
PNEL-W	0.886	0.596	0.712

Table 4.8: Evaluation on WebQSP

Analysis: PNEL comes in third best in this experiment, beaten by VCG and S-MART. S-MART has high recall because it performs semantic information retrieval apart from lexical matching for candidate generation. VCG is more similar to PNEL in that it is also an end-to-end system. It has higher recall but lower precision than PNEL.

Experiment 4 : EL over SimpleQuestions test set

Objective: Benchmark systems on the SimpleQuestions Dataset.

Method: Train and test PNEL on SimpleQuestions train and test sets respectively. PNEL is trained for 2 epochs.

Remarks: We extended the results from Falcon 2.0 [183] article.

Entity Linker	Precision	Recall	F1
OpenTapioca[180]	0.16	0.28	0.20
Falcon 2.0	0.38	0.44	0.41
PNEL-L	0.31	0.25	0.28
PNEL-S	0.74	0.63	0.68

Table 4.9: Evaluation on SimpleQuestions

Analysis: PNEL outperforms the competing systems both in precision and recall for SimpleQuestions dataset. As observed, PNEL has the best precision across all datasets, however, recall seems to be PNEL’s weakness.

Experiment 5 : Candidate generation accuracy

Objective: The purpose of this experiment is to see what percentage of correct entity candidates were made available to PNEL after the text search phase. This sets a limit on the maximum performance that can be expected from PNEL.

Remarks: PNEL considers each token a possible correct entity, but since it only considers top-K text search matches for each token, it also loses potentially correct entity candidates before the disambiguation phase. The results below are for K=30.

Dataset	PNEL (%)
WebQSP	73
LC-QuAD 2.0	82
SimpleQuestions	90

Table 4.10: Entity Candidates available post label search

Experiment 6 : Ablation of features affecting accuracy

Objective: Our final experiment is an ablation study on the WebQSP dataset to understand the importance of different feature vectors used in the model.

Analysis: It appears that the most important feature is the TransE entity embedding, which implicitly contains the entire KG structure information. On removing this feature there is drop in F1 score from 0.712 to 0.221. On the other hand the least important feature seem to be the description embedding. Removal of this feature merely leads to a drop in F1 from 0.712 to 0.700. A possible reason is that the Text Search Rank potentially encodes significant text similarity information, and TransE potentially encodes other type and category related information that description often adds. Removal of the Text Search Rank also results in a large drop in F1 reaching to 0.399 from 0.712.

Sentence Embed.	Word Embed.	Descript. Embed.	TransE	PoS Tags	Text Rank	n-gram length	Text Match Metric	F1 Score
✓	✓	✓	✓	✓	✓	✓	✓	0.712
	✓	✓	✓	✓	✓	✓	✓	0.554
✓		✓	✓	✓	✓	✓	✓	0.666
✓	✓		✓	✓	✓	✓	✓	0.700
✓	✓	✓		✓	✓	✓	✓	0.221
✓	✓	✓	✓		✓	✓	✓	0.685
✓	✓	✓	✓	✓		✓	✓	0.399
✓	✓	✓	✓	✓	✓		✓	0.554
✓	✓	✓	✓	✓	✓	✓		0.698

Table 4.11: Ablation test for PNEL on WebQSP test set features

Experiment 7 : Run Time Evaluation

Objective: We look at a comparison of run times across the systems we have evaluated on

System	Seconds	Target KG
VCG	8.62	Wikidata
PNEL	3.14	Wikidata
Falcon 2.0	1.08	Wikidata
EARL	0.79	DBpedia
TagME	0.29	Wikipedia
Spotlight	0.16	DBpedia
Falcon	0.16	DBpedia
OpenTapioca	0.07	Wikidata

Table 4.12: Time taken per question on the WebQSP dataset of 1639 questions

4.2.6 Discussion

In this work we have proposed PNEL, an end-to-end Entity Linking system based on the Pointer Network model. We make no modifications to the original Pointer Network model, but identify its utility for the problem statement of EL, and successfully model the problem so the Pointer Network is able to find the right set of entities. We evaluate our approach on three datasets of varying complexity and report state of the art results on two of them. On the third dataset, WebQSP, we perform best in precision but lag behind in recall. We select such features that require no real time KG queries during inference. This demonstrates that the Pointer Network model, and the choice of features presented in this work, result in a practical and deployable EL solution for the largest Knowledge Graph publicly available - Wikidata.

A prominent feature of PNEL is high precision and low recall. We focus on loss in recall in this

section. For LC-QuAD 2.0 test set consisting of 6046 questions, the precision, recall and F-score are 0.688, 0.516 and 0.589 respectively. We categorise the phases of loss in recall in two sections 1) Failure in the candidate generation phase 2) Failure in re-ranking/disambiguation phase. When considering the top 50 search candidates during text label search, it was found that 75.3% of the correct entities were recovered from the entity label index. This meant that before re-ranking we had already lost 24.7% recall accuracy. During re-ranking phase, a further 23.7% in absolute accuracy was lost, leading to our recall of 0.516. We drill down into the 23.7% absolute loss in accuracy during re-ranking, attempting to find the reasons for such loss, since this would expose the weaknesses of the model. In the plots below, we consider all those questions which contained the right candidate entity in the candidate generation phase. Hence, we discard those questions for our analysis, which already failed in the candidate generation phase.

Entity Count	Questions Count	Precision	Recall	F1
1	3311	0.687	0.636	0.656
2	1981	0.774	0.498	0.602
3	88	0.666	0.431	0.518

Table 4.13: Comparison of PNEL’s performance with respect to number of entities in a question.

4.3 ELiDi

All question answering or entity-linking systems along with EARL and PNEL suffer from two main problems. Firstly, such systems are not able to handle relations which are not seen during training [184], and hence exhibits poor performance in terms of adaptability to new domains and datasets. Secondly, most systems, and specifically publicly available entity-linking resources are trained on a particular dataset, over a specific Knowledge Graph and are not easily extendable to new datasets or Knowledge Graph, in a zero-shot setting [185]. Moreover, an open-source entity-linking system should be fast and easily be integrated into systems with minimal hardware requirements. Knowledge graph embeddings capture semantic similarity between relations which can potentially aid in learning relations not seen during training but are semantically similar to relations present in the training set. In this chapter, we propose a novel, end-to-end, fast system for entity linking leveraging Knowledge Graph embeddings. The system achieves comparable entity-linking performance with respect to other state-of-the-art methods for entity linking. Furthermore, the model is extendable to unseen datasets and exhibits state-of-the-art performance in zero-shot entity linking settings.

A paper [186] about ELiDi is submitted in the extended semantic web conference. The author was responsible in devising and implementing the main algorithm, running the experiments and also co-writing the paper.

4.3.1 Introduction

Knowledge graphs are data structures that lack a default numerical representation that allows their straightforward application in a standard machine learning context. Statistical relation learning therefore relies beside other approaches on latent feature models for making prediction over KGs. These latent features usually correspond to embedding vectors of the KG entities and relations. Given

the embeddings of the entities and relations of a fact (h, r, t) , a score function outputs the a value indicating the probability that the fact is existing in the KG. In this chapter, we use TransE [187] to learn the KG embeddings used by our model. Let the embedding vector of subject and object entity be given by \vec{h} and \vec{t} respectively, and that of the relation by a vector \vec{r} , then the score function of TransE is given by $f(h, r, t) = -\|\vec{h} + \vec{r} - \vec{t}\|$. We employ two kinds of embeddings in the proposed model namely word embeddings and KG embeddings which are defined in the previous section.

For matching a question to the entities and relations of a KG, likely candidates are first selected in an reprocessing step to reduce the enormous number of candidates in the KG. This is described in the following sections.

Entity Candidates Generation

For the end-to-end training process, we first start with a simple language based candidate generation process selecting potential candidate entities for a given question. That is, given a question we generate candidates by matching the tf-idf vectors of the query with that of the entity labels of all the entities in the Knowledge Graph, resulting in a list of n entity candidates $e_c^1, e_c^2 \dots e_c^n$ for a given question. This list is then re-ranked based on the tf-idf similarity score, whether the candidate label is present in the question, number of relation the candidate is connected to in the KG and whether it has a direct mapping to wikipedia or not. This is done to give importance to important entities (defined by connectivity in the KG) following previous works [6] [110].

Relation Candidate Generation

To generate a set of entity specific relation candidates, for each entity e_c^j in the entity candidate set we extract the list of relations connected to this candidate at a 1-hop distance in the Freebase Knowledge Graph. For the j^{th} entity candidate e_c^j , the relation candidates are $r_j^1 \dots r_j^n$.

4.3.2 Model Description

The proposed neural network model is quintessentially composed of three parts as visualized in 4.8:

1. A word-embedding based **entity span detection model**, which selects the probable words of an entity in a natural language question, represented by a bi-LSTM.
2. An word-embedding based **relation prediction model** which links the the question to one of the relations in the Knowledge Graph, represented by a bi-LSTM with self-attention.
3. An **entity prediction model** which takes the predictions of the previous two submodels into account and employs a sentinel gating mechanism that performs disambiguation based on similarity measures.

The different model parts and the training objective of the resulting model will be described in more detail in the following.

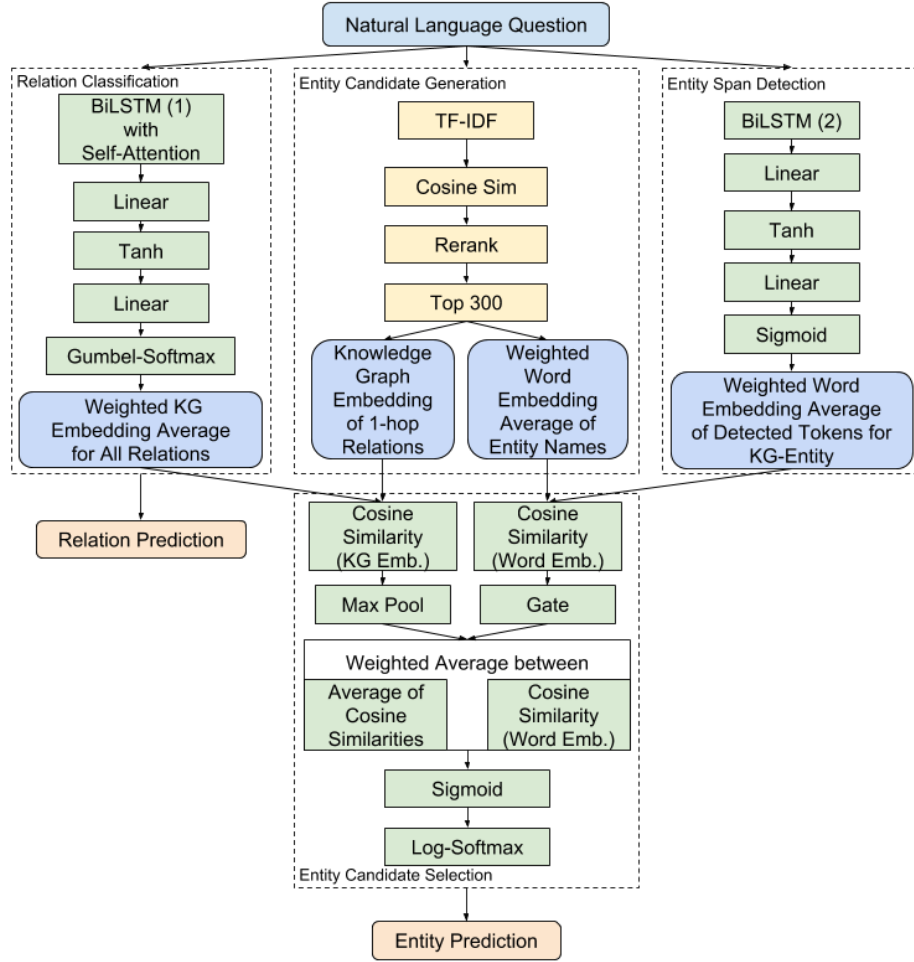


Figure 4.8: Architecture of ELiDi.

Entity Span Detection

The span detection module is inspired by [6]. The given question is firstly passed through a bi-directional LSTM. Its output hidden states are then passed through a fully connected layer and a sigmoid (σ) activation function which outputs the probability of the word at time-step t corresponding to an entity (or not). Mathematically, this can be described as

$$h_t = f_{span}(x_t) ; o_t = \sigma(Wh_t) \quad (4.5)$$

where, o_t is the output probability, W the weight matrix of the fully connected layer, and h_t the hidden state vector from the applying the bi-directional LSTM f_{span} on the input question x_t . We use I-O encoding for the output for training.

Relation Prediction

For relation prediction, the question is passed into a self-attention based bi-directional LSTM which was inspired by [188]. The attention weighted hidden states are then fed into a fully-connected classification layer outputting a probability over the r_n relations in the Knowledge Graph.

$$y_r = \tanh(W_r(\alpha_r * f_{rel}(x_t))) \quad (4.6)$$

W_r , being a set of model parameters, α_r the self-attention weights and f_{rel} is the Bi-LSTM function which produces a response for every time-step t for the input query x_t .

Entity prediction

Word-based Entity Candidate Selection: With the help of the entity span identified by the span detection submodule described in 4.3.2, the questions are now compared to the entity candidates based on vector similarity methods. More specifically, the word embedding of each word of the question is multiplied with corresponding output probability from the entity-span detection model leading to an "entity-weighted" word representation

$$e_t = o_t * w_t^{emb} \quad (4.7)$$

where, w_t^{emb} denotes the word embedding of the t -th word in the question and o_t is the sigmoid output from 4.3.2. We then take a simple average of the entity-weighted representations of all words of the questions to yield the entity embedding of the question e_q^{emb} .

Similarly, the entity candidates $e_c^1, e_c^2 \dots e_c^n$ generated in the preprocessing step described in 4.3.1 are represented by the word embeddings of their labels $e_c^{emb1}, e_c^{emb2} \dots e_c^{embn}$. If a label consists of multiple words, the word embeddings are averaged to yield a single representation. Finally, to compute the similarity between a question and an entity candidate, the cosine between the question embedding and the entity embedding is estimated. For the j^{th} candidate, that is

$$sim_c^j = \cos(e_q^{emb}, e_c^{embj}) \quad (4.8)$$

and the vector $sim_c = (sim_c^1, \dots, sim_c^n)$ represents the word based similarity of the question to all entity candidates.

KG-based Entity Candidate Selection To leverage the relational information encoded in the KG, we firstly take the logits over r_n from the relation prediction model and draw a categorical representation using gumbel softmax [189]. This representation is multiplied with the KG embeddings over r_n to get a KG embedding based representation of the query r_q^{emb} . This relation specific representation is then compared against the full relation candidate set of each candidate entity, where each candidate relation is as well represented by its KG embedding. To match the relation specific question representation to relation candidates for a given entity, we estimate the cosine similarity of the corresponding KG embeddings followed by a max-pooling operation over all the candidate relations of an entity which produces an entity specific similarity metric sim_{kg}^j , which indicates the degree of matching between the question and an entity candidate from a KG perspective, which specifically takes relation information into account. Mathematically, for the j -th entity candidate, let the embedding of the k -th relation candidate r_j^k be denoted by r_j^{embk} . The KG based similarity sim_{kg}^j between the question and

the j -th entity then given by

$$\begin{aligned} sim_{kg}^j = \maxpool(&cos(r_q^{emb}, r_j^{emb1}), \\ &cos(r_q^{emb}, r_j^{emb2}) \\ &\dots \\ &cos(r_q^{emb}, r_j^{embk})) , \end{aligned} \quad (4.9)$$

and the vector $sim_{kg} = (sim_{kg}^1, \dots, sim_{kg}^n)$ represents the KG based similarity of the question to all entity candidates.

Disambiguation and final prediction The final entity prediction is based on the word- and KG-based similarity measures sim_c and sim_{kg} . First, for disambiguation, the word based similarity vector sim_c is passed into a gating mechanism

$$g_{amb} = W_g sim_c \quad (4.10)$$

with $W_g \in \mathbf{R}^{n \times 1}$, which aims at estimating if there is more than one single likely candidate in the entity candidate set based on word similarity. If so, the KG based similarity sim_{kg} should also be taken into account, which is done by averaging sim_c and sim_{kg} and predicting the final entity candidate by

$$y_p^e = \sigma(g_{amb} * \text{mean}(sim_{kg}, sim_c) + (1 - g_{amb}) * sim_c) \quad (4.11)$$

Note that, $y_p^e \in \mathbf{R}^n$ are the logits over the set of candidate entities, from which the entity with the highest probability can be picked. During inferencing, we perform an additional step for ensuring the entity and relation predicted from the model forms a pair in the KG. In order to achieve that, we take the top 5 probable relation from the relation linker and choose the one which is connected to the predicted entity at 1-hop.

4.3.3 Training & Inference

Training objective

The model is trained based on a multi-task objective, where the total loss is the sum of the losses from the entity span detection, relation detection, entity candidate prediction, and disambiguation. The individual loss function are given below⁷.

The loss function for the entity span detection model is the average binary cross entropy L_{span} over the words of the input question, with

$$\begin{aligned} L_{span} &= \frac{1}{T} \sum_{t=1}^T l_t \\ l_t &= -[y_t \cdot \log \sigma(o_t) + (1 - y_t) \cdot \log(1 - \sigma(o_t))] \end{aligned} \quad (4.12)$$

where y_t is the label denoting if the t -th word belongs to the entity span or not. For relation prediction,

⁷ y is used to denote the true label for all tasks here

a weighted cross-entropy loss L_{rel} is used (where the weights are given by the relative ratio of relations in the training set having the same class as the sample) and for entity prediction a vanilla cross-entropy loss L_{ent} , which depends on the parameters of all sub-models. Furthermore, an additional cross entropy loss function L_{amb} is used to train the gating function. Last but not least, we add an regularization term for soft-parameter sharing following [190] resulting in a total loss given by

$$L = L_{span} + L_{rel} + L_{ent} + L_{amb} + \|W_{span}^1 - W_{rel}^1\|^2 \quad (4.13)$$

where, W_{span}^1 and W_{rel}^1 are the hidden layer weights of the entity span detection and relation detection module. Given L , all parameters of the model are jointly trained in an end-to-end manner.

Inferencing

During inference, for a given question, firstly the entity span is predicted using 4.3.2. The candidate generation step from 4.3.1 during inference is different than the process used during training. Instead of using the whole question, we get a set of candidates using the predicted span. Henceforth, using these candidate labels, all the other steps namely relation linking 4.3.2, entity prediction using labels 4.3.2, entity candidate selection using the KG 4.3.2 and finally the disambiguation and prediction using the gating mechanism 4.3.2 is performed. For extending the model to new knowledge graphs, like wikidata [111], in a zero-shot setting we firstly try to get the WikiData ID of the predicted freebase entity id in case there is a direct mapping available. Otherwise, we get the predicted span, get ngrams out of them and search in wikidata with entity labels using these extracted ngrams 2. All the SPARQL calls are made using wikidata query service ⁸.

Algorithmus 2 : Wikidata Entity ID mapping

```

1 WikiID: Wikidata ID of the linked entity
2  $fb\_id \leftarrow get\_freebase\_id(Q)$   $wd\_entity \leftarrow get\_freebase\_mapping(fb\_id)$ 
3 if  $wd\_entity \neq \emptyset$  then
4   |  $WikiID \leftarrow wd\_entity$ 
5 else
6   |  $e\_span \leftarrow detect\_entity\_span(Q)$   $span\_ngram \leftarrow get\_span\_ngram(e\_span)$  while
   |    $WikiIDs \neq \emptyset$  do
7   |   |  $WikiID \leftarrow get\_ID\_from\_label(span)$ 
8 return WikiID
```

Model Settings

We use the pre-processed data and word-embeddings provided by [6] to train our models. To obtain KG embeddings, we train TransE [187] on the provided Freebase KG of 2 million entities. The size of the word embedding vectors is 300, and that of the KG embeddings is 50. The KG embeddings are kept fixed but the word embeddings are fine-tuned during optimization. For training the disambiguation gate g_{amb} we use a label of 1 if the correct entity label is present more than one times in the entity candidates, and label of 0 otherwise. For training, a batch-size of 100 is used and the model is trained

⁸<https://query.wikidata.org/>

Table 4.14: Entity Linking Accuracy over Freebase Knowledge Graph

Model	Accuracy(% of Cand. Present)
BiLSTM [6]	65.0 (-)
Attentive CNN [95]	73.6 (-)
ELiDi (n=100)	77.8(92.0)
ELiDi (n=200)	78.3(94.3)
ELiDi (n=300)	78.6(95.4)

for 100 epochs. We save the model with the best validation accuracy of entity prediction and evaluate it on the test set. We apply Adam [159] for optimization with a learning rate of $1e-4$. The size of the hidden layer of both the entity span and relation prediction Bi-LSTM is set to 300. The training process is conducted on a GPU with 3072 CUDA cores and a VRAM of 12GB.

The average inference speed is 1.5 secs/query in a machine with 2 cores and 2.60 Ghz. It must be noted that, this time includes the time require to make the Wikidata API call (0.7 secs/query).

4.3.4 Datasets

SimpleQuestions (SQ): The SimpleQuestions dataset [106], which, as the name implies, consists only of simple questions, is used to test the performance of ELiDi on simple questions. More than 30,000 questions are in the training set, whereas the test set has close to 10,000 questions. This dataset was published based on Freebase in its initial release. Later a mapping from freebase entity IDs to corresponding Wikidata entity IDs was published to use Wikidata as the knowledge base. Nearly half of the questions in the test data are answerable by the Wikidata version of the data.

WebQSP: The WebQSP dataset is consists of both simple and complex types of questions. The original version of the WebQSP was released by [174]. Later, [191] released a Wikidata version of the data by mapping the Freebase entity IDs to corresponding Wikidata entity IDs. We use the dataset released by [191] as Freebase is no longer maintained. The dataset contains 3098 questions in the train set and 1639 questions in the test set.

4.3.5 Evaluation

In this section, we compare our model with other state-of-the-art end-to-end entity linking systems as well as other open-source resources for entity linking over both freebase and Wikidata.

Entity-linking over Freebase

We compare our system for the task of entity-linking over freebase with other state-of-the-art, end-to-end entity linking systems for the simple question dataset [106]. The results are summarized in 4.14. The system is baselined against a simple BiLSTM model as proposed by [6] and Attentive CNN [174]. It must be noted that although there are more recent systems on the task with pre-trained transformers, specifically BERT [49] based models such as the works by [192]. However, in this chapter we have focused on proposing a resource which is energy and cost efficient, and can be deployed in low-resource configurations unlike pre-trained transformers [193].

Table 4.15: Entity Linking on SimpleQuestions over Wikidata Knowledge Graph

Model	Precision	Recall	F1-score
OpenTapioca[180]	16.0	28.0	20.0
Falcon 2.0[194]	38.0	44.0	41.0
PNEL-S[171]	74.0	63.0	68.0
ELiDi (n=100)	76.8	76.3	76.5

Table 4.16: Entity Linking on WebQSP over Wikidata Knowledge Graph

Model	Precision	Recall	F1-score
TAGME[195]	53.1	27.3	36.1
DBpedia Spotlight[165]	70.4	51.4	59.5
ELiDi (n=100)	71.5	61.3	66.0

Entity-linking over Wikidata

Simple Questions over Wikidata: Additionally, the pre-trained ELiDi, trained on SimpleQuestions over Freebase is used for the SimpleQuestions over Wikidata test-set in a zero-shot setting. The entities in the questions are converted to their corresponding Wikidata IDs and is also used in the recent works by [171]. The results are summarized in Table 4.15. We are baselining against 3 systems which are trained on the simplequestion dataset namely: OpenTapioca [180], Falcon 2.0[194] and PNEL [171]. As observed, ELiDi outperforms all the other systems, even in a zero-shot setting.

WebQSP over Wikidata: We furthermore have evaluated ELiDi on the WebQSP dataset over Wikidata proposed by [191] in a similar zero-shot setting like before. The results are portrayed in Table 4.16. We have baselined against open-source entity linking tools for this dataset namely DBpedia Spotlight [165] and TAGME [195]. For the latter, we have searched in wikidata with the extracted wikipedia entity using the same method as in 2. It must be noted that the performance of some other systems like those from [171] and [170] are better than ours by 6.2 and 7 % respectively; however, we haven't baselined against them because those are trained explicitly on the WebQSP data.

4.3.6 Discussion

Ablation Study

Finally, we do an ablation study where we remove some parts of the proposed model and observe the performance of entity linking for $n = 300$. The results are in Table 4.17. As observed, the entity-linking accuracy from not training the relation linker are at par with [6] in Table 4.17. The gating mechanism adds 3.97 %, because doing only a mean from the entity and relation prediction similarity scores would add in extra information overhead for the candidate selection for wrongly classified relation. The proposed soft-loss aids in 0.43 % increase in entity-linking accuracy and the candidate re-ranking improves it by 1.04%.

Table 4.17: Ablation Study

Approach	Entity-linking
Removing L_{rel} from total loss	67.55
Removing gating mechanism g_{amb}	74.63
Removing soft-loss from total loss	78.17
Without re-ranking candidates	77.56
Our Best Model: ELiDi (n=300)	78.60

Table 4.18: Span Detection Error (Green - correct span, blue - detected span).

what 's a rocket that has been flown
who is a swedish composer
what 's the name of an environmental disaster in italy
which korean air flight was in an accident

Additionally, as discussed previously, since relation classification systems are not able to generalize to unseen relations as observed in the works by [184], we do an ablation study on the relation classification task using a vanilla BiLSTM [6] and from the output of our relation detector 4.3.2. Our model scores 55.2 % on the SQB dataset [184] compared to 34.2 % for the former. This proves that using Knowledge Graph embeddings can help the model generalize better to unseen data.

Quantitative and Error Analysis

We do a quantitative analysis from the results of our best model with $n=300$. Percentage of questions with soft-disambiguity is 21.1 % and with hard-ambiguity is 18.51 %. Our model is able to predict 84.81 % of correct entity candidates for soft-disambiguation cases, out of which 75.02 % of times the correct relation was identified and 9.78 % the model predicted the wrong relation but the correct candidate is picked using our proposed KG embeddings based method; which proves that our intuition for using KG embeddings for the final task can be beneficial. For hard-ambiguity cases, the model was able to predict the correct candidate with an accuracy of 35.66 % (1432 out of 4015 cases), out of which the model predicted wrong relations 4.4 % of cases. But, it should be noted that there are no explicit linguistic signals to solve hard-disambiguity, following previous works we are predicting these cases based solely on candidate importance.

The model is able to predict the correct candidate 97.70 % of the times for cases where no disambiguation is required. Out of the 440 such wrongly classified candidates, 165 cases are because the true entity and correct relation are not connected in the KG at 1-hop, 162 because the entity span detector was not able to predict the correct span and the rest for wrong prediction in the disambiguate gating mechanism.

For some of the cases where the entity-span detector has failed to identify the correct entity is in table 4.18. In such cases, there are more than 1 entity in the question. Hence, it is difficult for the entity span detector to detect the correct entity.

For the final question-answering task, as mentioned previously, although the end-to-end accuracy for [6] is better than ours, but the task of question answering is particularly challenging in this case because we do not use any scores from string matching based methods such as Levenshtein distance for entity linking as done as an additional post-processing step by [6], especially in cases where the entity candidates and the entity mention in the question consists of out-of-vocabulary words. Also, for some cases, it is challenging to disambiguate between the predicted relations because there are no explicit linguistic signals available. To exemplify, let us consider the question *what county is sandy balls near ?*. The predicted relation for this question by our model is "fb:location.location.containedby", while the true relation in the dataset is "fb:travel.tourist_attraction.near_travel_destination".

Moreover, for the zero-shot entity linking task, the model is often not able to handle entity labels with punctuation such as for the extracted entity span *james k polk* the corresponding correct wikidata entity has label *james k. polk*. Additionally for WebQSP, there are often more than one entity mention in the question, but ELiDi always predicts single entity as output.

Improving Retrieval-Based Chatbots with added Domain Knowledge.

5.1 Introduction

In the previous chapter, we focus on answering the first research question, where we propose several techniques for question answering using structured knowledge graphs as a source. In this chapter we move to chatbots and focus on answering the second research question: "Can Domain Knowledge in the form of Unstructured Text improve Chatbots' performance?". A paper about this is published in SIGNLL conference on computational natural language learning [196]. The author has devised the main algorithm, did the implementation, carried out the evaluations and co-wrote the paper.

As also previously discussed, in a conversation scenario, a dialogue system can be applied to the task of freely generating a new response or to the task of selecting a response from a set of candidate responses based on the previous utterances, i.e. the context of the dialogue. The former is known as *generative* dialogue system while the latter is called *retrieval-based* (or response selection) dialogue system.

Both approaches can be realized using a modular architecture, where each module is responsible for a certain task such as natural language understanding, dialogue state-tracking, natural language generation, or can be trained in an end-to-end manner optimized on a single objective function.

Previous work, belonging to the latter category, by [146] applied neural networks to multi-turn response selection in conversations by encoding the utterances in the context as well as the possible responses with a Long Short-term Memory (LSTM) [41]. Based on the context and response encodings, the neural network then estimates the probability for each response to be the correct one given the context. More recently, several enhanced architectures have been proposed that build on the general idea of encoding response and context first and performing some embedding-based matching after [148–151].

Although such approaches result in efficient text-pair matching capabilities, they fail to attend over logical consistencies for longer utterances in the context, given the response. Moreover, in domain specific scenarios, a system's ability to incorporate additional domain knowledge can be very beneficial, e.g. for the example shown in Table 5.1. In this chapter, we propose a novel neural network architecture for multi-turn response-selection that extends the model proposed by [146]. Our major contributions are: (1) a neural network paradigm that is able to attend over important words

Context
Utterance 1: My networking card is not working on my Ubuntu, can somebody help me?
Utterance 2: What's your kernel version? Run <i>uname -r</i> or <i>sudo dpkg -l grep linux-headers grep ii awk '{print \$3}'</i> and paste the output here.
Utterance 3: It's 2.8.0-30-generic.
Utterance 4: Your card is not supported in that kernel. You need to upgrade, that's like decade old kernel!
Utterance 5: Ok how do I install the new kernel??
Response
Just do <i>sudo apt-get upgrade</i> , that's it.

Table 5.1: Illustration of a multi-turn conversation with domain specific words (UNIX commands) in italics.

in a context utterance given the response encoding (and vice versa), (2) an approach to incorporate additional domain knowledge into the neural network by encoding the description of domain specific words with a GRU and using a bilinear operation to merge the resulting domain specific representations with the vanilla word embeddings, and (3) an empirical evaluation on a publicly available multi-turn dialogue corpus showing that our system outperforms all other state-of-the-art methods for response selection in a multi-turn setting.

5.2 Background

In this section, we will explain the task at hand and give a brief introduction to the neural network architectures our proposed model is based on.

5.2.1 Problem definition

Let the data set $\mathcal{D} = \{(c_i, r_i, y_i)\}_{i=1}^M$ be a set of M triples consisting of context c_i , response r_i , and ground truth label y_i . Each context is a sequence of utterances, that is $c_i = \{u_{il}\}_{l=1}^L$, where L is the maximum context length. We define an utterance as a sequence of words $\{w_{it}\}_{t=1}^T$. Thus, c_i can also be viewed as a sequence of words by concatenating all utterances in c_i . Each response r_i is an utterance and $y_i \in \{0, 1\}$ is the corresponding label of the given triple which takes a value of 1 if r_i is the correct response for c_i and 0 otherwise. The goal of retrieval-based dialogue systems is then to learn a predictive distribution $p(y|c, r, \theta)$ parameterized by θ . That is, given a context c and response r , we would like to infer the probability of r being a response to context c .

5.2.2 RNNs, BiRNNs and GRUs

Recurrent neural networks are one of the most popular classes of models for processing sequences of words $W = \{w_t\}_{t=1}^T$ with arbitrary length $T \in \mathbb{N}$, e.g. utterances or sentences. Although, explained in section 2, we are revisiting the concepts of RNNs for continuity and notation purposes. Each word w_t is first mapped onto its vector representation \mathbf{w}_t (also referred to as word embedding), which serves as input to the RNN at time step t . The central element of RNNs is the recurrence relation of its hidden units, described by

$$\vec{\mathbf{h}}_t = f(\vec{\mathbf{h}}_{t-1}, \mathbf{w}_t | \phi) , \quad (5.1)$$

where ϕ are the parameters of the RNN and f is some nonlinear function. Accordingly, the state $\vec{\mathbf{h}}_t$ of the hidden units at time step t depends on the state $\vec{\mathbf{h}}_{t-1}$ in the previous time step and the t -th word in the sequence. This way, the hidden state $\vec{\mathbf{h}}_T$ obtained after T updates contains information about the whole sequence W , and can thus be regarded as an embedding of the sequence.

The RNN architecture can also be altered to take into account dependencies coming from both the past and the future by adding an additional sub-RNN that moves backward in time, giving rise to the name bi-directional RNN (BiRNN). To achieve this, the network architecture is extended by an additional set of hidden units. The states $\overleftarrow{\mathbf{h}}_t$ of those hidden units are updated based on the current input word and the hidden state from the next time step. That is for $t = 1, \dots, T - 1$:

$$\overleftarrow{\mathbf{h}}_{T-t} = f(\overleftarrow{\mathbf{h}}_{T-t+1}, \mathbf{w}_{T-t} | \phi) . \quad (5.2)$$

Here, the words are processed in reverse order, i.e. w_T, \dots, w_1 , such that $\overleftarrow{\mathbf{h}}_T$ (analogous to $\vec{\mathbf{h}}_T$ in the forward directed RNN) contains information about the whole sequence. At the t -th time step, the model's hidden representation of the sequence is then usually obtained by the concatenation of the hidden states from the forward and the backward RNN, i.e. by $\mathbf{h}_t = [\vec{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t]$ and the embedding of the whole sequence W is given by $\mathbf{h}^W = [\vec{\mathbf{h}}_T, \overleftarrow{\mathbf{h}}_T]$.

Modeling very long sequences with RNNs is hard: [40] showed that RNNs suffer from vanishing and exploding gradients, which makes training over long-term dependency difficult. Such problems can be addressed by augmenting the RNN with additional gating mechanisms, as it is done in LSTMs and the Gated Recurrent Unit (GRU) [197]. These mechanisms allow the RNN to learn how much to update the hidden state flexibly in each step and help the RNN to deal with the vanishing gradient problem in long sequences better than vanilla RNNs. The gating mechanism of GRUs is motivated by that of LSTMs, but is much simpler to compute and implement. It contains two gates, namely the reset and update gate, whose states at time t are denoted by \mathbf{z}_t and \mathbf{r}_t , respectively. Formally, a GRU is defined by the following update equations

$$\begin{aligned} \mathbf{z}_t &= \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}) , \\ \mathbf{r}_t &= \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}) , \\ \tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h \mathbf{r}_t \odot \mathbf{h}_{t-1}) , \\ \mathbf{h}_t &= \mathbf{z}_t \odot \tilde{\mathbf{h}}_t + (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} , \end{aligned}$$

where \mathbf{x}_t is the input (corresponding to \mathbf{w}_t in our setting) and the set of weight matrices $\phi = \{\mathbf{W}_z, \mathbf{U}_z, \mathbf{W}_r, \mathbf{U}_r, \mathbf{W}_h, \mathbf{U}_h\}$ constitute the learnable model parameters.

5.2.3 Dual Encoder

Recurrent neural networks and their variants have been used in many applications in the field of natural language processing, including retrieval-based dialogue systems. In this area the dual encoder (DE) [146] became a popular model. It uses a single RNN encoder to transform both context and response into low dimensional vectors and computes their similarity. More formally, let \mathbf{h}^c and \mathbf{h}^r be the encoded context and response, respectively. The probability of r being the correct response for c is then computed by the DE as

$$p(y|c, r, \theta) = \sigma((\mathbf{h}^c)^T \mathbf{M} \mathbf{h}^r + b) , \quad (5.3)$$

where $\theta = \{\phi, \mathbf{M}, b\}$ (recall, that ϕ is the set of parameters of the encoder RNN that outputs \mathbf{h}^c and \mathbf{h}^r) is the set of parameters of the full model and σ is the sigmoid function. Note, that the same RNN is used to encode both context and response.

In summary, this approach can be described as first creating latent representations of context and response in the same vector space and then using the similarity between these latent embeddings (as induced by matrix \mathbf{M} and bias b) for estimating the probability of the the response being the correct one for the given context.

5.3 Model description

Our model extends the DE described in Section 5.2.3 by two attention mechanisms which make the context encoding response-aware and vice versa. Furthermore, we augment the model with a mechanism for incorporating external knowledge to improve the handling of rare words. Both extensions are described in detail in the following subsections.

5.3.1 Attention augmented encoding

As described above, in the DE context and response are encoded independently from each other based on the same RNN. Instead of simply taking the final hidden state \mathbf{h}^c (and \mathbf{h}^r) of the RNN as context (and response) encoding, we propose to use a response-aware attention mechanism to calculate the context embedding and vice versa.

Subsequently, we will describe this mechanism formally. Recall that a context c can be seen as sequence of words $\{w_t^c\}_{t=1}^T$ where all utterances are concatenated and T is the total number of words in the context. Given this sequence, the RNN (in our experiments a bi-directional GRU) produces a sequence of hidden states $\mathbf{h}_1^c, \dots, \mathbf{h}_T^c$ and an encoding of the whole context sequence \mathbf{h}^c as described in Section 5.2.2. Analogously, we get $\mathbf{h}_1^r, \dots, \mathbf{h}_{T'}^r$ and \mathbf{h}^r for a response consisting of a sequence of words $\{w_t^r\}_{t=1}^{T'}$, where T' is the total number of words in the response.

For calculating the response-aware context encoding, we first estimate attention weights α_t^c for the hidden state \mathbf{h}_t^c in each time step, depending on the response encoding \mathbf{h}^r :

$$\alpha_t^c \propto \exp((\mathbf{h}_t^c)^T \mathbf{W}_c \mathbf{h}^r) , \quad (5.4)$$

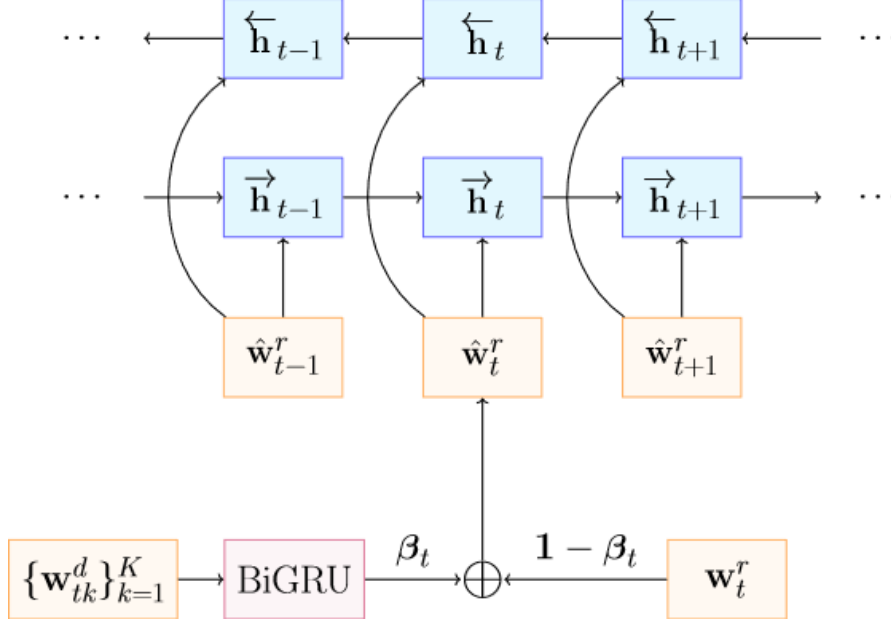


Figure 5.1: Our proposed way to incorporate domain knowledge into the model. β_t and $\mathbf{1} - \beta_t$ represent the (multiplicative) weights for the description embedding and the word embedding respectively. The resulting combination, $\hat{\mathbf{w}}_t^r$ acts as an input of the encoder.

where \mathbf{W}_c is a learnable parameter matrix. The response-aware context embedding then is given by

$$\hat{\mathbf{h}}^c = \sum_{t=1}^T \alpha_t^c \mathbf{h}_t^c . \quad (5.5)$$

Intuitively this means, that depending on the response we focus on different parts of the context sequence, for judging on how well the response matches the context. This may resemble human focus.

Similarly, we calculate the context-aware response encoding by

$$\hat{\mathbf{h}}^r = \sum_{t=1}^T \alpha_t^r \mathbf{h}_t^r , \quad (5.6)$$

with attention weights

$$\alpha_t^r \propto \exp((\mathbf{h}_t^r)^T \mathbf{W}_r \mathbf{h}^c) . \quad (5.7)$$

The two attention-weighted encodings (for response and context, respectively) then replace the vanilla encodings in equation (5.3), that is

$$p(y|c, r, \theta) = \sigma((\hat{\mathbf{h}}^c)^T \mathbf{M} \hat{\mathbf{h}}^r + b) . \quad (5.8)$$

5.3.2 Incorporating domain keyword descriptions

[198] proposed a method for learning embeddings for OOV words based on external dictionary definitions. They learn these description embeddings of words using an LSTM for encoding the corresponding definition. If a particular word included in the dictionary also appears in the corpus' vocabulary (for which vanilla word embeddings are given), they add the word embedding and the description embedding together. Otherwise, in the case of OOV words, they use solely the description embedding in place of the missing word embedding. Inspired by this approach, we use a similar technique to incorporate domain keyword descriptions into word embeddings.

If a word w_t^r in the response utterance is in the set of domain keywords \mathcal{K} , we firstly extract its description. The description of w_t^r is a sequence of words $\{w_{tk}^d\}_{k=1}^K$, which is projected onto sequence of embeddings $\{\mathbf{w}_{tk}^d\}_{k=1}^K$. This sequence is encoded using another bi-directional GRU to obtain a vector representation \mathbf{h}_t^d of the same dimension as the vanilla word embeddings. If w_t^r is not in \mathcal{K} , we simply set \mathbf{h}_t^d to zero. We call \mathbf{h}_t^d the *description embedding*.

Some domain specific words might also happen to be common words. For instance, in the case of the UDC's vocabulary, there exist tokens such as *shutdown*¹ or *who*², which are ambiguous, i.e., although they are valid UNIX commands, they are also common words in natural language. The description embeddings of domain specific words can be simply added to the vanilla word embeddings as suggested by [198]. However, it might be advantageous if the model can determine itself whether to treat the current word as a domain specific word, a common word, or something in between, depending on the context. For instance, if the context is mainly talking about system users, then *who* is most likely a UNIX keyword. Therefore, we propose a more flexible way to combine the description embedding \mathbf{h}_t^d and the word embedding \mathbf{w}_t^r , that is, we define the final word embedding to be a convex combination of both, and let the combination coefficients be given by a function of \mathbf{h}_t^d and the context embedding $\hat{\mathbf{h}}^c$. Intuitively, this allows the model to flexibly focus on the description or the vanilla embedding, in dependence on the context and the description. Formally, the combination coefficients β_t of t-th word in the response is given by

$$\beta_t \propto \exp(\mathbf{U}^T \hat{\mathbf{h}}^c + \mathbf{V}^T \mathbf{w}_t^r) , \quad (5.9)$$

where \mathbf{U} and \mathbf{V} are learnable parameter matrices. Note that β_t is a vector of the same dimension as the embeddings. The final embedding of w_t^r (which serves as input to the response encoder) is then the weighted sum

$$\hat{\mathbf{w}}_t^r = \beta_t \odot \mathbf{h}_t^d + (\mathbf{1} - \beta_t) \odot \mathbf{w}_t^r , \quad (5.10)$$

where \odot denotes the element wise multiplication.

¹UNIX command for system shutdown.

²UNIX command to get a list of currently logged-in users.

5.4 Experiment

5.4.1 Ubuntu multi-turn dialogue corpus

Extending the work of [199], [146] introduced a version of the Ubuntu chat log conversations which is the largest publicly available multi-turn, dyadic, and domain-specific dialogue data set. The chats are extracted from Ubuntu related topic specific chat rooms in the Freenode Internet Relay Chat (IRC) network. Usually, experienced users address a problem of someone by suggesting a potential solution and a *name mention* of the addressed user. A conversation between a pair of users often stops when the problem has been solved. However, they might continue having a discussion which is not related to the topic.

A preprocessed version of the above corpus and the needed vocabulary are provided by [200]. The preprocessing consisted of replacing numbers, URLs, and system paths with special placeholders as suggested by [19]. No additional preprocessing is performed by us. The data set consists of 1 million training triples, 500k validation triples, and 500k test triples. One half of the 1 million training triples are positive (triples with $y = 1$, i.e. the provided response fits the context) the other half negative (triples with $y = 0$). In contrast, in the validation and test set, for every context c_i , there exists one positive triple providing the ground-truth response to c_i and nine negative triples with unbecoming responses. Thus, in these sets, the ratio between positive and negative triples per context is 1:9 which makes evaluating the model with information retrieval metrics such as Recall@k possible (see Section 5.5).

Model	$R_2@1$	$R_{10}@1$	$R_{10}@3$	$R_{10}@5$
DE-RNN [147]	0.768	0.403	0.547	0.819
DE-CNN [147]	0.848	0.549	0.684	0.896
DE-LSTM [147]	0.901	0.638	0.784	0.949
DE-BiLSTM [147]	0.895	0.630	0.780	0.944
MultiView [149]	0.908	0.662	0.801	0.951
DL2R [148]	0.899	0.626	0.783	0.944
r-LSTM [19]	0.889	0.649	0.857	0.932
MV-LSTM [100]	0.906	0.653	0.804	0.946
Match-LSTM [153]	0.904	0.653	0.799	0.944
QA-LSTM [154]	0.903	0.633	0.789	0.943
SMN _{dyn} [200]	0.926	0.726	0.847	0.961
CCN [150]	-	0.727	0.858	0.971
ESIM [151]	-	0.734	0.854	0.967
AK-DE-biGRU (Ours)	0.933	0.747	0.868	0.972

Table 5.2: Evaluation results of our models compared to various baselines on Ubuntu Dialogue Corpus.

5.4.2 Model hyperparameters

We chose a word embedding dimension of 200 as done by [200]. We use fastText [201] to pre-train the word embeddings using the training set instead of using off-the-shelf word embeddings, following [200]. We set the hidden dimension of our GRU to be 300, as in the work of [146]. We restricted the sequence length of a context by a maximum of 320 words, and that of the response by 160. Because of the resulting size of the model and limited GPU memory, we had to use a smaller batch size of 32. We optimize the binary cross entropy loss of our model with respect to the training data using Adam [159] with an initial learning rate of 0.0001. We train our model for a maximum of 20 epochs as according to our experience, this is more than enough to achieve convergence. The training is stopped when the validation recall does not increase after three subsequent epochs. The test set is evaluated on the model with the best validation recall.

For the implementation, we use PyTorch [202]. We train the model end-to-end with a single 12GB GPU. The implementation³ of our models along with the additional domain knowledge base⁴ are publicly available.

5.5 Results

Following [146] and [147], we use the Recall@k evaluation metric, where $R_n@k$ corresponds to the fraction of examples for which the correct response is under the k best out of a set of n candidate responses, which were ranked according to their probabilities under the model.

In our evaluation specifically, we use $R_2@1$, $R_{10}@1$, $R_{10}@3$, and $R_{10}@5$.

5.5.1 Comparison against baselines

We compare our model, which we refer to as *Attention and external Knowledge augmented DE with bi-directional GRU (AK-DE-biGRU)*, against models previously tested on the same data set: the basic DE models analyzed by [146] and [147] using different encoders, such as convolutional neural network (DE-CNN), LSTM (DE-LSTM) and bi-directional LSTM (DE-BiLSTM); the **Multi-View**, **DL2R** and **r-LSTM** models proposed by [149], [148] and [19], respectively; architectures for advanced context and response matching, namely **MV-LSTM** [100], **Match-LSTM** [153], and **QA-LSTM** [154]; architectures processing the context utterances individually, namely **SMN_{dyn}** [200] and **CCN**; and we also use recently proposed **ESIM** [151] as a baseline.

The results are reported in Table 5.2. Our model outperforms all other models used as baselines. The largest improvement of our model compared to the best of the baselines (i.e. ESIM in general and SMN_{dyn} for $R_2@1$ metric) are with respect to the $R_{10}@1$ and $R_{10}@3$ metric, where we observed absolute improvements of 0.013 and 0.014 corresponding to 1.8% and 1.6% relative improvement, respectively. For $R_2@1$ and $R_{10}@5$ we observed more modest improvements of 0.007 (0.8%) and 0.005 (0.5%), respectively. Our results are significantly better with $p < 10^{-6}$ for a one-sample one-tailed t-test compared to the best baseline (ESIM), on $R_{10}@1$, $R_{10}@3$, $R_{10}@5$ metrics, using the outcome of 15 independent experiments. The variance between different trials is smaller than 0.001 for all evaluation metrics.

³<https://github.com/SmartDataAnalytics/AK-DE-biGRU>.

⁴Command descriptions scraped from Ubuntu man pages.

Model	$R_{10}@1$	$R_{10}@3$	$R_{10}@5$
DE-GRU	0.685	0.831	0.960
DE-biGRU	0.678	0.813	0.956
A-DE-GRU	0.712	0.845	0.964
A-DE-biGRU	0.739	0.864	0.968
AK ₊ -DE-biGRU	0.743	0.867	0.969
AK-DE-biGRU _{w2v}	0.745	0.866	0.970
AK-DE-biGRU	0.747	0.868	0.972

Table 5.3: Ablation study with different settings.

5.5.2 Ablation study

Our model differs in various ways from the vanilla DE: it uses a GRU instead of an LSTM for the encoding, introduces an attention mechanism for the encoding of the context and another for the encoding of the response, and incorporates additional knowledge in the response encoding process.

To analyze the effect of these components on the overall performance, we analyzed different model variants: a DE using a GRU or a bi-directional GRU as encoder (**DE-GRU** and **DE-biGRU**, respectively) and both of these models with attention augmented encoding for embedding both context and response (**A-DE-GRU** and **A-DE-biGRU**, respectively). We also tested the effects of using a simple addition instead of the weighted summation given in equation (5.10) for merging the word embedding with the description embedding (**AK₊-DE-biGRU**). Finally, we investigated a version of our model (**AK-DE-biGRU_{w2v}**) where we used pre-trained word2vec embeddings, as done by [200], instead of learning our own word embeddings from the data set.

The results of the study are presented in Table 8.6(b). With the basic models, i.e. DE-GRU and DE-biGRU, as baselines, we observed around 4% and 9% improvement on $R_{10}@1$ when incorporating the attention mechanism (A-DE-GRU and A-DE-biGRU, respectively).

When domain knowledge is incorporated by simple addition (as in the work of [198]), i.e. in AK₊-DE-biGRU, we noticed 0.5% further improvement. Note however, that the results are not as good as when using the proposed weighted addition. Finally, using our method of incorporating domain knowledge in combination with embeddings trained from scratch with fastText [201], the performance gets 0.3% better than when using pretrained word2vec embeddings. In total, compared to the DE-biGRU baseline, our model (AK-DE-biGRU) achieves 10% of improvement in terms of the $R_{10}@1$ metric. Thus, the results clearly suggest that both the attention mechanism and the incorporation of domain knowledge, are effective approaches for improving the dual encoder architecture. Curiously, we noticed that for the baseline models, using a GRU as the encoder is better than using a biGRU. This finding is in line with the results from [147] reported in Table 5.2. However, the table is turned when augmenting the models with an attention mechanism where the biGRU-based model outperforms the one with the GRU. This observation motivates us to consider a biGRU instead of a GRU in our final model.

5.5.3 Visualizing response attentions

To further investigate the results given by our model, we qualitatively inspected several samples of response utterances and their attention weights, as shown in Table 5.4.

Example Response Utterances
gui for shutdown try typing <code>sudo shutdown -h</code> now
<code>sudo apt-get install qt4-designer</code> there could be some qt dev packages too but i think the above will install them as dependencies
certainly won't make a difference i'm sure but maybe try <code>sudo shutdown -r</code> now <code>shutdown</code> works just fine graphical and command line
pci can you put the output of <code>lspci</code> on <code>__url__</code> and give me the link please i do n't see a line in <code>xorg</code> conf for <code>hsync</code> and <code>vsync</code> do you get the same you'd create it i'm looking at <code>gentoo</code> and <code>ubuntu</code> forums a sec
can be many reasons of <code>traceroute</code> <code>__url__</code> you will not get a complete result

Table 5.4: Visualization of attention weight in utterance samples, darker shade means higher attention weight.

Context utterances
Utterance 1: Ubuntu <version>
Utterance 2: hi all sony vaio fx120 will not turn off when shutting down, any ideas? btw acpi=off in boot parameters anything else i should be trying?
Utterance 3: how are you shutting down i.e. terminal or gui?

Table 5.5: Sample context utterances from UDC's test set whose correct response is the first utterance in Table 5.4.

We noticed that our model learned to focus on technical terms, such as `lspci`, `shutdown`, and `traceroute`. We also observed that the model is able to capture contextual importance, i.e. it is able to focus on context relevant words. For example, given the context in Table 5.5 and the correct response in the first row of Table 5.4, one can see the attention on the word `shutdown`, where it gets a lower weight when used as a common word in the first occurrence than as a UNIX command in the second.⁵

5.5.4 Error analysis

We qualitatively analyzed our model's errors. We observed that our model's predictions are biased toward high information utterances. That is, we observed for some examples that the correct response is generic (i.e. has low information), our model chooses a non-generic response, as shown in Table 5.6. Furthermore, we computed the average utterance information content (the entropy) for both the correct and predicted responses, based on [203], where we obtained 9.25 bits and 9.34 bits, respectively. This quantitatively indicates that our model is slightly biased toward high information responses.

⁵N.B. The conversations are taken directly from the corpus and can be grammatically inconsistent.

Examples of model error:

Correct: ok will do :), nope.

Predicted: __url__ if you go down to the bottom of that tutorial i also have a post there that is a bit more detailed about my problem poster name is trent

Correct: hmm! ok

Predicted: as did i w/ fbsd ... just check out the livedcd for a bit

Correct: okay thank you a thread i hope :)

Predicted: hmm ok because im not sure about iwconfig and wpa but we can give it a try do gksudo gedit __path__ then add a record like this __url__

Correct: right .. it is, it exists i verified

Predicted: i want to connect to your computer remotely if you allow me to so i can fix the problem for you just follow the following procedure.

Correct: roger .. lemme check, got it ... thanks dude :)

Predicted: just click the partition and then click the blue text next to mount point or you can simply navigate to that path

Table 5.6: Examples on the error our model made. We observed that our model's predictions are biased towards non-generic responses.

Improving task-oriented, generative Chatbots using Knowledge Graph Embeddings

6.1 Introduction

After dealing with unstructured knowledge sources in the last chapter, we are moving back to structured knowledge sources and trying to answer the third research question "Can Jointly-Trained KG and Word Embeddings improve Task-Oriented Dialogue Systems?". Here, we propose end-to-end neural network-based model which incorporates knowledge graphs into the response generation process using novel loss functions. The methods proposed is published as a research paper [134] in the Extended semantic web conference. The author proposed and implemented on the loss functions proposed in this work and also co-wrote the paper.

Last decade has seen a considerable rise in the need for effective task-oriented dialogue agents that can help the user to achieve specific goals and everyday tasks such as weather forecast assistance, schedule arrangement, and location navigation. Neural network based dialogue models proved to be the most promising architectures so far which can effectively use dialogue corpora in order to build such agents. However, these models struggle to reason over and incorporate state-full knowledge while preserving their end-to-end text generation functionality. They often require additional user supervision or additional data annotation to incorporate some dialogue-state information along with additional knowledge.

To help the dialogue agents overcome these problems, they have often been built with several pipelined modules such as language understanding, dialog management, question answering components, and natural language generation [10]. However, modeling the dependencies between such modules is complex and may not result in very natural conversations.

The problem becomes much more complicated in multi-domain scenarios, because the text generation in such scenarios hugely depends on the domain the user is talking about, where each domain will have its own vocabulary that differs from other domains. Hence, understanding user intent during the text generation process can be beneficial. Table 6.1 shows sample user utterances along with its intent. While generating new tokens, the model can benefit from this because separate intents will follow different vocabulary distributions.

User utterance	Intent
Book a seat for me in any good Chinese restaurant nearby	Restaurant Booking
Book a flight ticket for me from Berlin to Paris for tomorrow	Flight Booking
Please show me the direction to the main train station	Navigation
Please cancel my appointment with the dentist for tomorrow	Scheduling

Table 6.1: User query and respective intents

In order to tackle the aforementioned problems, we propose a novel neural network architecture trained using a multi-task learning paradigm, which along with generating tokens from the vocabulary also tries to predict the intent of the user utterances. By doing so, the model is able to relate the generated words to the predicted intent and uses this to exclude words unrelated to the current conversation from the next word candidate prediction. Our model utilizes joint text and Knowledge Graph embeddings as inputs to the model inspired by the works of [204].

Additionally, after projecting the entities and text into the same vector space, we propose an additional training criterion to use as a novel regularization technique called entity loss, which further penalizes the model if the predicted entity is different from the original. The loss is the mean of the vector distance between the predicted and correct entities.

Furthermore, to guarantee a state-full knowledge incorporation we included a Knowledge Graph (KG) key-value look-up to implement a way of knowledge tracking during the utterances of a dialogue.

A KG in the context of this thesis is a directed, multi-relational graph that represents entities as nodes, and their relations as edges, and can be used as an abstraction of the real world. KGs consists of triples of the form $(h,r,t) \in KG$, where h and t denote the head and tail entities, respectively, and r denotes their relation.

Our main contributions in this chapter are as follows:

- Utilizing joint text and Knowledge Graph embeddings into an end-to-end model for improving the performance of task-oriented dialogue systems.
- A multi-task learning of dialogue generation and learning user intent during decoding.
- A novel entity loss based regularization technique which penalizes the model if the predicted entity is further from the true entity in vector space.

6.2 Model Description

In this section, we present our model architecture. Firstly, we discuss the attention based RNN encoder-decoder architecture, and then we explain the intent predictor, joint embedding vector space and additional entity loss based regularization.

6.2.1 Attention Based Seq-to-seq Model

Our model is quintessentially based on an RNN-based encoder-decoder architecture taking inspirations from the works proposed by [129], [128] and [130]. Given the input utterances from the user in a dialogue scenario D , the system has to produce an output utterance o_t for each time-step t , i.e. there is

a sequence $\{(i_1, o_1), (i_2, o_2), \dots, (i_n, o_n)\}$ where n denotes the total number of utterances in a dialogue. We reprocess these utterances and present them as:

$\{(i_1, o_1), ([i_1; o_1; i_2], o_2), \dots, ([i_1; o_1; \dots; o_{n-1}; i_n], o_n)\}$, where we present an input as the sequence of all previous utterances concatenated with the current input. This is done in order to maintain the context in the dialogue.

Let x_1, \dots, x_m denote the words in each input utterance (where m is the length of the utterance). We firstly map these words to their embedding vector representations using Φ^{x_t} which can be either a randomly initialized or a pretrained embedding. Then these distributed representations are fed into the RNN-encoder (LSTM [41] in this case) as follows:

$$h_t = LSTM(\Phi^{x_t}, h_{t-1}) \quad (6.1)$$

$h_t = (h_0, \dots, h_m)$ represents the hidden states of the encoder.

These encoder inputs are fed into a decoder which is again a recurrent module. It generates a token for every time-stamp t given by the probability

$$p(y_t | y_{t-1}, \dots, y_1, x_t) = g(y_{t-1}, s_t, c_t) \quad (6.2)$$

Where, $g(\cdot)$ is a softmax function, s_t is the hidden state of the decoder at time-step t given by

$$s_t = LSTM(s_{t-1}, y_{t-1}, c_t) \quad (6.3)$$

Additionally, we use an attention mechanism over the encoder hidden-states as proposed by [47], [205]. The attention mechanism is calculated as

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^m \exp(e_{ik})} \quad (6.4)$$

$$e_{ij} = \tanh(W_c[s_{i-1}, h_j]) \quad (6.5)$$

The final weighted context from the encoder is given by

$$c_i = \sum_m \alpha_{ij} h_j \quad (6.6)$$

Where, α_{ij} represent the attention weights learned by the model, c_i the context vector, W_c a weight parameter that has to be learned by the model.

6.2.2 Predicting Intent

The intent predictor module is used to predict the intent from the user utterance. It is computed using the encoder LSTM's hidden representation h_t as computed at every timestamp t . The intent output prediction is given by:

$$i_{out} = W_o(\tanh(W_i[h_t; c_t])) \quad (6.7)$$

Where, i_{out} is the intent score $\in \mathbf{R}^{i_c}$ for the t^{th} time-step, W_i and W_o are the trainable weight parameters for the intent predictor and i_c is the total number of intent classes. The latter can be the

different domains the dialogue system wants to converse upon, for example restaurant booking, flight booking et cetera.

6.2.3 Training Joint Text and Knowledge Graph Embeddings

Learning distributed embeddings for words has been widely studied for neural network based natural language understanding. [58] was the first to propose a model to learn such distributed representations. In order to leverage additional information from Knowledge Graphs in goal-oriented dialogues, we need to first project the embeddings for both texts appearing in the dialogue corpus and Knowledge Graphs in the same vector space. For learning such word embeddings, we adopted the work done by [204], which proposes a model for training the embeddings from both a corpus and a Knowledge Graph jointly. The proposed method is described in three steps: firstly training the Global Vectors (GloVe) following [206], secondly incorporating the Knowledge Graph, and jointly learning the former two using the proposed model with a linear combination of both their objective functions.

Glove Vectors (GloVe)

Firstly, we train the embedding vectors by creating a global word co-occurrence matrix X from both the input and target sentences, where each word in these sentences is represented by a row in X containing the co-occurrence of context words in a given contextual window. Finally, the GloVe embedding learning method minimizes the weighted least squares loss presented as:

$$J_C = \frac{1}{2} \sum_{i \in V} \sum_{j \in V} f(X_{ij})(w_i^\top w_j + bi + bj - \log(X_{ij}))^2 \quad (6.8)$$

Where X_{ij} denote the total occurrences of target word w_i and the context word w_j and the weighting function f assigns a lower weight for extremely frequent co-occurrences to prevent their over-emphasis.

Incorporating the Knowledge Graph

The GloVe embeddings themselves do not take the semantic relation between corresponding words into account. Therefore, in order to leverage a Knowledge Graph while creating the word embeddings, we define an objective J_S that only considers the three-way co-occurrences between a target word w_i and one of its context word w_j along with the semantic relation R that exists between them in the KG. The KG-based objective is defined as follows:

$$J_S = \frac{1}{2} \sum_{i \in V} \sum_{j \in V} R(w_i, w_j)(w_i - w_j)^2 \quad (6.9)$$

Where $R(w_i, w_j)$ indicates the strength of the relation R between w_i and w_j , which assigns a lower weight for extremely frequent co-occurrences to prevent over-emphasising such co-occurrences and if there is no semantic relation between w_i and w_j then $R(w_i, w_j)$ will be set to zero.

For the final step, to train both the corpus and the KG objectives jointly. We define the combined objective function J as their linearly weighted combination

$$J = J_C + \lambda J_S \quad (6.10)$$

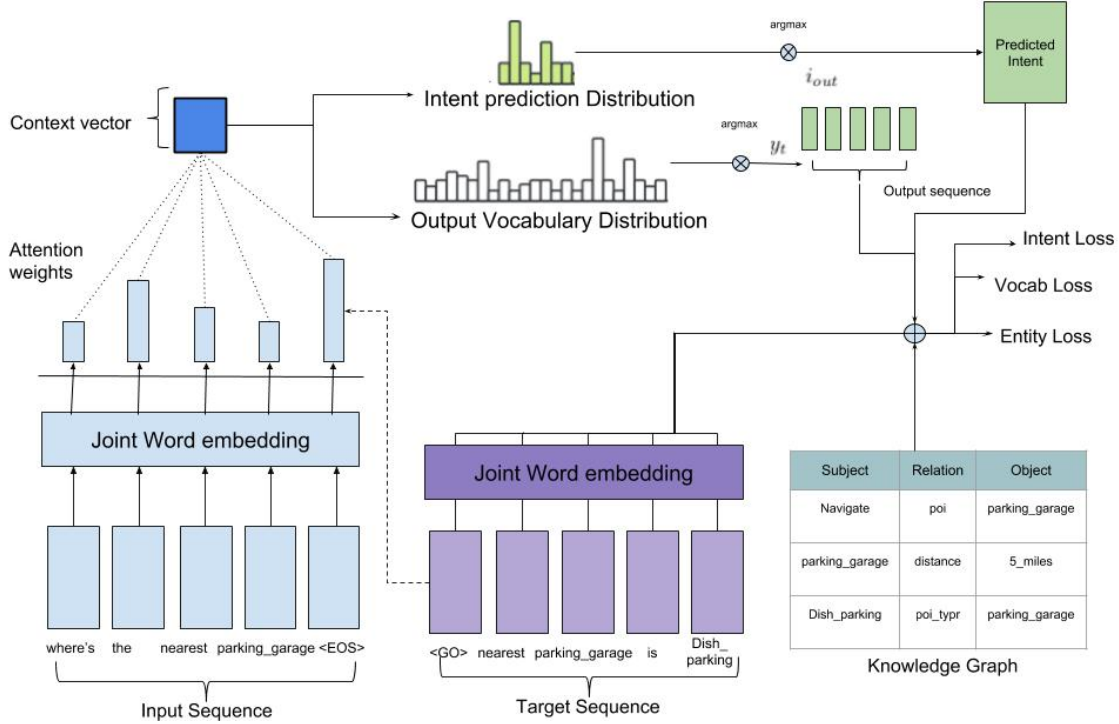


Figure 6.1: Sequence-to-Sequence based module with multi-task learning of user intent regularized with additional Entity Loss

λ is the regularization coefficient that regulates the influence of the Knowledge Graph on those learned from the corpus.

6.2.4 Regularizing using additional Entity Loss

After projecting the Knowledge Graphs and the text in the same vector space, we compute an additional loss which is equal to the vector distances between the predicted entity and correct entity. The intuition behind using this loss is to penalize the model for predicting the wrong entity during decoding, in vector space. We use cosine distance to measure the entity loss as given by:

$$\text{Entity}_l = 1 - \frac{\phi^{e_{corr}} \cdot \phi^{e_{pred}}}{\|\phi^{e_{corr}}\| \|\phi^{e_{pred}}\|} \quad (6.11)$$

$\phi^{e_{corr}}$ and $\phi^{e_{pred}}$ being the vector embeddings for the correct and predicted entities, respectively.

6.2.5 Final objective Function

The final objective function for training the model is the summation of the cross-entropy (CE) loss from the decoder (Vocab_l), the cross-entropy loss from the intent predictor (Intent_l) and the entity

loss (Entity_l).

The total loss is given by:

$$L_{tot} = \text{Vocab}_l + \text{Intent}_l + \text{Entity}_l \quad (6.12)$$

The model is trained with back-propagation using Adam [159] optimizer.

6.2.6 Key-Value Entity Look-up

As mentioned before the Knowledge Graph (KG) can change from one dataset to another or even as in our case from one dialogue to another. This will result in the generation of KG entities that do not belong to the current dialogue. Since we are incorporating global knowledge information using the joint embedding, this additional step would ensure local KG integration aiding in the improvement of the model. To accomplish this, we added an additional lookup step during decoding. While generating a new token during decoding at time-step t , we firstly check if the predicted token is an entity. If it's an entity, we first do a lookup into the local dialogue KG to check its presence. If the entity is not present, then we pick the one with the entity with the highest softmax probability that is present in the local KG using greedy search. The technique is illustrated in figure 6.2. As seen in the figure, for the query *what is the weather like in New York, today?*, during decoding at $t = 2$, the model outputs a softmax distribution with highest probability¹ (0.08) for the token *raining* after predicting *it is*. We keep a copy of all the global KG entities and first check whether the predicted token is an entity in the global KG or not. Since *raining* is indeed an entity, we further do a look up into the local dialogue KG if it exists. For this scenario it doesn't, hence we do a greedy search for the next best-predicted tokens which are present in the local KG. In this specific case, *sunny* is the next most probable entity that exists in the local KG, We pick this entity for the time-step and feed it into the next. The final output will be *it is sunny today in New York*.

One such example in a real-case setting is shown in table 6.6. The distance for the gas station was generated as *2_miles*, whereas we can observe that in the local KG, the correct distance entity is *5_miles*. Hence by performing the additional entity lookup, we replace *2_miles* with the correct entity during decoding. Empirical evidence suggests that this strategy can gain in performance as depicted in table 8.6(b).

6.3 Experiments

6.3.1 Dataset

To test our hypothesis of multi-task learning of intent and dialogues, we needed a dataset which has multiple domains and user intents annotated along with a Knowledge Graph for task-oriented dialogues. [11] introduced such a multi-turn, multi-domain task-oriented dialogue dataset. They performed a Wizard-of-Oz based data collection scheme inspired by [123]. They used the Amazon Mechanical Turk (AMT) platform for data collection. The statics of the datasets are mentioned in Table 6.2. The dataset is a conversation workflow in an in-car setting where the driver (user) is asking the assistant for scheduling appointments, weather forecasts, and navigation. Both roles are played by AMT turkers

¹this is a fictitious example for explaining the algorithm, the scores are not what is being predicted from the real case scenarios

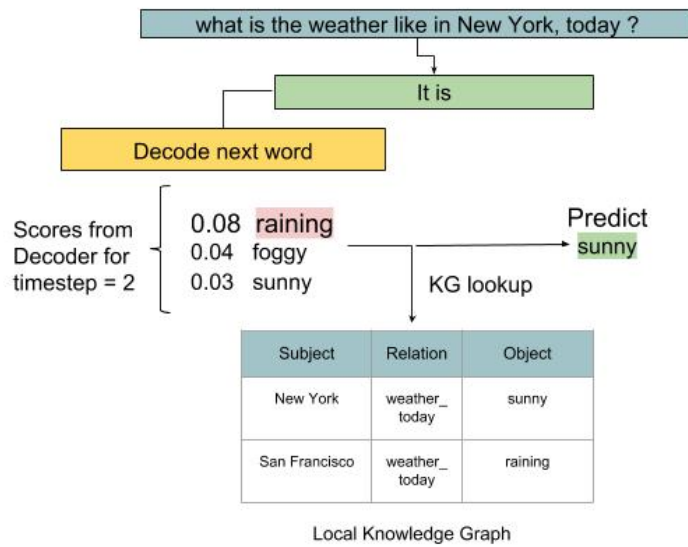


Figure 6.2: An example of how KVL works to replace predicted entities with correct ones from the local KG.

(workers). In driver mode, the turker asked a question while the assistant turker responds to the query using the provided Knowledge Graph. The Knowledge graph statistics are provided in Table 6.3.

Training Dialogues	2,425
Validation Dialogues	302
Test Dialogues	304
Avg. # Utterances Per Dialogue	5.25
Avg. # Tokens Per Utterance	9
Vocabulary Size	1,601

Table 6.2: Statistics of the in-car multi-turn, multi-domain, goal-oriented dialogue dataset.

6.3.2 Pre-processing and Model Hyper-parameters

We processed the dataset by creating inputs out of the driver’s utterances and the accompanied KG and the model outputs would be the current task intent and the assistants’ output. We also processed the entire Knowledge Graph to identify the entities and convert them into canonicalized forms. We also split entities with multiple objects. For instance, a weather forecast that looks like "frost, low of 20F, high of 30F" can be split into (weather-condition="frost", low-temperature="low of 20F", high-temperature="high of 30F").

For training the Joint Embedding we used a contextual window for the co-occurrence matrix equal to 15 which is $\frac{1}{4}^{th}$ of the mean of input size. We trained the model with $\alpha = 0.01$ and $\lambda = 10000$ and

the embedding vectors with dimensions of 300 for 500 epochs with learning rate equals to $1e - 4$ and a weight-decay of $1e - 6$. The stats regarding the Knowledge Graph is provided in Table 6.3.

As for the sequence-to-sequence model, we trained each of them on a GPU with 3072 CUDA cores and a VRAM of 12GB. The model is trained for 1000 epoch with a batch size of 128 and a hidden layer size equal to the embedding dimension of 300. We set the learning rate to be $1e - 4$ for the encoder and $5e - 4$ for the decoder, we also added a gradient clipping of 50.0 for countering the ‘exploding gradient’ problem; by doing so we prevent the gradients from growing exponentially and either overflow (undefined values), or overshoot steep cliffs in the cost function. We didn’t do any exhaustive hyper-parameter optimization, the values reported here are the ones used for all models. Our codes and preprocessing scripts are available in ².

Number of entities	291
Number of Triples	2,512
Number of Triples found in context	1,242
Max Triple co-occurrence	750
Unique words co-occurrences	1,30,803

Table 6.3: Statistics of the trained Joint Embedding.

6.4 Results

In this section, we report the results from our model on the mentioned dataset. We evaluate our system using BLEU scores as suggested by [207]. BLEU as defined by [207] analyzes the co-occurrences of n-grams in the reference and the proposed responses. It computes the n-gram precision for the whole dataset, which is then multiplied by a brevity penalty to penalize short translations We are reporting the geometric means of BLEU-1, BLEU-2, BLEU-3, and BLEU-4. In addition to that we evaluated the both our model and the Mem2Seq model proposed by [10] using an Embedding-based metrics such as Greedy Matching, Embedding Average and Vector Extrema as proposed by [208] ³. We are using an Attention based seq-to-seq model as a baseline along with the other state-of-the-art models on this corpora [11] (KV Retrieval Net) and [10] (Mem2Seq H3). We are reporting our best model which is attention based Sequence-to-sequence model (S2S+Intent+JE+EL) with Joint Embeddings (JE) as inputs and trained as a multi-task objective of learning response utterance along with user Intent further, regularized using Entity Loss (EL). This model is further improved with an additional step called Key-Value Entity Look-up (KVL) which is done during inference. The method was explained in 6.2.6.

As seen from the results, our proposed architecture has absolute improvements of 0.92 over KV Retrieval Net and 1.52 over Mem2Seq (H3) on BLEU. Although, the results are not directly comparable with the latter because we use canonicalized entity forms like [11]. We are reporting BLEU scores because for task-oriented dialogues there’s not much variance between the generated answers, unlike open-domain dialogues [208]. We are using the official version (i.e. moses multi-bleu.perl script) ⁴ for

²https://github.com/s6fikass/Chatbot_KVNN

³We report these metrics for our best model and only for Mem2Seq because their implementation is open-source.

⁴multi-bleu: <https://raw.githubusercontent.com/moses-smt/mosesdecoder/master/>

Model	BLEU	Emb. Avg.	Vec. Ext.	Greedy
Attention based Seq-to-Seq	8.4	-	-	-
KV Retrieval Net(no enc attention)	10.8	-	-	-
KV Retrieval Net	13.2	-	-	-
Mem2Seq H3	12.6	0.668	0.742	0.528
S2S+Intent+JE+EL	14.12	-	-	-
S2S+Intent+JE+EL+KVL	18.31	0.955	0.974	0.625

Table 6.4: Results compared to baseline and State-of-the-art models.

doing all the evaluations. Also, as seen in table 8.2, our proposed model performs significantly better than Mem2Seq on emedding-based metrics too.

Model Used	BLEU Score	BLEU with KVL
S2S+glove	10.42	14.63
S2S+JE	13.35	15.29
S2S+Intent+JE	12.65	17.89
S2S+Intent+glove	13.25	17.27
S2S+Intent+JE+EL	14.12	18.31

Table 6.5: Ablation Study.

6.4.1 Ablation Study

As mentioned in the results, our best model is based on Joint embeddings as inputs, followed by jointly learning user intent and text generation; with entity loss based on further regularization techniques. To analyze which specific part is influencing the performance, we did an ablation study dissecting the different components of the model one by one. The performances are reported in Table 8.6(b).

The first model is a sequence-to-sequence (S2S) model with Glove [206] embeddings as input. These embeddings are trained on the training set. It can be seen that with the domain-vocabulary knowledge the model already performs better than vanilla attention sequence-to-sequence model. The next (S2S+JE) is the same model with Joint text and Knowledge Graph embeddings (also trained on the dialogue corpus and provided Knowledge Graph). This model has an absolute improvement of 2.93 in BLEU scores. Interestingly, adding the intent along with Joint Embeddings drops the performance by 0.7 but the encounters a little improvement with glove vectors. The model sees a further boost in performances (relative improvement of 3.7) over vanilla models with GloVe. All the models encounter better performances with the proposed key-value lookup (KVL) technique during inferencing.

6.4.2 Qualitative Analysis

To qualitatively understand the performance improvements of the model with a different setting as in Table 8.6(b) we analyze the outputs from all of them for a given user query. For the user (driver)

```
scripts/generic/multi-bleu.perl
```

Driver	Where is the nearest gas station?
S2S	chevron is 5_pm away at <UNK> it is going to snow
S2S+GloVe	chevron is 3_pm at room_215
S2S+JE	chevron is 5_miles at <UNK> high_of_30_f
S2S+Intent+JE	chevron is at 783_arcadia_pl
S2S+Intent+JE+EL	chevron is 2_miles away at 783_arcadia_pl
S2S+Intent+JE+EL+KVL	chevron is 5_miles away at 783_arcadia_pl
Target	chevron is 5_miles away at 783_arcadia_pl

Table 6.6: Example of generated responses on the Navigation domain explaining the improvement process through the model stages.

query *Where is the nearest gas station ?*, predicted responses are given in Table 6.6. The Knowledge Graph snapshot for this particular dialogue scenario is in Table 6.7.

As observed, the sequence-to-sequence (S2S) model is not able to produce knowledge grounded responses since it has no information about the background Knowledge Graph. Using Joint Embeddings (S2S+JE), although produces more knowledge grounded responses (since chevron is an entity of type gas_station and is at a distance of 5_miles), it outputs entities like high_of_30_f which is not related to navigation intent but the weather. Incorporating intent into the model (S2S+Intent+JE) ensures that it is generating words conditionally dependent on the intent also. Further grammatical improvements in the model response are seen with entity loss (S2S+Intent+JE+EL) which is further made more knowledge grounded with the proposed KVL method as seen in the last response (S2S+Intent+JE+EL+KVL). To further understand the quality of the produced responses, we did a human evaluation of a subset of the predicted responses from the test set. We asked the annotators to judge the response whether it is human-like or not on a scale of 1-5. The average score given by the annotators is 4.51.

Subject	Relation	Object
chevron	distance	5_miles
chevron	traffic_info	moderate_traffic
chevron	poi_type	gas_station
chevron	address	783_arcadia_pl

Table 6.7: KG triples for the dialogue in Table 6.6, for navigation.

6.4.3 Error Analysis

To further understand the model performance, we did a further analysis of the responses from the test set which gave very low BLEU scores during evaluation. The target and the predicted response are in Table 6.8.

As observed, the model produces fewer word overlaps for generic responses like *have a good day*. But, the responses are grammatically and semantically correct, which cannot be measured using BLEU scores. The other types of errors are factual errors where the model fails because it requires reasoning as in case of the 4th example. The user is asking if the weather is cloudy in this case, in Fresno.

Predicted Sentence	Target Sentence
You're welcome	Have a good day
What city do you want the forecast for	What city can i give you this weather for
Setting gps for quickest route now	I picked the route for you drive carefully
It is cloudy with a low of 80f in fresno	There are no clouds in fresno right now

Table 6.8: Error Analysis.

Subject	Relation	Object
fresno	monday	clear_skies
fresno	monday	low_40f

Table 6.9: KG triples and context for the error analysis in Table 6.8 for weather.

Using a Knowledge Copy Mechanism into the response-generation process

7.1 Introduction

Moving in the same direction of generating dialogues using a knowledge graph, in this chapter and in the next chapter we define models to answer the fourth research question "How response generation in a Generative Dialogue system can be made Knowledge Graph aware?". In the following chapter we explore techniques to directly incorporate the Knowledge Graph information into the response generation process. The methods proposed are published as a paper [136] in the International semantic web conference. The author proposed the novel architecture proposed in this chapter, ran the experiments and also co-wrote the paper with peers.

With the recent advancements in neural network based techniques for language understanding and generation, there is an upheaved interest in having systems which are able to have articulate conversations with humans. Dialogue systems can generally be classified into goal and non-goal oriented systems, based on the nature of the conversation. The former category includes systems which are able to solve specific set of tasks for users within a particular domain, e.g. restaurant or flight booking. Non-goal oriented dialogue systems, on the other hand, are a first step towards chit-chat scenarios where humans engage in conversations with bots over non-trivial topics. Both types of dialogue systems can benefit from added additional world knowledge [196],[11],[209].

For the case of non-goal oriented dialogues, the systems should be able to handle factoid as well as non-factoid queries like chit-chats or opinions on different subjects/domains. Generally, such systems are realized by using an extrinsic dialogue managers using intent detection subsequently followed by response generation (for the predicted intent) [210], [211]. Furthermore, in case of factoid queries posed to such systems, it is very important that they generate well articulated responses which are knowledge grounded. The systems must be able to generate a grammatically correct as well as factually grounded responses to such queries, while preserving co-references across the dialogue contexts. For better understanding, let us consider an example dialogue and the involved Knowledge Graph snippet in Figure 7.1. The conversation consists of chit-chat as well as factoid queries. For the factoid question "do you know what is the home ground of Arsenal?", the system must be able to answer with the correct entity (Emirates Stadium) along with a grammatically correct sentence; as well as handle co-references("its" in the third user utterance meaning the stadium). Ideally, for an

end-to-end system for non-goal oriented dialogues, the system should be able to handle all these kind of queries using a single, end-to-end architecture.

There are existing conversation datasets supported by Knowledge Graphs for well-grounded response generation. [11] introduced an in-car dialogue dataset for multi-domain, task-oriented dialogues along with a Knowledge Graph which can be used to answer questions about the task the user wants to be assisted with. The dataset consists of dialogues from the following domains: calendar scheduling, weather information retrieval, and point-of interest navigation. For non-goal oriented dialogues, [212] proposed a dataset in the movie domain. The proposed dataset contains short dialogues for factoid question answering over movies or for recommendations. They also provide a Knowledge Graph consisting of triples as (s, r, o). Where s is the subject, r stands for relations and o being the object. An example of a triple from the dataset is: (Flags of Our Fathers, directed_by, Clint Eastwood). The movie dialogues can utilize this provided Knowledge Graph for recommendation and question answering purposes. However, this dataset only tackles the problem of factual response generation in dialogues, and not well articulated ones.

To cater to the problem of generating well articulated, knowledge grounded responses for non-goal oriented dialogue systems, we propose a new dataset in the domain of soccer. We also propose the KG-Copy network which is able to copy facts from the KGs in case of factoid questions while generating well-articulated sentences as well as implicitly handling chit-chats, opinions by generating responses like a traditional sequence-to-sequence model.







The contributions of the paper can be summarized as follows:

- A new dataset of 2,990 conversations for non-goal oriented dialogues in the domain of soccer, over various club and national teams.
- A soccer Knowledge Graph which consists of facts, as triples, curated from wikipedia.
- An end-to-end based, novel neural network architecture as a baseline approach on this dataset. The network is empirically evaluated against other state-of-the-art architectures for knowledge grounded dialogue systems. The evaluation is done based on both knowledge groundedness using entity-F1 score and also standard, automated metrics (BLEU) for evaluating dialogue systems.

7.2 Soccer Dialogues Dataset

7.2.1 Wizard-of-Oz Style Data Collection

The proposed dataset for conversations over soccer is collected using AMT (Amazon Mechanical Turk) [213]. The dialogues are collected in an wizard-of-oz style [214] setup. In such a setup, humans believe they are interacting with machines, while the interaction is completely done by humans. The turkers, acting as users, were instructed to initiate a conversation about the given team with any query or opinion or just have some small-talks. This initial utterance is again posted as another AMT task for replying, this time a different turker is acting as a system. Turkers assigned to the system role were asked to use Wikipedia to answer questions posed by the user. We encouraged the turkers to ask factual questions as well as posing opinions over the given teams, or have chit chat conversations. After a sequence of 7-8 utterances, the turkers were instructed to eventually end the conversation. A screenshot from the experimental setup is shown in Figure 7.2. We restricted the Knowledge Graph to a

	USER:	Hey what's up?	
	SYSTEM:	Nothing much, how are you? Do you want to talk soccer?	
	USER:	I am fine, thanks. Yeah, do you know what is the home ground of Arsenal ?	
	SYSTEM:	Arsenal's home ground is <i>Emirates Stadium</i> .	
	USER:	Oh ok, what's its capacity?	
	SYSTEM:	It has a capacity of <i>60,338</i> .	

Subject	Predicate	Object
Arsenal	Home venue	Emirates Stadium
Emirates Stadium	capacity	60,338
Arsenal	chairman	Chips Keswick
Arsenal	Head coach	Unai Emery

Arsenal Knowledge Graph snippet.

Figure 7.1: A conversation about the football club Arsenal and the Knowledge Graph involved.

limited set of teams. The teams are picked based on popularity, the national teams chosen are: Sweden, Spain, Senegal, Portugal, Nigeria, Mexico, Italy, Iceland, Germany, France, Croatia, Colombia, Brazil, Belgium, Argentina, Uruguay and Switzerland. The club teams provided for conversing are: F.C. Barcelona, Real Madrid, Juventus F.C., Manchester United, Paris Saint Germain F.C., Liverpool F.C., Chelsea F.C., Atletico Madrid, F.C. Bayern Munich, F.C. Porto and Borussia Dortmund. We also encouraged people to converse about soccer without any particular team. The number of conversations are equally distributed across all teams. The statistics of the total number of conversations are given in Table 7.1.

Dataset	# of Dialogues	# of Utterances
Train	2,493	12,243
Validation	149	737
Test	348	1,727

Table 7.1: Statistics of Soccer Dataset.

Curious Football: Conversations over Football

Instructions: (Please read carefully)

- **Goal:** Based on the team/teams mentioned in the **cue** have a **question-answer** based **conversation** about **football (soccer)**.
- **Suggestions:** Ask questions about teams' performance, trivia, its players, and upcoming fixtures.
- **Answering:** Feel free to use **Wikipedia** or any other source for answering a questions.
- **Context:** Maintain a **context** of the conversations. Please ask questions or answer according to the last question in the **conversation box**.
- **Starting:** If conversation history is empty, start with a salutation.
- **Length:** A conversation length of 7-9 messages is optimum. Try to end a conversation after that.
- **Other:** We're **passionate** about football, and you might be too. But please keep the conversations **polite**.

Conversation History:
Q1: Who is the best player in the world?
A1: I would say Messi.

Type your question/answer here.

Figure 7.2: AMT setup for getting conversations over soccer.

7.2.2 Ensuring Coherence

In order to ensure coherent dialogues between turkers, an additional task is created for each dialogue, where turkers were asked to annotate if the give dialogue is coherent or incoherent. Dialogues which are tagged incoherent by turkers are discarded.

7.2.3 Soccer Knowledge Graph

A KG in the context of this paper is a directed, multi-relational graph that represents entities as nodes, and their relations as edges, which can be used as an abstraction of the real world. KGs consists of triples of the form $(s,r,o) \in KG$, where s and o denote the subject and object entities, respectively, and r denotes their relation.

Following [215], we created a soccer Knowledge Graph from WikiData [111] which consists of information such as a team's coach, captain and also information such as home ground and its capacity for soccer clubs. For information about players, we have parsed individual wikipedia pages of the teams and mined goals scored, position, height and age of players. This ensures that the info in the KG is up to date. Finally, we curated the Knowledge Graphs for each team manually and added information such as jersey color. The KG schema is provided in 7.3 and additional statistics about KG and conversation is provided in table 7.2.

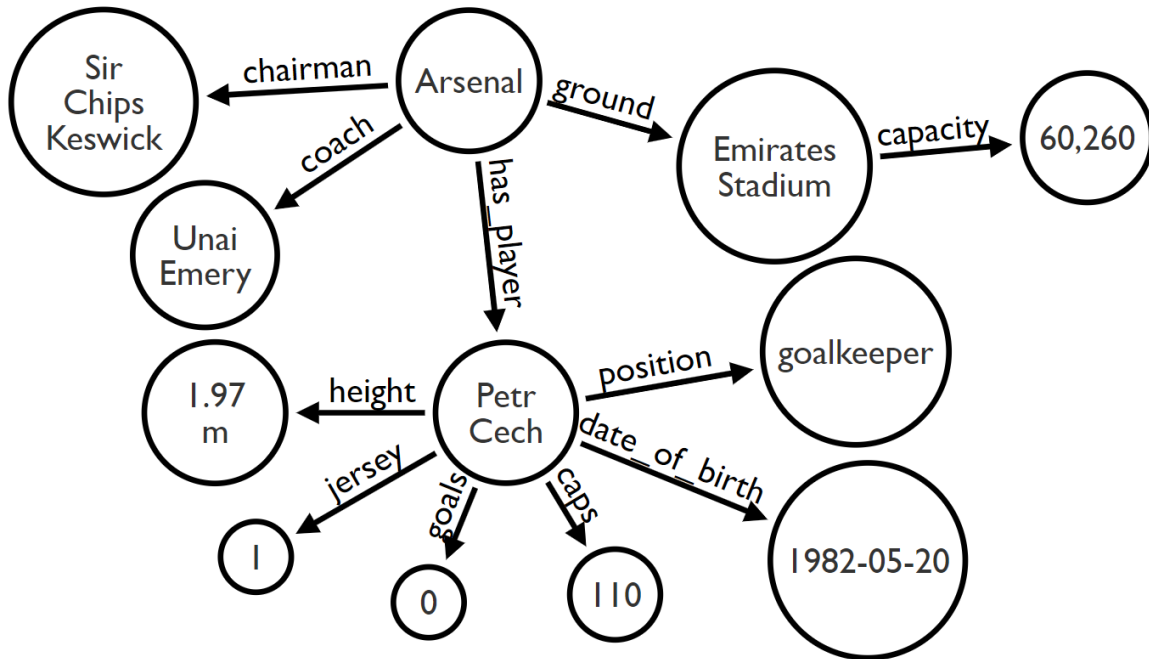


Figure 7.3: Schema of the proposed Knowledge Graph for Arsenal.

Statistics	Count
Total Vocabulary Words (v)	4782
Avg. Number of Conversations/team	83
Avg. Number of Triples/team	148
Avg. Number of Entities/ team	108
Avg. Number of Relations/team	13

Table 7.2: KG statistics.

7.3 KG-Copy Model

The problem we are tackling in this chapter is: given a Knowledge Graph (KG), and an input context in a dialogue, the model should be able to generate factual as well as well articulated response. During the dialogue generation process, at every time-step t , the model could either use the KG or generate a word from the vocabulary. We propose the KG Copy model which tackles this particular problem of producing well-grounded response generation.

KG-Copy is essentially a sequence-to-sequence encoder-decoder based neural network model, where the decoder can generate words either from the vocabulary or from the Knowledge Graph. The model is mainly influenced by the copynets approach [216]. However, unlike copynets, KG-Copy copies tokens from the local Knowledge Graph using a special gating mechanism. Here, local KG depicts the KG for the team the dialogue is about. We introduce the KG-Copy's encoder, decoder and the gating mechanism below.

7.3.1 KG-Copy Encoder

The encoder is based on a recurrent neural network (RNN), more specifically a long-short term memory network (LSTM). It encodes the given input word sequence $X = [x_1, x_2, \dots, x_T]$ to a fixed length vector c . The hidden states are defined by

$$h_t = f_{enc}(x_t, h_{t-1}) \quad (7.1)$$

where f_{enc} is the encoder recurrent function and the context vector c is given by

$$c = \phi(h_1, h_2, \dots, h_T) \quad (7.2)$$

Here in, ϕ is the summarization function given the hidden states h_t . It can be computed by taking the last hidden state h_T or applying attention over the hidden states [47, 217] and getting a weighted value of the hidden states (attention).

7.3.2 KG-Copy Decoder

The decoder is an attention based RNN (LSTM) model. The input to the decoder is the context c from the encoder along with h_T . At time-step t , the hidden-state of the decoder is given by

$$h_t^d = f_{dec}(x_t, h_{t-1}^d) \quad (7.3)$$

Where f_{dec} is the recurrent function of the decoder. The decoder hidden-states are initialized using h_T and the first token is < sos >. The attention mechanism [217]. The attention weights are calculated by concatenating the hidden states h_t^d along with h_t .

$$\alpha_t = softmax(W_s(tanh(W_c[h_t; h_t^d]))) \quad (7.4)$$

Here in, W_c and W_s are the weights of the attention model. The final weighted context representation is given by

$$\tilde{h}_t = \sum_t \alpha_t h_t \quad (7.5)$$

This representation is concatenated (represented by ;) with the hidden states of the decoder to generate an output from the vocabulary with size v .

The output is then given by

$$o_t = W_o([h_t; \tilde{h}_t^d]) \quad (7.6)$$

In the above equation, W_o are the output weights with dimension $\mathbf{R}^{h_{dim} \times v}$. h_{dim} is the dimension of the hidden layer of the decoder RNN.

7.3.3 Sentient Gating

The sentient gating, as mentioned previously, is inspired mainly by [216, 218]. This gate acts as a sentinel mechanism which decides whether to copy from the local KG or to generate a word from training vocabulary (v). The final objective function can be written as the probability of predicting the

next word during decoding based on the encoder hidden-states and the Knowledge Graph (KG).

$$p(y_t|h_t..h_1, KG) \quad (7.7)$$

The proposed gating is an embedding based model. At every time-step t , the input query and the input to the decoder are fed into the sentient gate. Firstly, a simple averaging of the input query embedding is done generating emb_q , which can be treated as a vector representation of the input context.

$$emb_q = \frac{1}{N} \sum (emb_{w_1} \dots emb_{w_t}) \quad (7.8)$$

emb_{w_t} is the embedding of the t^{th} word in the context. N.B. we only consider noun and verb phrases in the context to calculate emb_q . For the KG representation, an embedding average of the local KG's subject entity and relation labels for each triple is performed yielding a KG embedding emb_{kg} . We consider a total of k triples in the local KG.

Finally, the query embedding is matched with these KG embeddings using a similarity function (cosine similarity in this case).

$$kg_{sim} = \tanh(\cos(emb_q, emb_{kg}^1), \cos(emb_q, emb_{kg}^2) \dots \cos(emb_q, emb_{kg}^k)) \quad (7.9)$$

The input to the decoder at t is fed into the embedding too as mentioned previously yielding emb_d . The final sentient value at t is given by :

$$s_t = \text{sigmoid}(W_{sent} [emb_q + emb_d; kg_{sim}; s_{t-1}]) \quad (7.10)$$

W_{sent} is another trainable parameter of the model and ";" is the concatenation operation. The final prediction is given by:

$$out_t = s_t * kg_{sim} + (1 - s_t) * o_t \quad (7.11)$$

The model is visualized in Figure 7.4.

7.4 Training and Model Hyper-parameters

7.4.1 Training Objective

The model is trained based on a multi-task objective, where the final objective is to optimize the cross-entropy based vocabulary loss (L_{vocab}) and also the binary cross-entropy loss ($L_{sentient}$) for the sentient gate (s_g). This value is 1 if the generated token at that step comes from the KG, otherwise 0. For example, in the example provided in 7.1, for the 2nd system utterance, this value would be 1 for $t = 5$ (Emirates Stadium), but 0 for the previous time-steps.

The total loss is given by:

$$L_{tot} = L_{vocab} + L_{sentient} \quad (7.12)$$

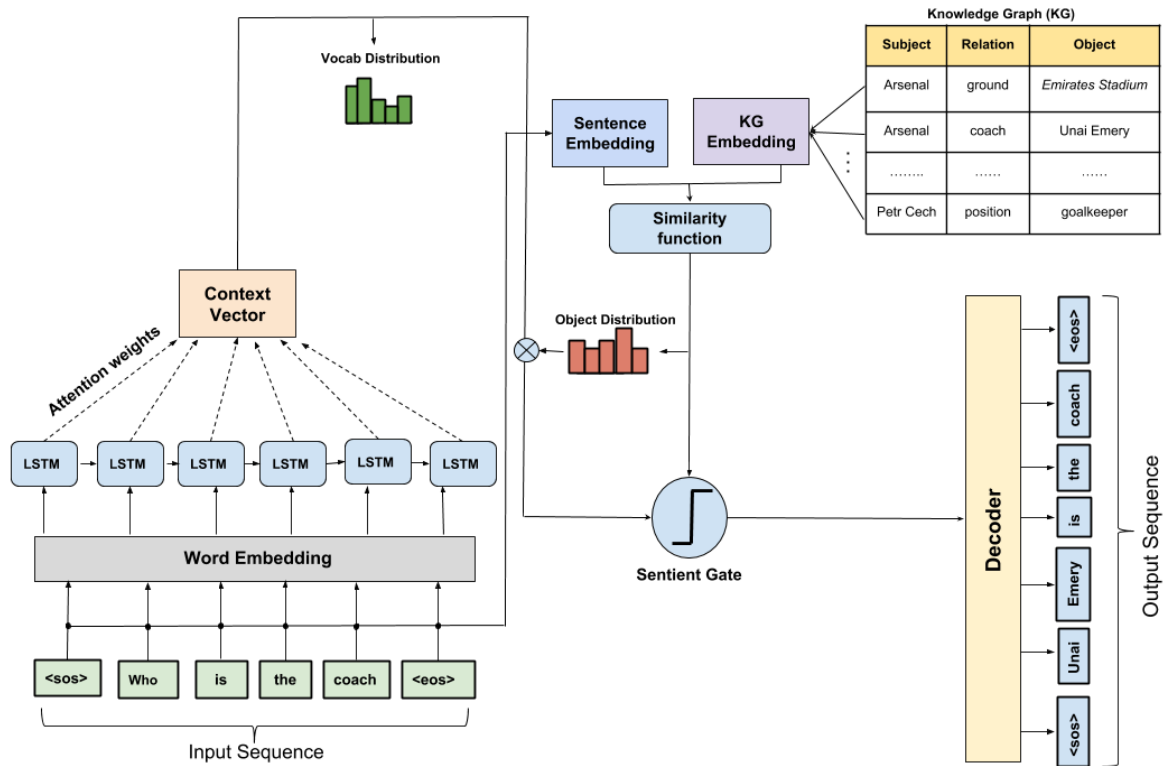


Figure 7.4: KG-Copy Model Encoder-Decoder Architecture for Knowledge Grounded Response Generation.

7.4.2 Training Details

To train the model, we perform a string similarity over KG for each of the questions in training data set to find which questions are answerable from the KG. Then we replace those answers with the position number of the triples where the answer (object) belongs in the KG, during pre-processing. This is followed by a manual step where we verify whether the input query is simple, factoid question or not and also the correctness of answer (object). The vocabulary is built only using the training data. No additional pre-processing is done for the validation and test sets except changing words to their corresponding indices in the vocabulary.

For training, a batch-size of 32 is used and the model is trained for 100 epochs. We save the model with the best validation f1-score and evaluate it on the test set. We apply Adam [159] for optimization with a learning rate of 1e-3 for the encoder and 5e-3 for the decoder. The size of the hidden layer of both the encoder and decoder LSTM is set to 64. We train the decoder RNN with teacher-forcing [219]. The input word embedding layer is of dimension 300 and initialized with pretrained fasttext [201] word embeddings. A dropout [220] of 0.3 is used for the encoder and decoder RNNs and 0.4 for the input embedding. The training process is conducted on a GPU with 3072 CUDA cores and a VRAM of 12GB. The soccer dataset (conversation and KG) and the KG-Copy model's code are open-sourced¹ for ensuring reproducibility.

¹https://github.com/SmartDataAnalytics/KG-Copy_Network

7.5 Evaluation

We compare our proposed model with Mem2Seq and a vanilla encoder-decoder with attention. We report the BLEU scores [207] and also the entity-F1 scores on both the proposed soccer dataset and the In-car dialogue dataset. The results show that our proposed model performs better than both the vanilla attention sequence-to-sequence models and Mem2Seq model across both metrics. Our model outperforms Mem2Seq by 1.51 in BLEU score and 15 % on entity-F1 score. It performs better than the vanilla sequence-to-sequence model by 1.21 on the BLEU metric on the soccer dataset. Interestingly, Mem2Seq performs better than the vanilla model on validation, but it fails to generalize on test set. The proposed model although has lower BLEU on the in-car dialogue dataset, but has a better entity f1 scores (by 19.4 %), implying stronger reasoning capabilities over entities and relations [10].

Model	BLEU	Entity-F1
	Valid Test	Valid Test
Vanilla Encoder-decoder with Attention	1.04 0.82	_ _
Mem2Seq [10]	1.30 0.52	6.78 7.03
KG Copy (proposed model)	2.56 2.05	24.98 23.58

Table 7.3: Results on Soccer Dataset.

Model	BLEU	Entity-F1
Vanilla Encoder-decoder with Attention	8.4	10.3
Mem2Seq [10]	12.6	33.4
KG Copy (proposed model)	9.6	52.8

Table 7.4: Results on the In-car Dialogue Dataset.

7.6 Discussion

7.6.1 Qualitative Analysis

In this section, we will qualitatively analyze the response generation of our model along with the background knowledge integration (grounding) and compare it with both Mem2Seq and vanilla sequence-to-sequence models.

Some example response from test are given in Table 7.6. As seen, the KG-copy model is able to have more articulate responses compare to sequence-to-sequence and Mem2Seq models. The model is also able to form well articulate opinions compared to other models (2nd column)².

²Seq2Seq model has generated a more articulated response based on the given context but it is factually wrong: Senegal is nicknamed the Lions of Teranga and not Nigeria.

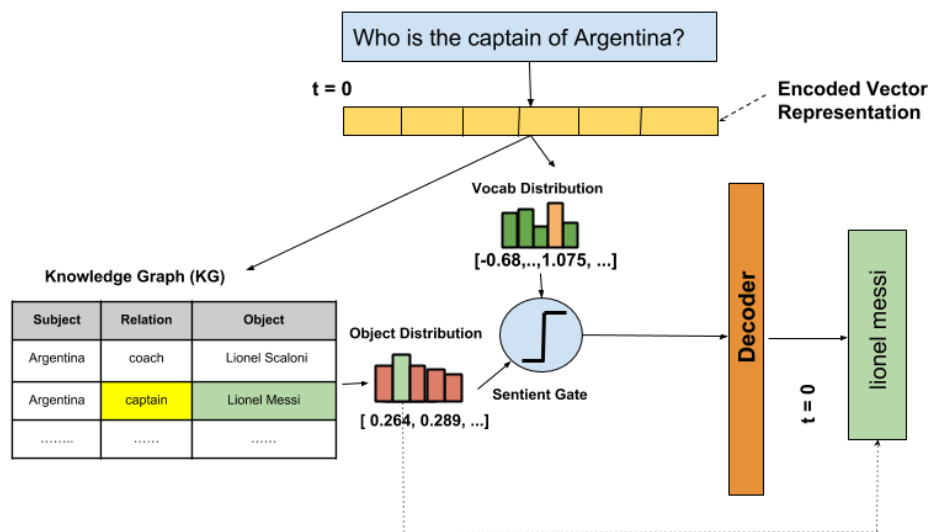


Figure 7.5: Response Generation during Decoding for KG-Copy Model.

Some more examples along with the response from our model are given in Table 7.7. As observed, all those responses are well grounded. The first response is factually correct and also a well-articulate one; interestingly, even the true human response on the other hand is not. The last response is knowledge ground but not well articulate. The model is also able to perform co-reference resolution implicitly while generating responses. To verify, let us consider another conversation between an user with the deployed KG-copy model.

User utterance: i like the team pretty much

Response: i don't think they're a lot of winning.

User utterance: who is the captain of argentina ?

Response: lionel messi is the captain

User utterance: do you know the name of their coach ?

Response: lionel scaloni is the coach

In the last response, the model is able to identify that the pronoun "their" refers to the team and is able to maintain a knowledge grounded, as well as articulate responses even for relatively long dialogue turns. For time-step $t = 0$, the visualization of the sentient gating mechanism is provided in Figure 7.5. The vocabulary distribution is over v , and the object distribution here is over the local KG for the team.

Furthermore, following [11], we did an internal survey regarding the responses generated by KG-Copy network, judging the quality of responses based on the context on a scale of 1-5 on correctness and human-like sentence formation. The former measures how correct the generated response is with respect to the true response from the turker, and the latter how grammatically correct the produced response is. We randomly pick 50 conversation utterances from the test set and report this human evaluation both on Mem2Seq and KG-copy in Table 7.5.

Model	Correctness	Human-like
Mem2Seq	1.30	2.44
KG-Copy	2.26	3.88

Table 7.5: Human evaluation of generated responses.

Context Type	<i>factoid</i>	<i>opinions</i>
Input contexts	what is the name of the captain of mexico ?	I like this team.
True response	andres guardado (captain)	nigeria is a very well performing team and i like them a lot as well
Seq2Seq	The is is	They are nicknamed the Lions of Teranga.
Mem2Seq	Mexico is the	They are a
KG-Copy	andres guardado.	they are a good team.

Table 7.6: KG-copy Response for Factoid and non-Factoid queries.

input contexts	Turker Response	KG copy response
who is the captain of iceland?	aron gunnarsson	aron gunnarsson is the captain.
who is the captain of italy ?	chiellini	giorgio chiellini is the captain.
who is the coach for italy ?	i think roberto mancini	roberto mancini is the coach
who is the coach of bayern munich ?	niko kovac is the current manager	niko kovac

Table 7.7: KG Copy Model’s Knowledge Grounded Responses.

7.6.2 Error Analysis

Although the model is able to generate some well articulated, knowledge grounded responses for factual queries as seen in Tables 7.6 and 7.7, the model often fails in producing factually correct responses as also evident from Table 7.5. More of those cases are analyzed below.

The model produces too generic and non-factual responses to queries about opinions about favorite players as shown in Table 7.8. This is mostly because the vocabulary size is relatively large compare to the size of training dialogues. This can be improved with more training data, especially with more knowledge grounded conversations. For the first response, the model is not able to interpret the question and generates a generic response. For the second case, the generated response is not factual because the question is about Argentina, but Eden Hazard is from a different team (Belgium).

The KG-copy model also often suffers when more complex quantitative and other reasoning skills are required to respond to the context. For example, for the first context in Table 7.9, the model needs to perform a count operation over the KG to answer it, which is currently unsupported. Similarly,

input contexts	True Response	KG copy response
who is senegal 's best current player not including mane ?	keita balde diao	i think it is the best player in the world cup
who 's your favorite player ?	messi	i think eden hazard is the best player

Table 7.8: Incorrect opinionated responses from KG-Copy model.

for the second case the model would require better language inferencing to respond. The model also suffers from the problem of unknown words in the test set.

input contexts	how many world cups has the brazil team won ?	who was the top scorer in the world cup for belgium ?
True Response	brazil has won the fifa world cup five times	eden hazard
Predicted	they won the world cup	i think it was the top scorer for the world cup

Table 7.9: Incorrect factual responses from KG Copy model.

Grounding Dialogue Systems via Knowledge Graph Aware Decoding with Pre-trained Transformers

8.1 Introduction

While continuing to answer the fourth research question from the previous chapter, in this chapter we propose a better knowledge-grounded model which directly answers using the relation information from the Knowledge Graph rather than copying the object information like in the previous KG-copy model. The work is submitted as a research paper in the extended semantic web conference [221]. The author has devised the main algorithm, partly ran experiments and also co-wrote the paper.

From the last chapter, we have seen that generative dialogue systems can benefit from the integration of additional world knowledge [20]. In particular, Knowledge Graphs, which are an abstraction of real world knowledge, have been shown to be useful for this purpose. We will focus on more ways of integrating the KG into the dialogue generation process in this chapter.

To model the response using the KG, all current methods [10, 20][7] assume that the entity in the input query or a sub-graph of the whole KG, which can be used to generate the answer, is already known [10, 135]. This assumption makes it difficult to scale such systems to real-world scenarios, because the task of extracting sub-graphs or, alternatively, performing entity linking in large Knowledge Graphs is non-trivial [222]. An example of a Knowledge Graph based dialogue system is shown in Figure 8.1. In order to generate the response *James Cameron is the director*, the system has to link the entity mentioned in the question in the first turn i.e. *Titanic*, and identify the relation in the KG connecting the entities *Titanic* with *James Cameron*, namely *directed by*. Additionally, to obtain a natural dialogue system, it should also reply with coherent responses (eg. "James Cameron is the director") and should be able to handle small-talk such as greetings, humour and so on. Furthermore, in order to perform multi-turn dialogues, the system should also be able to perform co-reference resolution and connect the pronoun (*he*) in the second question with *James Cameron*.

In order to tackle these research challenges, we model the dialogue generation process by jointly learning the entity and relation information during the dialogue generation process using a pre-trained BERT model in an end-to-end manner. The model's response generation is designed to learn to predict relation(s) from the input KG instead of the actual object(s) (intermediate representation).

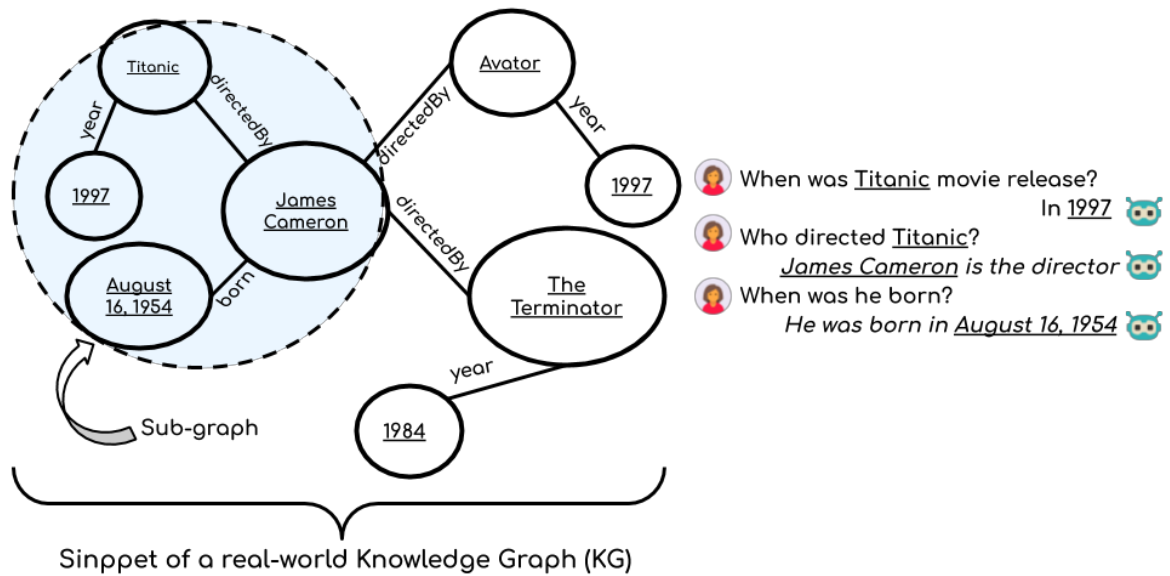


Figure 8.1: Example of a knowledge grounded conversation.

Additionally, a graph Laplacian based method is used to encode the input sub-graph and use it for the final decoding process.

Experimental results suggest that the proposed method improves upon previous state-of-the-art approaches for both goal and non-goal oriented dialogues. Overall, the contributions of this paper are as follows:

- A novel approach, leveraging the Knowledge Graph elements (entities and relations) in the questions along with pre-trained transformers, which helps in generating suitable knowledge grounded responses.
- We have also additionally encoded the sub-graph structure of the entity of the input query with a Graph Laplacian, which is traditionally used in graph neural networks. This novel decoding method further improves performance.
- An extensive evaluation and ablation study of the proposed model on two datasets requiring grounded KG knowledge: an in-car dialogue dataset and soccer dialogues for goal and non-goal oriented setting, respectively. Evaluation results show that the proposed model produces improved knowledge grounded responses compared to other state-of-the-art dialogue systems w.r.t. automated metrics, and human-evaluation for both goal and non-goal oriented dialogues.

8.2 Model Description

We aim to solve the problem of answer generation in a dialogue using a KG as defined below.

In this work, the vertices V of the (Knowledge Graph) KG represent entities $e \in V$, while the edges represent the relationships between those entities. A fact is an ordered triple consisting of an entity e (or subject s), an object o and the relation r (a.k.a. predicate p) between them, denoted by (s, p, o) .

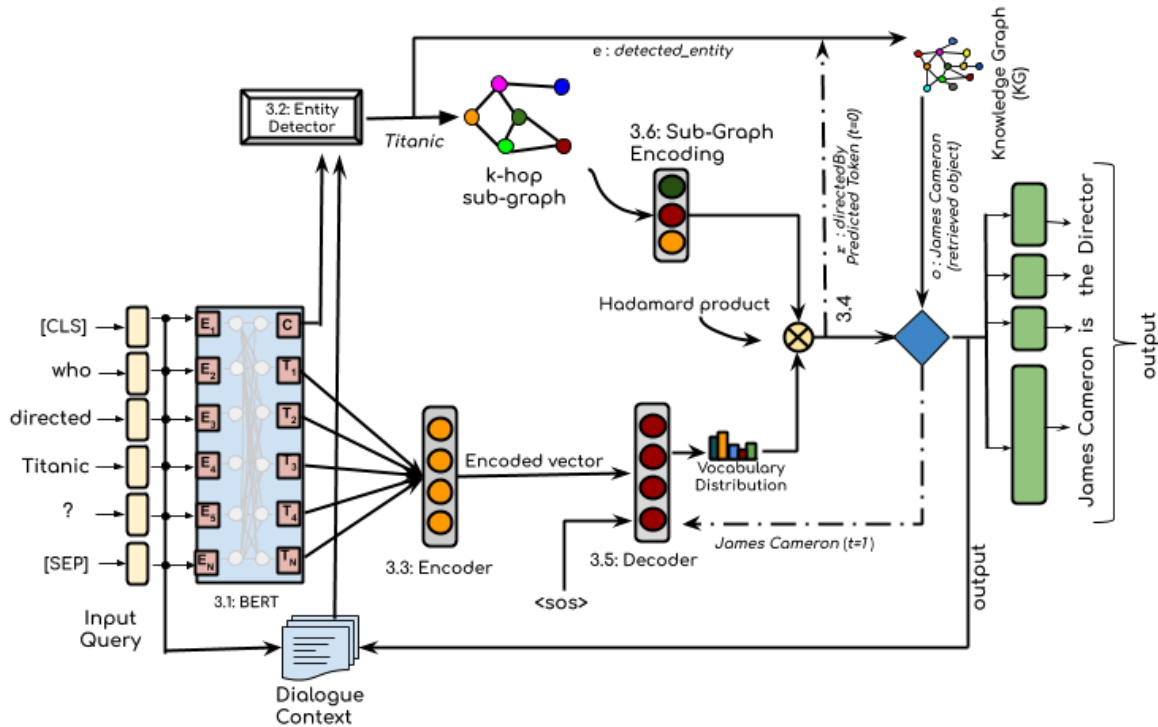


Figure 8.2: KGIRNet model diagram.

The proposed model for dialogue generation, which we call KGIRNet, is quintessentially a sequence-to-sequence based model with a pre-trained transformer serving as its input as illustrated in Figure 8.2. In contrast to previous works, we introduce an intermediate query representation using the relation information, for training. We also employ a Graph Laplacian based method for encoding the input sub-graph of the KG that aids in predicting the correct relation(s) as well as filter out irrelevant KG elements. Our approach consists of the following steps for which more details are provided below: Firstly, we encode the input query q with a pre-trained BERT model (Section 8.2.1). Next we detect a core entity e occurring in q (Section 8.2.2). Input query q and generated output is appended to the dialogue context in every utterance. The input query, encoded using the BERT model is then passed through an LSTM encoder to get an encoded representation (Section 8.2.3). This encoded representation is passed onto another LSTM decoder (Section 8.2.5), which outputs a probability distribution for the output tokens at every time-step. Additionally, the model also extracts the k -hop neighbourhood of e in the KG and encodes it using graph based encoding (Section 8.2.6) and perform a Hadamard product with token probability distribution from the decoder. The decoding process stops when it encounters a special token, $\langle EOS \rangle$ (end of sentence). Dotted lines in the model diagram represent operations performed at a different time-step t in the decoding process.

8.2.1 Query Encoding

BERT is a pre-trained multi-layer, bi-directional transformer [48] model proposed by [49]. It is trained on unlabelled data for two-phased objective: masked language model and next sentence prediction. For encoding any text, special tokens [CLS] and [SEP] are inserted at the beginning and the end of the text, respectively. In the case of KGIRNet, the input query $q = (q_1, q_2, \dots, q_n)$ at turn t_d in the dialogue, along with the context upto turn $t_d - 1$ is first encoded using this pre-trained BERT model which produces hidden states (T_1, T_2, \dots, T_n) for each token and an aggregated hidden state representation C for the [CLS] (first) token. We encode the whole query q along with the context, concatenated with a special token, $\langle EOU \rangle$ (end of utterance).

8.2.2 Entity Detection

The aggregated hidden representation from the BERT model C is passed to a fully connected hidden layer to predict the entity $e_{inp} \in V$ in the input question as given by

$$e_{inp} = \text{softmax}(w_{ent}C + b_{ent}) \quad (8.1)$$

Where, w_{ent} and b_{ent} are the parameters of the fully connected hidden layer.

8.2.3 Input Query Encoder

The hidden state representations (T_1, T_2, \dots, T_n) of the input query q (and dialogue context) using BERT is further encoded using an LSTM [41] encoder which produces a final hidden state at the n -th time-step given by

$$h_n^e = f_{enc}(T_n, h_{n-1}^e) \quad (8.2)$$

f_{enc} is a recurrent function and T_n is the hidden state for the input token q_n from BERT. The final representation of the encoder response is a representation at every n denoted by

$$H_e = (h_0^e, h_1^e, \dots, h_N^e) \quad (8.3)$$

8.2.4 Intermediate Representation

As an intermediate response, we let the model learn the relation or edge label(s) required to answer the question, instead of the actual object label(s). In order to do this, we additionally incorporated the relation labels obtained by applying the label function f_l to all edges in the KG into the output vocabulary set. If the output vocabulary size for a vanilla sequence-to-sequence model is v_o , the total output vocabulary size becomes v_{od} which is the sum of v_o and v_{kg} . The latter being the labels from applying the f_l to all edges (or relations) in the KG.

For example, if in a certain response, a token corresponds to an object label o_l (obtained by applying f_l to o) in the fact (e, r, o) , the token is replaced with a KG token v_{kg} corresponding to the edge or relation label r_l of $r \in E$ in the KG. During training, the decoder would see the string obtained by applying f_l to the edge between the entities *Titanic* and *James Cameroon*, denoted here as $r:directedBy$. Hence, it will try to learn the relation instead of the actual object. This makes the system more generic and KG aware, and easily scalable to new facts and domains.

During evaluation, when the decoder generates a token from v_{kg} , a KG lookup is done to decode the label o_t of the node $o \in V$ in the KG (V being the set of nodes or vertices in the KG). This is generally done using a SPARQL query.

8.2.5 Decoding Process

The decoding process generates an output token at every time-step t in the response generation process. It gets as input the encoded response H_e and also the KG distribution from the graph encoding process as explained later. The decoder is also a LSTM, which is initialized with the encoder last hidden states and the first token used as input to it is a special token, $\langle SOS \rangle$ (start of sentence). The decoder hidden states are similar to that of the encoder as given by the recurrent function f_{dec}

$$h_n^d = f_{dec}(w_{dec}, h_{n-1}^d) \quad (8.4)$$

This hidden state is used to compute an attention over all the hidden states of the encoder following [217], as given by

$$\alpha_t = softmax(W_s(tanh(W_c[H_e; h_t^d]))) \quad (8.5)$$

Where, W_c and W_s are the weights of the attention model. The final weighted context representation is given by

$$\tilde{h}_t = \sum_t \alpha_t h_t \quad (8.6)$$

This representation is concatenated (represented by $:$) with the hidden states of the decoder to generate an output from the vocabulary with size v_{od} .

The output vocab distribution from the decoder is given by

$$O_{dec} = W_o([h_t; \tilde{h}_t^d]) \quad (8.7)$$

In the above equation, W_o are the output weights with dimension $\mathbf{R}^{h_{dim} \times v_{od}}$. h_{dim} being the dimension of the hidden layer of the decoder LSTM. The total loss is the sum of the vocabulary loss and the entity detection loss. Finally, we use beam-search [223] during the the decoding method.

8.2.6 Sub-Graph Encoding

In order to limit the KGIRNet model to predict only from those relations which are connected to the input entity predicted from step 8.2.1, we encode the sub-graph along with its labels and use it in the final decoding process while evaluating.

The k -hop sub-graph of the input entity is encoded using Graph Laplacian [142] given by

$$G_{enc} = D^{-1} \tilde{A} f_{in} \quad (8.8)$$

Where, $\tilde{A} = A + I$. A being the adjacency matrix, I is the identity matrix and D is the degree matrix. f_{in} is a feature representation of the vertices and edges in the input graph. G_{enc} is a vector with dimensions \mathbf{R}^{ik} corresponding to the total number of nodes and edges in the k -hop sub-graph of e . An example of the sub-graph encoding mechanism is shown in 8.3.

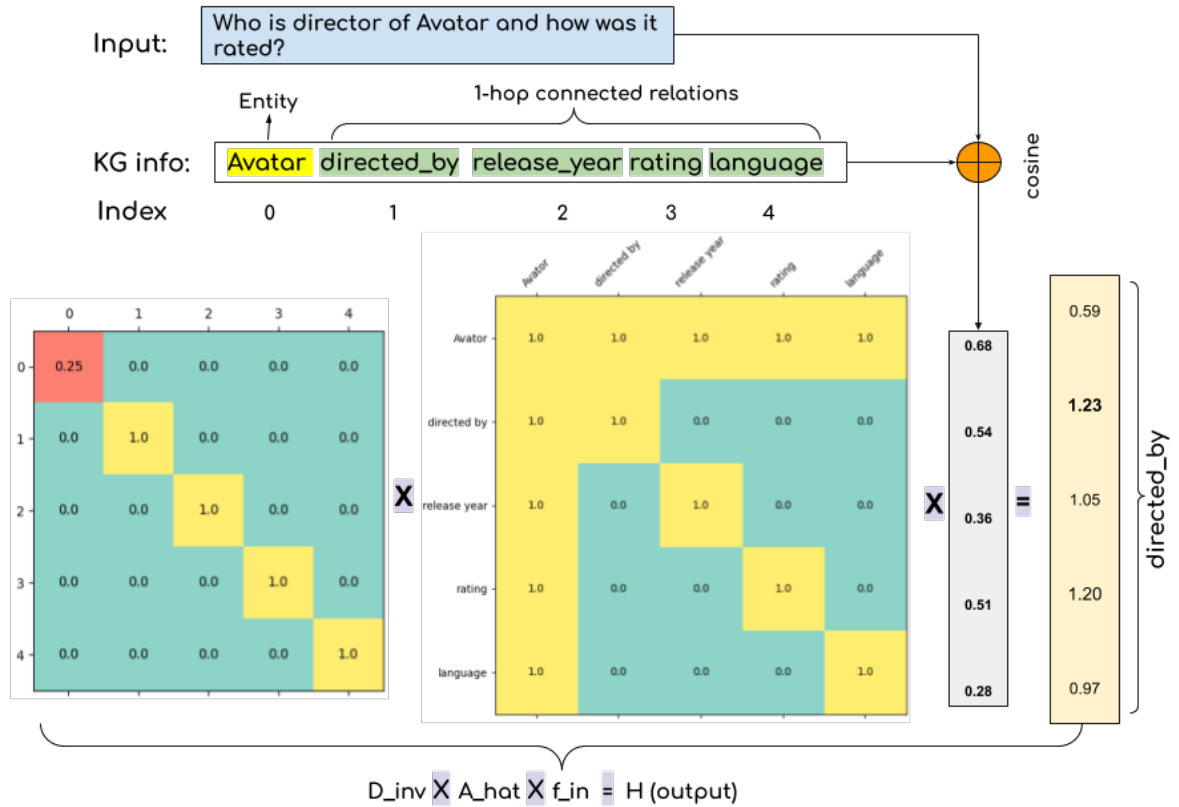


Figure 8.3: Sub-Graph Encoding using Graph Laplacian.

The final vocabulary distribution $O_f \in R^{v_{od}}$ is a Hadamard product of this graph vector and the vocabulary distribution output from the decoder.

$$O_f = O_{dec} \odot G_{enc} \quad (8.9)$$

This step essentially helps the model to give additional importance to the relations connected at k-hop based on its similarity with the query also to filter(mask) out relations from the response which are not connected to it. For the query in 8.3 *who is the director of Avatar and how was it rated ?* The graph laplacian based encoding method using only relation labels already gives higher scores for the relations *directed_by* and *rating*, which are required by the question. This vector when multiplied with the final output vocabulary helps in better relation learning.

8.3 Experimental Setup

8.3.1 Datasets

Available datasets for knowledge grounded conversations are the *in-car dialogue* data as proposed by [20] and *soccer dialogues* over football club and national teams using a Knowledge Graph [136]. The former contains dialogues for assisting the driver of a car with data related to weather, schedules and navigation, in a goal-oriented setting. The soccer dataset contains non-goal oriented dialogues

Table 8.1: Dataset statistics.

	In-car Dialogues	Soccer Dialogues
Number of triples	8561	4301
Number of entity	271	932
Number of relations	36	30
Number of dialogues in train data	2011	1328
Number of utterances in train data	5528	6523
KG-grounded questions in train data (%)	44.95%	6.53%
Number of dialogues in validation data	242	149
Number of utterances in validation data	657	737
KG-grounded questions in validation data (%)	33.94%	4.61%
Number of dialogues in test data	256	348
Number of utterances in test data	709	1727
KG-grounded questions in test data (%)	34.84%	3.88%

about questions on different soccer teams along with a Knowledge Graph consisting of facts extracted from Wikipedia about the respective clubs and national soccer teams. Both the datasets were collected using Amazon Mechanical Turk (AMT). The statistics of the datasets are provided in Table 8.1. As observed, the number of dialogues for both the dataset is low.

To perform a KG grounded dialogue as in KG based question-answering [224], it is important to annotate the dialogues with KG information such as the entities and relations, the dialogues are about. Such information were missing in the soccer dataset, hence we have semi-automatically annotated them with the input entity e_{inp} in the query q and the relations in the k -hop sub-graph of the input entity required to answer q . For the domain of in-car it was possible to automatically extract the entity and relation information from the dialogues and input local KG snippets. All datasets and code used for this paper are publicly available ¹.

8.3.2 Evaluation Metrics

We evaluate the models using the standard evaluation metrics BLEU [207] and entity F1 scores as used in the discussed state-of-the-art models. However, unlike [10] we use entity F1 scores based on the nodes V of the KG, as inspired by previous works on KG based question answering [106]. Additionally, we use METEOR [225] because it correlates the most with human judgement scores [226].

8.3.3 Model Settings

We use BERT-base-uncased model for encoding the input query. The encoder and decoder is modelled using an LSTM (long short term memory) with a hidden size of 256 and the KGIRNet model is trained with a batch size of 20. The learning rate of the used encoder is $1e-4$. For the decoder we select a learning rate of $1e-3$. We use the Adam optimizer to optimize the weights of the neural networks.

¹<https://github.com/DeepInEvil/kgirnet>

Table 8.2: Evaluation on goal and non-goal oriented dialogues.

Models	In-Car Dialogues			Soccer Dialogues		
	BLEU	Entity F1	METEOR	BLEU	Entity F1	METEOR
Seq2Seq	4.96	10.70	21.20	1.37	0.0	7.8
Mem2Seq[10]	9.43	24.60	28.80	0.82	04.95	7.8
GLMP[135]	9.15	21.28	29.10	0.43	22.40	7.7
Transformer[48]	8.27	24.79	29.06	0.45	0.0	6.7
DialoGPT[139]	7.35	21.36	20.50	0.76	0.0	5.5
KG-Copy[7]	-	-	-	1.93	03.17	10.89
KGIRNet	11.76	32.67	30.02	1.51	34.33	8.24

For all of our experiments we use a sub-graph size of $k=2$. For calculating f_{in} as in Equation 8.8, we use averaged word embedding similarity values between the query and the labels of the sub-graph elements. The similarity function used in this case is cosine similarity. We save the model with best validation entity f1 score. The code of the model is added as a supplementary material.

Table 8.3: Relation Linking Accuracy on SQB [184] Dataset.

Method	Model	Accuracy
Supervised	Bi-LSTM [6]	38.5
	HR-LSTM [8]	64.3
Unsupervised	Embedding Similarity	60.1
	Graph Laplacian (this work)	69.7

8.4 Results

In this section, we summarize the results from our experiments. We evaluate our proposed model KGIRNet against the current state-of-the-art systems for KG based dialogues; namely Mem2Seq [10], GLMP [135], and KG-Copy [136]². To include a baseline method, the results from a vanilla Seq2Seq model using an LSTM encoder-decoder are also reported in Table 8.2, along with a vanilla transformer [48] and a pre-trained GPT-2 model, DialoGPT [139] fine-tuned individually on both the datasets.

We observe that KGIRNet outperforms other approaches for both goal (in-car) and non-goal oriented (soccer) dialogues except for BLEU and METEOR scores on the soccer dataset. The effect of knowledge groundedness is particularly visible in the case of soccer dialogues, where most models (except GLMP) produces feeble knowledge grounded responses.

²KG-Copy reports on a subset of the in-car testset hence it is not reported here

In addition to evaluating dialogues, we also evaluate the proposed graph laplacian based relation learning module for the task of knowledge-graph based relation linking. Although, it is a well-researched topic and some systems claim to have solved the problem [110], but such systems are not able to handle relations which are not present in the training data [184]. The later also proposed a new, balanced dataset (SQB) for simple question answering which has same proportions of seen and unseen relations in the test or evaluation set. We have evaluated our unsupervised graph laplacian based method for relation linking on the SQB dataset against supervised methods namely Bi-LSTM [6], hr-bilstm [8] and unsupervised method such as text embedding based similarity between the query and the relations connected to the subject entity with 1-hop. The results are reported in 8.3. As observed, Graph Laplacian performs better wrt. supervised methods on unseen relations and also better than shallow embedding based similarity. This is one of the motivation for using this simple method during KGIRNet’s decoding process.

8.5 Discussion

Table 8.4: Analyzing sample predictions.

Input Query	True Response	Predicted Responses		
		GLMP	Mem2Seq	KGIRNet
who is currently coaching bvb dortmund	lucien favre	<i>the is the coach</i>	<i>yes , they have a good</i>	<i>lucien favre is the coach of bvb dortmund</i>
when did italy last win the world cup	2006	<i>italy won the world cup in 2006</i>	<i>i think they have a good team</i>	<i>italy won the world cup in 2006</i>
what time is my doctor appointment?	your doctor appointment is scheduled for friday at 11am	<i>your next is is at 1pm at 7pm</i>	<i>your doctor appointment is at 1pm</i>	<i>your doctor appointment is on friday at 11am</i>
i need gas	valero is 4_miles away	<i>there is a valero away</i>	<i>chevron is gas_station away chevron is at away</i>	<i>there is a valero nearby</i>

For in-car dialogues, we train and evaluate on queries which require knowledge from the KG, hence we omit the scheduling conversations where the driver asks the car to schedule a meeting/conference. In order to get the Knowledge Graph candidate triples for all the other models (Mem2Seq and GLMP), we provide them with the 2-hop sub-graph of the correct input entity instead of the local knowledge as used in the respective models; this, represents a scenario closer to a real-world KG grounded conversation. For the in-car dataset, the human baseline BLEU score as reported in [20] is 13.5 (the KGIRNet score is 11.76). The BLEU scores for soccer are low because non-goal oriented dialogues are more complicated to model due to large vocabulary sizes (more than 3 times the vocabulary size of the of in-car dataset). Also in the case of soccer dialogues, number of factual conversation is low

(4%) compared to the total number of utterances and also the conversations are more generic and repetitive in nature.

We also performed a human-based evaluation on the whole test dataset of the generated responses from the KGIRNet model and its closest competitors i.e. Mem2Seq, GLMP and DialoGPT models. We asked 3 (1 from CS and 2 from non-CS background) annotators to rate the quality of the generated responses between 1-5 with respect to the dialogue context (higher is better). The measures requested to evaluate upon are correctness (Corr.) and human-like (human) answer generation capabilities. The Cohen’s kappa of the annotations is 0.55. The results are reported in Table 8.6(a). It is evident from the results that the correctness of KGIRNet is the highest on both datasets with a larger margin in the in-car setting. DialoGPT produces more human-like answers in the case of soccer, but performs worse in the case of in-car dialogues.

8.5.1 Ablation Study

As an ablation study we train a sequence-to-sequence model with pre-trained fasttext embeddings as input (S2S), and the same model with pre-trained BERT as input embedding (S2S_BERT). Both these models do not have any information about the structure of the underlying Knowledge Graph. Secondly, we try to understand how much the intermediate representation aids to the model, so we train a model (KGIRNet_NB) with fasttext embeddings as input instead of BERT along with intermediate relation representation. Thirdly, we train a model with pre-trained BERT but without the haddamard product of the encoded sub-graph and the final output vocabulary distribution from Step 8.2.6. This model is denoted as KGIRNet_NS. As observed, models that are devoid of any KG structure has very low entity F1 scores, which is more observable in the case of soccer dialogues since the knowledge grounded queries are very low, so the model is not able to learn any fact(s) from the dataset. The proposed intermediate relation learning along with Graph Laplacian based sub-graph encoding technique observably assists in producing better knowledge grounded responses in both the domains; although, the latter aids in more knowledge groundedness in the domain of soccer dialogues (higher entity F1 scores). We also did an ablation study on the entity detection accuracy of the end-to-end KGIRNet model in the domain of in-car and compared it with a standalone Convolutional neural network (CNN) model which predicts the entity from the input query, the accuracies are 79.69% and 77.29% respectively.

Table 8.5: In-depth evaluation of KGIRNet model.

Models	In-Car		Soccer		Models	In-Car Dialogues		Soccer Dialogues	
	Corr.	Human	Corr.	Human		BLEU	EntityF1	BLEU	EntityF1
Mem2Seq	3.09	3.70	1.14	3.48	S2S	4.96	10.70	1.49	0.0
GLMP	3.01	3.88	1.10	2.17	S2S_BERT	7.53	09.10	1.44	0.0
DialoGPT	2.32	3.43	1.32	3.88	KGIRNet_NB	9.52	29.03	0.91	29.85
KGIRNet	3.60	4.42	1.59	3.78	KGIRNet_NS	11.40	33.03	1.05	28.35
					KGIRNet	11.76	32.67	1.51	34.32

(a) Human evaluation.

(b) Ablation study.

8.5.2 Qualitative Analysis

The responses generated by KGIRNet are analyzed in this section. Responses from some of the discussed models along with the input query are provided in Table 8.4. We compare the results with two other state-of-the-art models with the closest evaluation scores, namely Mem2Seq and GLMP. The first two responses are for soccer dialogues, while the latter two are for in-car setting. We inspect that qualitatively the proposed KGIRNet model produces better knowledge grounded and coherent responses in both the settings. In the case of soccer dialogues, predicting single relation in the response is sufficient, while for the case of in-car dialogues, responses can require multiple relation identification. KGIRNet model is able to handle such multiple relations as well (e.g., *r:date* friday and *r:time* 11am for the third utterance).

Table 8.7: Analysing fact-fullness of KGIRNet.

Input Query	True Response	Predicted Response	Intermediate Response
who is senegal captain ?	cheikhou kouyate	<i>sadio mane is the captain of senegal</i>	<i>r:captain is the captain of @entity</i>
who is the goalie for iceland	hannes halldorsson	<i>runar alex runarsson</i>	<i>r:goalkeeper</i>

To understand more about the advantages of KG grounded generation process, consider the examples from the soccer dataset below in Table 8.7, in which the model has failed. We have also included the intermediate responses that are generated from the decoder before the KG lookup phase for the reference. In the first case, while constructing the KG, the captain was *Sadio Mane*, whereas the during the dialogue collection phase the probable captain was *Cheikhou Kouyate* (as per the annotator). These errors on temporal information can easily be tackled using Knowledge Grounded dialogue systems, where the system can potentially reflect current information with updated real world knowledge. For the second case, the current main goalkeeper for the Iceland national football team is *Hannes Halldórsson*; however, *Rúnar Alex Rúnarsson* is also among the 3 goalkeepers for the team and the model has no additional information to disambiguate between them to produce the same response as the human annotator. KGIRNet is also able to infer conversational context implicitly in dialogues with the system (Table 8.8).

8.5.3 Error Analysis

To analyze more on the shortcomings of our proposed KGIRNet, we stated some sample erroneous responses from the model in Table 8.9. The first three are from soccer and the rest in-car dialogues. The model often produces generic responses for soccer as in the first example. Our model is only able to handle simple factoid queries and not able to answer queries requiring comparative reasoning such as *best player*, *youngest player* etc, as observable from the second and third cases. In the second case, the model answers with the coach’s name and when asked for the best player for Argentina and it replies with the name of an Uruguayan player in the third case. The model also produces generic relations like coach name, number of world cup championship as response when any such keyword is present in the input. For the in-car dataset, the replies are less generic and more coherent; however, in

Table 8.8: A conversation with KGIRNet.

User:	what s the weather forecast for today and tomorrow ?
KGIRNet:	what city do you want the weather for
User:	los angeles
KGIRNet:	it will be 40f - 60f on monday warm on tuesday windy on wednesday hot on thursday
User:	thanks
KGIRNet:	you are welcome

cases like the fourth example, the model is not able to perform logical reasoning (negation). Moreover, the model fails in cases where the input entity is not recognized correctly as in case of the last example. Although, the name of the friend is fetched from the vocabulary, but the predicted entity is wrong hence the model is not able to decode the relation r :address from the KG during decoding.

Table 8.9: Error analysis of KGIRNet.

Input Query	True Response	Predicted Response
portugal win the last eu- rocup ?	yes they won the last eu- ropcup	<i>portugal has won the world cup.</i>
who is switzerland's best player ?	xherdan shaqiri	<i>petkovic is the coach of switzerland</i>
who is considered their best player ?	traditionally their top scor- ers.	<i>luis suarez is the best player in argentina</i>
will it be warm in grand_rapids on saturday	it will not be warm in grand_rapids on saturday	<i>it is not predicted to fall on tuesday in grand_rapids</i>
where does my friend live ?	tom lives at 580_van_ness_ave	<i>tom s house is located at r:address</i>

In general, the model's entity detection fails in case of conversations with long dialogue contexts. Incorporating KG information into the entity detection process, where we consider the sub-graph of the entity in the first interaction can potentially help in further performance improvements.

Conclusion and Future Directions

Integrating external knowledge into human-machine interactive systems is one of the most important steps in order to achieve general artificial intelligence. Inspired by previous works in cognition, this research is primarily focused on integrating knowledge into text-based interactive systems. The main focus of this research is to answer the research question: *How to integrate external knowledge into text-based human machine interactive systems?*, as defined in 1. Several end-to-end as well as modular architectures are proposed to integrate external knowledge into question answering and dialogue systems. As also mentioned previously, the focus of this thesis is to integrate both structured and unstructured knowledge into interactive systems, but the main focus has been on integrating structure knowledge available in the form of Knowledge Graphs. The overall summary of the contributions are depicted in Fig 9.1.

9.1 Answering the Research Questions

* Research Question 1

How the Knowledge Graph Structure can be incorporated for Knowledge Graph Question Answering?

In order to answer the first research question (RQ1) three models, namely EARL, PNEL and ELiDi. EARL, is a modular architecture leveraging connected relation information for the task of entity linking while the other two namely PNEL and ELiDi being end-to-end models. A comparison is made with the other state-of-the-art models for entity linking models for the final task of Knowledge Graph based question answering. A final evaluation of all the models is performed on the task of simple question answering. As evident from the results, end-to-end approaches perform better on the task of entity linking. Additionally, as also evident ELiDi is also able to generalize better to unseen data in a zero-shot scenario.

* Research Question 2

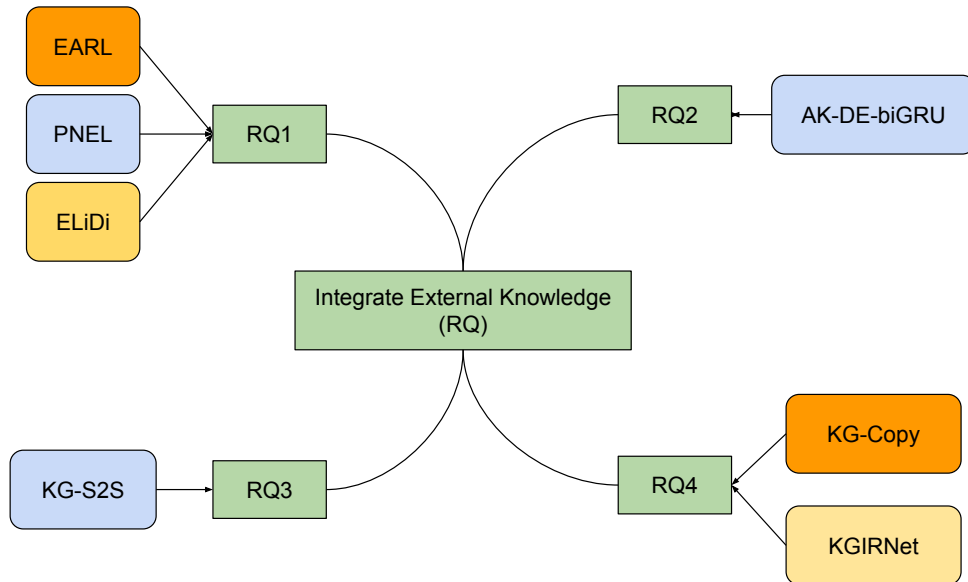


Figure 9.1: A Figure depicting the overall contribution of this Thesis towards Answering the main Research Question(RQ).

Can Domain Knowledge in the form of Unstructured Text improve Chatbots’ performance?

For answering the second research question, a novel end-to-end neural network architecture is proposed namely AK-DE-biGRU. The model is able to integrate additional knowledge in the form of domain keyword definitions into the final task of retrieval-based dialogue. An additional GRU is used to encode the keyword definition along with the query tokens with a novel gating mechanism. The experimental results demonstrate that our model outperformed other state-of-the-art methods for response selection in a multi-turn dialogue setting measured in term of recall @top-k, and that the attention mechanism and incorporating additional domain knowledge are indeed effective approaches for improving the response selection performance of the dual encoder architecture.

• **Research Question 3**

Can Jointly-Trained KG and Word Embeddings improve Task-Oriented Dialogue Systems?

To answer the third research question, we propose a novel neural network architecture called KG-S2S to improve goal-oriented dialogue systems using joint text and Knowledge Graph embeddings with learning user query intent as a multi-task learning paradigm. The model is enhanced with further regularization techniques based on entity loss which aids in further performance improvements. Empirical evaluations suggest that our suggested KG-S2S model gains over existing state-of-the-art systems on BLEU scores for multi-domain, task-oriented dialogue systems. Although, we are not reporting entity-f1 scores since we are treating canonicalized entity forms as vocabulary tokens and BLEU scores will already reflect the presence of the correct entity in the responses.

- **Research Question 4**

How response generation in a Generative Dialogue system can be made Knowledge Graph aware?

We have a two-fold approach to answer the fourth and the last research question. Firstly, a new dataset is proposed for the task of non-goal oriented knowledge-grounded dialogue generation. Secondly, we propose two end-to-end models namely KG-Copy and KGIRNet for the task of dialogue generation in both goal and non-goal oriented setting. KG-Copy is based on copying facts (objects) from the Knowledge Graph while generation responses while the later is based on generating relation(s) during the generation phase using graph laplacian and relation aware decoding. Empirical evidence suggest that KG-Copy performs better than other state-of-the-art models wrt. BLEU and Entity-f1 scores, while KGIRNet further improves on the results and also produces more coherent responses as evident from the human evaluations.

9.2 Future Directions

The main future research directions we would like to extend the ideas proposed in this thesis are

9.2.1 Transfer Learning for KGQA

For the task of Knowledge Graph based question answering, we would like to focus on transfer learning methodologies and adapt pre-trained models for the task. For entity linking, we would like to use pre-trained transformers those are fine-tuned for the task of named entity recognition and adapt them for entity linking for questions by incorporating additional Knowledge Graph information.

The biggest challenge for relation linking for KGQA is that the models are not able to adapt to handle relations that are not seen in the training phase [184]. We would like to handle this using relation label information during the training process and use pre-trained models and use them in a discriminative manner using relation label and questions for relation linking.

9.2.2 Annealing Language Models with Knowledge Graph Information

Although, pre-trained transformer based language models exhibits good understanding of text and can easily be fine-tuned into different tasks with improved performance. However, such models often fail when it comes to generating knowledge grounded responses. As also evident from the results of DialoGPT [139] from 8.2, although it produces more natural responses but fails in the cases where additional knowledge was required. As a future work, we would like to focus on adapting pre-trained language models to be more knowledge aware and use them for the purpose of dialogue generation.

9.2.3 Evaluating Knowledge Groundedness of Generated Dialogues

Another future directions we would like to explore is having automatic metrics for evaluation of knowledge groundedness for the evaluated dialogues. Although, automatic evaluation of generated dialogues, or natural language generation is a well-researched topics, only few recent works have focused on factual correction of the generated dialogues. Evaluating knowledge grounded dialogues would depend on the query posed, and the entities generated during the dialogue generation process. We would like to investigate automated evaluation metrics which will also consider the underlying knowledge graph while evaluating the generation dialogues. Potentially using jointly trained Knowledge Graph embeddings can be used for designing such evaluation metrics.

Bibliography

- [1] T. Winograd. “Understanding natural language.”
In: *Cognitive psychology* 3.1 (1972), pp. 1–191 (cit. on p. 1).
- [2] D. H. Warren and F. C. Pereira.
“An efficient easily adaptable system for interpreting natural language queries.”
In: *American journal of computational linguistics* 8.3-4 (1982), pp. 110–122 (cit. on p. 1).
- [3] S. Hussain, O. A. Sianaki, and N. Ababneh.
“A survey on conversational agents/chatbots classification and design techniques.”
In: *Workshops of the International Conference on Advanced Information Networking and Applications*. Springer. 2019, pp. 946–956 (cit. on p. 1).
- [4] C. C. Shilakes. “Enterprise information portals.” In: *Merrill Lynch in-depth report* (1998) (cit. on p. 1).
- [5] L. Ehrlinger and W. Wöß. “Towards a Definition of Knowledge Graphs.”
In: *SEMANTiCS (Posters, Demos, SuCCESS)* 48 (2016), pp. 1–4 (cit. on p. 1).
- [6] S. Mohammed, P. Shi, and J. Lin. “Strong Baselines for Simple Question Answering over Knowledge Graphs with and without Neural Networks.”
In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 2018, pp. 291–296 (cit. on pp. 1, 29, 53, 54, 57–61, 106, 107).
- [7] D. Lukovnikov et al. “Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level.”
In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 1211–1220 (cit. on pp. 1, 4, 29).
- [8] M. Yu et al. “Improved Neural Relation Detection for Knowledge Base Question Answering.”
In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
Vancouver, Canada: Association for Computational Linguistics, 2017, pp. 571–581.
DOI: [10.18653/v1/P17-1053](https://doi.org/10.18653/v1/P17-1053).
URL: <http://aclweb.org/anthology/P17-1053> (cit. on pp. 1, 29, 30, 106, 107).
- [9] R. Lowe et al.
“Incorporating unstructured textual knowledge sources into neural dialogue systems.”
In: *Neural Information Processing Systems Workshop on Machine Learning for Spoken Language Understanding*. 2015 (cit. on pp. 1, 4, 32).

- [10] A. Madotto, C.-S. Wu, and P. Fung. “Mem2Seq: Effectively Incorporating Knowledge Bases into End-to-End Task-Oriented Dialog Systems.” In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 1468–1478. DOI: [10.18653/v1/P18-1136](https://doi.org/10.18653/v1/P18-1136). URL: <https://www.aclweb.org/anthology/P18-1136> (cit. on pp. 1, 31, 75, 82, 95, 99, 105, 106).
- [11] M. Eric and C. D. Manning. “Key-Value Retrieval Networks for Task-Oriented Dialogue.” In: *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. Saarbrücken, Germany: Association for Computational Linguistics, Aug. 2017, pp. 37–49. URL: <http://www.aclweb.org/anthology/W17-36%206> (cit. on pp. 1, 80, 82, 87, 88, 96).
- [12] C. Spearman. ““ General Intelligence” Objectively Determined and Measured.” In: (1961) (cit. on pp. 2, 3).
- [13] K. S. McGrew. “The cattell-horn-carroll theory of cognitive abilities: past, present, and future.” In: (2005) (cit. on p. 2).
- [14] A. Newell, J. Shaw, and H. A. Simon. “A general problem-solving program for a computer.” In: *Computers and Automation* 8.7 (1959), pp. 10–16 (cit. on p. 3).
- [15] O. Wilhelm and R. W. Engle. *Handbook of understanding and measuring intelligence*. Sage Publications, 2004 (cit. on p. 3).
- [16] R. Schank and L. Birnbaum. “Enhancing intelligence.” In: *What is intelligence?* Cambridge University Press, 1994, pp. 72–106 (cit. on p. 3).
- [17] A. K. Barbey. “Network Neuroscience Theory of Human Intelligence.” In: *Trends in Cognitive Sciences* 22.1 (2018), pp. 8–20. ISSN: 1364-6613. DOI: <https://doi.org/10.1016/j.tics.2017.10.001>. URL: <http://www.sciencedirect.com/science/article/pii/S1364661317302218> (cit. on p. 3).
- [18] S. Mohammed, P. Shi, and J. Lin. “Strong Baselines for Simple Question Answering over Knowledge Graphs with and without Neural Networks.” In: *arXiv preprint arXiv:1712.01969* (2017) (cit. on p. 4).
- [19] Z. Xu et al. “Incorporating loose-structured knowledge into lstm with recall gate for conversation modeling.” In: *arXiv preprint arXiv:1605.05110* 3 (2016) (cit. on pp. 4, 32, 69, 70).
- [20] M. Eric et al. “Key-Value Retrieval Networks for Task-Oriented Dialogue.” In: *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. Saarbrücken, Germany: Association for Computational Linguistics, Aug. 2017, pp. 37–49. DOI: [10.18653/v1/W17-5506](https://doi.org/10.18653/v1/W17-5506). URL: <https://www.aclweb.org/anthology/W17-5506> (cit. on pp. 4, 31, 99, 104, 107).
- [21] N. Chomsky. “The structure of language.” In: (1957) (cit. on p. 9).

-
- [22] Z. S. Harris. “Distributional structure.” In: *Word* 10.2-3 (1954), pp. 146–162 (cit. on p. 9).
- [23] R. Blumberg and S. Atre. “The problem with unstructured data.” In: *Dm Review* 13.42-49 (2003), p. 62 (cit. on p. 10).
- [24] R. Sint et al.
“Combining unstructured, fully structured and semi-structured information in semantic wikis.” In: *CEUR Workshop Proceedings*. Vol. 464. 2009, pp. 73–87 (cit. on p. 10).
- [25] S. Abiteboul, P. Buneman, and D. Suciu.
Data on the Web: from relations to semistructured data and XML. Morgan Kaufmann, 2000 (cit. on p. 10).
- [26] S. Ji et al. *A Survey on Knowledge Graphs: Representation, Acquisition and Applications*. 2020. arXiv: 2002.00388 [cs.CL] (cit. on p. 11).
- [27] X. Glorot, A. Bordes, and Y. Bengio. “Deep Sparse Rectifier Neural Networks.” In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*. Ed. by G. J. Gordon, D. B. Dunson, and M. Dudik. Vol. 15. JMLR Proceedings. JMLR.org, 2011, pp. 315–323. URL: <http://jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf> (cit. on p. 12).
- [28] I. J. Goodfellow, Y. Bengio, and A. C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. ISBN: 978-0-262-03561-3. URL: <http://www.deeplearningbook.org/> (cit. on pp. 13, 15, 20).
- [29] J. Devlin et al.
“BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: <http://arxiv.org/abs/1810.04805> (cit. on p. 14).
- [30] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004 (cit. on p. 14).
- [31] P. Auer, M. Herbster, and M. K. Warmuth.
“Exponentially many local minima for single neurons.” In: *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995*. Ed. by D. S. Touretzky, M. Mozer, and M. E. Hasselmo. MIT Press, 1995, pp. 316–322. ISBN: 0-262-20107-0. URL: <http://papers.nips.cc/paper/1028-exponentially-many-local-minima-for-single-neurons> (cit. on p. 14).
- [32] A. Choromanska et al. “The Loss Surfaces of Multilayer Networks.” In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*. Ed. by G. Lebanon and S. V. N. Vishwanathan. Vol. 38. JMLR Workshop and Conference Proceedings. JMLR.org, 2015. URL: <http://jmlr.org/proceedings/papers/v38/choromanska15.html> (cit. on p. 14).

- [33] S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 2015, pp. 448–456. URL: <http://jmlr.org/proceedings/papers/v37/ioffe15.html> (cit. on p. 14).
- [34] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization.” In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: <http://arxiv.org/abs/1412.6980> (cit. on p. 14).
- [35] I. Loshchilov and F. Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts.” In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017. URL: <https://openreview.net/forum?id=Skq89Scxx> (cit. on p. 14).
- [36] Y. Bengio et al. “Curriculum learning.” In: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*. Ed. by A. P. Danyluk, L. Bottou, and M. L. Littman. Vol. 382. ACM International Conference Proceeding Series. ACM, 2009, pp. 41–48. ISBN: 978-1-60558-516-1. DOI: [10.1145/1553374.1553380](https://doi.org/10.1145/1553374.1553380). URL: <https://doi.org/10.1145/1553374.1553380> (cit. on p. 14).
- [37] D. E. Rumelhart, G. E. Hinton, R. J. Williams, et al. “Learning representations by back-propagating errors.” In: *Cognitive modeling* 5.3 (1988), p. 1 (cit. on p. 16).
- [38] S. Hochreiter. “Untersuchungen zu dynamischen neuronalen Netzen.” In: *Diploma, Technische Universität München* 91.1 (1991) (cit. on p. 17).
- [39] Y. Bengio, P. Frasconi, and P. Simard. “The problem of learning long-term dependencies in recurrent networks.” In: *IEEE international conference on neural networks*. IEEE. 1993, pp. 1183–1188 (cit. on p. 17).
- [40] Y. Bengio, P. Y. Simard, and P. Frasconi. “Learning long-term dependencies with gradient descent is difficult.” In: *IEEE Trans. Neural Networks* 5.2 (1994), pp. 157–166. DOI: [10.1109/72.279181](https://doi.org/10.1109/72.279181). URL: <https://doi.org/10.1109/72.279181> (cit. on pp. 17, 65).
- [41] S. Hochreiter and J. Schmidhuber. “Long short-term memory.” In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on pp. 17, 63, 77, 102).
- [42] C. Olah. *Understanding LSTM Networks*. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Last checked on May 04, 2019. Aug. 2015. URL: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> (cit. on p. 17).

-
- [43] K. Cho et al. “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 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2014, pp. 1724–1734.
URL: <http://aclweb.org/anthology/D/D14/D14-1179.pdf> (cit. on p. 18).
- [44] G.-B. Zhou et al. “Minimal Gated Unit for Recurrent Neural Networks.”
In: *CoRR* abs/1603.09420 (2016). arXiv: 1603.09420.
URL: <http://arxiv.org/abs/1603.09420> (cit. on p. 18).
- [45] O. Vinyals, M. Fortunato, and N. Jaitly. “Pointer Networks.”
In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 2692–2700.
URL: <http://papers.nips.cc/paper/5866-pointer-networks.pdf>
(cit. on pp. 18, 44).
- [46] I. Sutskever, O. Vinyals, and Q. V. Le.
“Sequence to Sequence Learning with Neural Networks.”
In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 3104–3112.
URL: <http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf> (cit. on p. 18).
- [47] D. Bahdanau, K. Cho, and Y. Bengio.
“Neural machine translation by jointly learning to align and translate.”
In: *arXiv preprint arXiv:1409.0473* (2014) (cit. on pp. 18, 77, 92).
- [48] A. Vaswani et al. “Attention is All you Need.”
In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. Ed. by I. Guyon et al. 2017, pp. 6000–6010. URL:
<http://papers.nips.cc/paper/7181-attention-is-all-you-need>
(cit. on pp. 19, 31, 102, 106).
- [49] J. Devlin et al.
“BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.”
In: *NAACL-HLT (1)*. 2019 (cit. on pp. 19, 24, 29, 31, 58, 102).
- [50] E. Egonmwan and Y. Chali.
“Transformer-based Model for Single Documents Neural Summarization.”
In: *Proceedings of the 3rd Workshop on Neural Generation and Translation*. 2019, pp. 70–79
(cit. on pp. 19, 31).
- [51] Y. LeCun et al. “Backpropagation Applied to Handwritten Zip Code Recognition.”
In: *Neural Computation* 1.4 (1989), pp. 541–551. doi: 10.1162/neco.1989.1.4.541.
URL: <https://doi.org/10.1162/neco.1989.1.4.541> (cit. on p. 20).

- [52] H. Zhao et al. “Pyramid Scene Parsing Network.” In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 6230–6239. ISBN: 978-1-5386-0457-1. DOI: [10.1109/CVPR.2017.660](https://doi.org/10.1109/CVPR.2017.660). URL: <https://doi.org/10.1109/CVPR.2017.660> (cit. on p. 20).
- [53] Z. Cao et al. “Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields.” In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 1302–1310. ISBN: 978-1-5386-0457-1. DOI: [10.1109/CVPR.2017.143](https://doi.org/10.1109/CVPR.2017.143). URL: <https://doi.org/10.1109/CVPR.2017.143> (cit. on p. 20).
- [54] R. B. Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.” In: *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE Computer Society, 2014, pp. 580–587. ISBN: 978-1-4799-5118-5. DOI: [10.1109/CVPR.2014.81](https://doi.org/10.1109/CVPR.2014.81). URL: <https://doi.org/10.1109/CVPR.2014.81> (cit. on p. 20).
- [55] Y. Kim. “Convolutional Neural Networks for Sentence Classification.” In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2014, pp. 1746–1751. URL: <http://aclweb.org/anthology/D/D14/D14-1181.pdf> (cit. on p. 20).
- [56] L. Mou et al. “Discriminative Neural Sentence Modeling by Tree-Based Convolution.” In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. 2015, pp. 2315–2325. URL: <http://aclweb.org/anthology/D/D15/D15-1279.pdf> (cit. on p. 20).
- [57] Y. Goldberg. *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2017. DOI: [10.2200/S00762ED1V01Y201703HLT037](https://doi.org/10.2200/S00762ED1V01Y201703HLT037). URL: <https://doi.org/10.2200/S00762ED1V01Y201703HLT037> (cit. on p. 23).
- [58] T. Mikolov et al. “Efficient estimation of word representations in vector space.” In: *arXiv preprint arXiv:1301.3781* (2013) (cit. on pp. 23, 78).
- [59] Z. Dai et al. “Transformer-XL: Attentive Language Models beyond a Fixed-Length Context.” In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2978–2988 (cit. on p. 24).
- [60] A. Bordes et al. “Translating Embeddings for Modeling Multi-relational Data.” In: *NIPS*. 2013 (cit. on p. 25).
- [61] Z. Sun et al. “RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space.” In: *ArXiv abs/1902.10197* (2019) (cit. on p. 25).

-
- [62] H. Xiao, M. Huang, and X. Zhu. “TransG : A Generative Model for Knowledge Graph Embedding.” In: *ACL*. 2016 (cit. on p. 25).
- [63] C. Xu and R. Li. “Relation Embedding with Dihedral Group in Knowledge Graph.” In: *ACL*. 2019 (cit. on p. 25).
- [64] B. Yang et al. “Embedding Entities and Relations for Learning and Inference in Knowledge Bases.” In: *CoRR* abs/1412.6575 (2015) (cit. on p. 25).
- [65] I. Balazevic, C. Allen, and T. M. Hospedales. “TuckER: Tensor Factorization for Knowledge Graph Completion.” In: *EMNLP/IJCNLP*. 2019 (cit. on p. 25).
- [66] R. Socher et al. “Reasoning With Neural Tensor Networks for Knowledge Base Completion.” In: *NIPS*. 2013 (cit. on p. 25).
- [67] T. Dettmers et al. “Convolutional 2D Knowledge Graph Embeddings.” In: *AAAI*. 2017 (cit. on p. 25).
- [68] L. Guo, Z. Sun, and W. Hu. “Learning to Exploit Long-term Relational Dependencies in Knowledge Graphs.” In: *ArXiv* abs/1905.04914 (2019) (cit. on p. 25).
- [69] L. Yao, C. Mao, and Y. Luo. “KG-BERT: BERT for Knowledge Graph Completion.” In: *ArXiv* abs/1909.03193 (2019) (cit. on p. 25).
- [70] C. Shang et al. “End-to-end Structure-Aware Convolutional Networks for Knowledge Base Completion.” In: *ArXiv* abs/1811.04441 (2018) (cit. on p. 25).
- [71] A. Lerer et al. “PyTorch-BigGraph: A Large-scale Graph Embedding System.” In: *Proceedings of the 2nd SysML Conference*. Palo Alto, CA, USA, 2019 (cit. on p. 25).
- [72] J. Cannan and H. Hu. “Human-machine interaction (HMI): A survey.” In: *University of Essex* (2011) (cit. on p. 27).
- [73] P. Jia and H. Hu. “Active shape model-based user identification for an intelligent wheelchair.” In: *International Journal of Advanced Mechatronic Systems* 1.4 (2009), pp. 299–307 (cit. on p. 27).
- [74] I. S. MacKenzie. “An eye on input: research challenges in using the eye for computer input control.” In: *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*. 2010, pp. 11–12 (cit. on p. 27).
- [75] T. Kopinski, S. Geisler, and U. Handmann. “Gesture-based human-machine interaction for assistance systems.” In: *2015 IEEE International Conference on Information and Automation*. IEEE. 2015, pp. 510–517 (cit. on p. 27).

- [76] F. Echtler, T. Pototschnig, and G. Klinker. “An LED-based multitouch sensor for LCD screens.” In: *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*. 2010, pp. 227–230 (cit. on p. 27).
- [77] H. Roeber, J. Bacus, and C. Tomasi. “Typing in thin air: the canesta projection keyboard—a new method of interaction with electronic devices.” In: *CHI’03 extended abstracts on Human factors in computing systems*. 2003, pp. 712–713 (cit. on p. 27).
- [78] F. Templier et al. “High-resolution active-matrix 10-um pixel-pitch GaN LED microdisplays for augmented reality applications.” In: *Advances in Display Technologies VIII*. Vol. 10556. International Society for Optics and Photonics. 2018, p. 105560I (cit. on p. 27).
- [79] Y. Matsuoka, P. Afshar, and M. Oh. “On the design of robotic hands for brain–machine interface.” In: *Neurosurgical focus* 20.5 (2006), pp. 1–9 (cit. on p. 27).
- [80] T. W. Berger et al. “Brain-implantable biomimetic electronics as the next era in neural prosthetics.” In: *Proceedings of the IEEE* 89.7 (2001), pp. 993–1012 (cit. on p. 27).
- [81] M. Bear, B. Connors, and M. A. Paradiso. *Neuroscience: Exploring the brain*. Jones & Bartlett Learning, LLC, 2020 (cit. on p. 27).
- [82] K. R. Wheeler. “Device control using gestures sensed from EMG.” In: *Proceedings of the 2003 IEEE International Workshop on Soft Computing in Industrial Applications, 2003. SMCia/03*. IEEE. 2003, pp. 21–26 (cit. on p. 27).
- [83] L. Y. Deng et al. “EOG-based Human–Computer Interface system development.” In: *Expert Systems with Applications* 37.4 (2010), pp. 3337–3343 (cit. on p. 27).
- [84] S. Tomko et al. “Towards efficient human machine speech communication: The speech graffiti project.” In: *ACM Transactions on Speech and Language Processing (TSLP)* 2.1 (2005), 2–es (cit. on p. 28).
- [85] I. Tashev et al. “Commute UX: Voice enabled in-car infotainment system.” In: (2009) (cit. on p. 28).
- [86] J. Williams et al. “The dialog state tracking challenge.” In: *Proceedings of the SIGDIAL 2013 Conference*. 2013, pp. 404–413 (cit. on pp. 28, 30).
- [87] C. Baber. “Human factors aspects of automatic speech recognition in control room environments.” In: *IEE Colloquium on Systems and Applications of Man-Machine Interaction Using Speech I/O*. IET. 1991, pp. 10–1 (cit. on p. 28).
- [88] R. Corkrey and L. Parkinson. “Interactive voice response: Review of studies 1989–2000.” In: *Behavior Research Methods, Instruments, & Computers* 34.3 (2002), pp. 342–353 (cit. on p. 28).

-
- [89] A. Trotman, S. Geva, and J. Kamps. “Report on the SIGIR 2007 workshop on focused retrieval.” In: *ACM SIGIR Forum*. Vol. 41. 2. ACM New York, NY, USA. 2007, pp. 97–103 (cit. on p. 28).
- [90] Y. Yang, W.-t. Yih, and C. Meek. “Wikiqa: A challenge dataset for open-domain question answering.” In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015, pp. 2013–2018 (cit. on p. 28).
- [91] V. Yadav, R. Sharp, and M. Surdeanu. “Sanity check: A strong alignment and information retrieval baseline for question answering.” In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 2018, pp. 1217–1220 (cit. on p. 28).
- [92] S. Momtazi and D. Klakow. “A word clustering approach for language model-based sentence retrieval in question answering systems.” In: *Proceedings of the 18th ACM conference on Information and knowledge management*. 2009, pp. 1911–1914 (cit. on p. 28).
- [93] S. Momtazi and D. Klakow. “Bridging the vocabulary gap between questions and answer sentences.” In: *Information Processing & Management* 51.5 (2015), pp. 595–615 (cit. on p. 28).
- [94] L. Yu et al. “Deep learning for answer sentence selection.” In: *arXiv preprint arXiv:1412.1632* (2014) (cit. on p. 28).
- [95] W. Yin et al. “Simple Question Answering by Attentive Convolutional Neural Network.” In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, 2016, pp. 1746–1756. URL: <http://aclweb.org/anthology/C16-1164> (cit. on pp. 28, 29, 58).
- [96] D. Wang and E. Nyberg. “A long short-term memory model for answer sentence selection in question answering.” In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 2015, pp. 707–712 (cit. on p. 28).
- [97] Y. Tay et al. “Learning to rank question answer pairs with holographic dual lstm architecture.” In: *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 2017, pp. 695–704 (cit. on p. 28).
- [98] M. Tan et al. “Improved representation learning for question answer matching.” In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 464–473 (cit. on p. 28).
- [99] L. Yang et al. “aNMM: Ranking short answer texts with attention-based neural matching model.” In: *Proceedings of the 25th ACM international on conference on information and knowledge management*. 2016, pp. 287–296 (cit. on p. 29).

- [100] S. Wan et al. “Match-srnn: Modeling the recursive matching structure with spatial rnn.” In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. 2016 (cit. on pp. 29, 32, 69, 70).
- [101] K. M. Hermann et al. “Teaching machines to read and comprehend.” In: *Advances in neural information processing systems* 28 (2015), pp. 1693–1701 (cit. on p. 29).
- [102] D. Chen, J. Bolton, and C. D. Manning. “A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task.” In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 2358–2367 (cit. on p. 29).
- [103] W. Wang et al. “Gated Self-Matching Networks for Reading Comprehension and Question Answering.” In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 189–198. DOI: [10.18653/v1/P17-1018](https://doi.org/10.18653/v1/P17-1018). URL: <https://www.aclweb.org/anthology/P17-1018> (cit. on p. 29).
- [104] R. Kadlec et al. “Text Understanding with the Attention Sum Reader Network.” 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. 908–918. DOI: [10.18653/v1/P16-1086](https://doi.org/10.18653/v1/P16-1086). URL: <https://www.aclweb.org/anthology/P16-1086> (cit. on p. 29).
- [105] Z. Yang et al. “Xlnet: Generalized autoregressive pretraining for language understanding.” In: *Advances in neural information processing systems*. 2019, pp. 5753–5763 (cit. on p. 29).
- [106] A. Bordes et al. “Large-scale Simple Question Answering with Memory Networks.” In: *ArXiv abs/1506.02075* (2015) (cit. on pp. 29, 30, 33, 47, 58, 105).
- [107] X. He and D. Golub. “Character-Level Question Answering with Attention.” In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, 2016, pp. 1598–1607. DOI: [10.18653/v1/D16-1166](https://doi.org/10.18653/v1/D16-1166). URL: <http://aclweb.org/anthology/D16-1166> (cit. on p. 29).
- [108] Z. Dai, L. Li, and W. Xu. “CFO: Conditional Focused Neural Question Answering with Large-scale Knowledge Bases.” 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. 800–810. DOI: [10.18653/v1/P16-1076](https://doi.org/10.18653/v1/P16-1076). URL: <http://aclweb.org/anthology/P16-1076> (cit. on p. 29).
- [109] X. Huang et al. “Knowledge graph embedding based question answering.” In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 2019, pp. 105–113 (cit. on p. 30).

-
- [110] M. Petrochuk and L. Zettlemoyer.
“SimpleQuestions Nearly Solved: A New Upperbound and Baseline Approach.”
In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. 2018, pp. 554–558.
URL: <https://aclanthology.info/papers/D18-1051/d18-1051>
(cit. on pp. 30, 53, 107).
- [111] D. Vrandečić and M. Krötzsch. “Wikidata: a free collaborative knowledgebase.”
In: *Communications of the ACM* 57.10 (2014), pp. 78–85 (cit. on pp. 30, 57, 90).
- [112] J. Lehmann et al.
“DBpedia—a large-scale, multilingual knowledge base extracted from Wikipedia.”
In: *Semantic Web* 6.2 (2015), pp. 167–195 (cit. on p. 30).
- [113] D. Diefenbach et al. “Question answering benchmarks for wikidata.” In: *ISWC 2017*. 2017
(cit. on p. 30).
- [114] M. Azmy et al. “Farewell Freebase: Migrating the SimpleQuestions Dataset to DBpedia.”
In: *Proceedings of the 27th International Conference on Computational Linguistics*. 2018,
pp. 2093–2103 (cit. on p. 30).
- [115] C. Unger, A.-C. N. Ngomo, and E. Cabrio.
“6th open challenge on question answering over linked data (qald-6).”
In: *Semantic Web Evaluation Challenge*. Springer. 2016, pp. 171–177 (cit. on p. 30).
- [116] R. Usbeck et al. “7th Open Challenge on Question Answering over Linked Data (QALD-7).”
In: *Semantic Web Evaluation Challenge*. Springer. 2017, pp. 59–69 (cit. on p. 30).
- [117] R. Usbeck et al. “8th Challenge on Question Answering over Linked Data (QALD-8).”
In: *language* 7 (2018), p. 1 (cit. on p. 30).
- [118] D. Sorokin and I. Gurevych. “Modeling Semantics with Gated Graph Neural Networks for Knowledge Base Question Answering.”
In: *Proceedings of the 27th International Conference on Computational Linguistics*.
Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018,
pp. 3306–3317. URL: <http://aclweb.org/anthology/C18-1280> (cit. on p. 30).
- [119] M. A. Walker, R. Passonneau, and J. E. Boland.
“Quantitative and qualitative evaluation of DARPA Communicator spoken dialogue systems.”
In: *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*.
Association for Computational Linguistics. 2001, pp. 515–522 (cit. on p. 30).
- [120] J. M. M. E. F. Oliver and L. M. White.
“Generating tailored, comparative descriptions in spoken dialogue.” In: (2004) (cit. on p. 30).
- [121] A. Stent et al. “User-tailored generation for spoken dialogue: An experiment.”
In: *Seventh International Conference on Spoken Language Processing*. 2002 (cit. on p. 30).
- [122] T. Zhao and M. Eskenazi. “Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning.” In: *arXiv preprint arXiv:1606.02560* (2016)
(cit. on p. 30).
- [123] T.-H. Wen et al. “A network-based end-to-end trainable task-oriented dialogue system.”
In: *arXiv preprint arXiv:1604.04562* (2016) (cit. on pp. 30, 31, 80).

- [124] A. Bordes, Y.-L. Boureau, and J. Weston. “Learning end-to-end goal-oriented dialog.” In: *arXiv preprint arXiv:1605.07683* (2016) (cit. on pp. 30, 31).
- [125] L. Baird. “Residual algorithms: Reinforcement learning with function approximation.” In: *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 30–37 (cit. on p. 30).
- [126] A. Ritter, C. Cherry, and W. B. Dolan. “Data-driven response generation in social media.” In: *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics. 2011, pp. 583–593 (cit. on p. 30).
- [127] Z. Ji, Z. Lu, and H. Li. “An information retrieval approach to short text conversation.” In: *Proceedings of International Conference on Computation and Language*. 2014 (cit. on pp. 30, 31).
- [128] L. Shang, Z. Lu, and H. Li. “Neural Responding Machine for Short-Text Conversation.” 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, July 2015, pp. 1577–1586. DOI: [10.3115/v1/P15-1152](https://doi.org/10.3115/v1/P15-1152) (cit. on pp. 30, 76).
- [129] O. Vinyals and Q. V. Le. “A Neural Conversational Model.” In: *International Conference on Machine Learning* (2015) (cit. on pp. 30, 76).
- [130] T. Luong et al. “Addressing the Rare Word Problem in Neural Machine Translation.” 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, July 2015, pp. 11–19. DOI: [10.3115/v1/P15-1002](https://doi.org/10.3115/v1/P15-1002) (cit. on pp. 30, 76).
- [131] I. V. Serban et al. “Building end-to-end dialogue systems using generative hierarchical neural network models.” In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016 (cit. on p. 31).
- [132] A. Sordoni et al. “A neural network approach to context-sensitive generation of conversational responses.” In: *The 2015 Annual Conference of the North American Chapter of the ACL*. 2015 (cit. on p. 31).
- [133] B. Dhingra et al. “Towards end-to-end reinforcement learning of dialogue agents for information access.” In: *arXiv preprint arXiv:1609.00777* (2016) (cit. on p. 31).
- [134] F. Kassawat, D. Chaudhuri, and J. Lehmann. “Incorporating Joint Embeddings into Goal-Oriented Dialogues with Multi-task Learning.” In: *European Semantic Web Conference*. Springer. 2019, pp. 225–239 (cit. on pp. 31, 75).
- [135] C.-S. Wu, R. Socher, and C. Xiong. “Global-to-local Memory Pointer Networks for Task-Oriented Dialogue.” In: *arXiv preprint arXiv:1901.04713* (2019) (cit. on pp. 31, 99, 106).
- [136] D. Chaudhuri et al. “Using a KG-Copy Network for Non-Goal Oriented Dialogues.” In: *International Semantic Web Conference*. Springer. 2019, pp. 93–109 (cit. on pp. 31, 87, 104, 106).

-
- [137] T. Shao et al. “Transformer-based neural network for answer selection in question answering.” In: *IEEE Access* 7 (2019), pp. 26146–26156 (cit. on p. 31).
- [138] Q. Wang et al. “Learning deep transformer models for machine translation.” In: *arXiv preprint arXiv:1906.01787* (2019) (cit. on p. 31).
- [139] Y. Zhang et al.
DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation. 2019. arXiv: 1911.00536 [cs.CL] (cit. on pp. 31, 106, 114).
- [140] M. M. Bronstein et al. “Geometric deep learning: going beyond euclidean data.” In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42 (cit. on p. 31).
- [141] P. Veličković et al. “Graph attention networks.” In: *arXiv preprint arXiv:1710.10903* (2017) (cit. on p. 31).
- [142] T. N. Kipf and M. Welling.
“Semi-supervised classification with graph convolutional networks.”
In: *arXiv preprint arXiv:1609.02907* (2016) (cit. on pp. 31, 103).
- [143] D. Neil et al.
“Interpretable graph convolutional neural networks for inference on noisy knowledge graphs.”
In: *arXiv preprint arXiv:1812.00279* (2018) (cit. on p. 31).
- [144] W. Hao, L. Zhengdong, L. Hang, et al. “A dataset for research on short-text conversation.” In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, pp. 935–945 (cit. on p. 31).
- [145] Y. Wu et al. “Topic augmented neural network for short text conversation.”
In: *CoRR abs/1605.00090* (2016) (cit. on p. 31).
- [146] R. Lowe et al. “The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems.” In: *Proceedings of SIGDIAL*. 2015 (cit. on pp. 31, 63, 66, 69, 70).
- [147] R. Kadlec, M. Schmid, and J. Kleindienst.
“Improved deep learning baselines for ubuntu corpus dialogs.”
In: *NIPS on Machine Learning for Spoken Language Understanding*. 2015 (cit. on pp. 31, 69–71).
- [148] R. Yan, Y. Song, and H. Wu. “Learning to respond with deep neural networks for retrieval-based human-computer conversation system.” In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM. 2016, pp. 55–64 (cit. on pp. 31, 63, 69, 70).
- [149] X. Zhou et al. “Multi-view response selection for human-computer conversation.” In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 372–381 (cit. on pp. 32, 63, 69, 70).
- [150] G. An, M. Shafiee, and D. Shamsi. “Improving Retrieval Modeling Using Cross Convolution Networks And Multi Frequency Word Embedding.”
In: *arXiv preprint arXiv:1802.05373* (2018) (cit. on pp. 32, 63, 69).

- [151] J. Dong and J. Huang. “Enhance word representation for out-of-vocabulary on Ubuntu dialogue corpus.” In: *arXiv preprint arXiv:1802.02614* (2018) (cit. on pp. 32, 63, 69, 70).
- [152] P. Baudiš et al. “Sentence pair scoring: Towards unified framework for text comprehension.” In: *arXiv preprint arXiv:1603.06127* (2016) (cit. on p. 32).
- [153] S. Wang and J. Jiang. “Learning natural language inference with LSTM.” In: *Proceedings of NAACL-HLT 2016*. 2016 (cit. on pp. 32, 69, 70).
- [154] M. Tan et al. “LSTM-based deep learning models for non-factoid answer selection.” In: *Proceedings of the 4rd International Conference for Learning Representations*. 2016 (cit. on pp. 32, 69, 70).
- [155] M. Dubey et al. “EARL: joint entity and relation linking for question answering over knowledge graphs.” In: *International Semantic Web Conference*. Springer. 2018, pp. 108–126 (cit. on p. 33).
- [156] R. Collobert et al. “Natural language processing (almost) from scratch.” In: *Journal of Machine Learning Research* 12.Aug (2011), pp. 2493–2537 (cit. on p. 34).
- [157] Y. Pinter, R. Guthrie, and J. Eisenstein. “Mimicking Word Embeddings using Subword RNNs.” In: *EMNLP*. 2017, pp. 102–112 (cit. on p. 34).
- [158] P. Ristoski and H. Paulheim. “Rdf2vec: Rdf graph embeddings for data mining.” In: *International Semantic Web Conference*. Springer. 2016, pp. 498–514 (cit. on p. 34).
- [159] D. Kingma and J. Ba. “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on pp. 35, 47, 58, 70, 80, 94).
- [160] G. Laporte, H. Mercure, and Y. Nobert. “Generalized travelling salesman problem through n sets of nodes: the asymmetrical case.” In: *Discrete Applied Mathematics* 18.2 (1987), pp. 185–197. ISSN: 0166-218X. DOI: [https://doi.org/10.1016/0166-218X\(87\)90020-5](https://doi.org/10.1016/0166-218X(87)90020-5). URL: <http://www.sciencedirect.com/science/article/pii/0166218X87900205> (cit. on p. 36).
- [161] K. Helsgaun. “Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun Algorithm.” In: *Mathematical Programming Computation* (2015) (cit. on p. 36).
- [162] P. Trivedi et al. “Lc-quad: A corpus for complex question answering over knowledge graphs.” In: *International Semantic Web Conference*. Springer. 2017, pp. 210–218 (cit. on p. 40).
- [163] R. Speck and A.-C. Ngonga Ngomo. “Ensemble Learning for Named Entity Recognition.” In: *The Semantic Web – ISWC 2014*. Vol. 8796. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 519–534 (cit. on p. 42).
- [164] R. Usbeck et al. “AGDISTIS-graph-based disambiguation of named entities using linked data.” In: *International Semantic Web Conference*. Springer. 2014, pp. 457–471 (cit. on p. 42).

-
- [165] P. N. Mendes et al. “DBpedia spotlight: shedding light on the web of documents.”
In: *Proceedings of the 7th international conference on semantic systems*. 2011, pp. 1–8
(cit. on pp. 42, 48, 59).
- [166] A. Moro, A. Raganato, and R. Navigli.
“Entity linking meets word sense disambiguation: a unified approach.”
In: *Transactions of the Association for Computational Linguistics* (2014) (cit. on p. 42).
- [167] I. O. Mulang, K. Singh, and F. Orlandi. “Matching Natural Language Relations to Knowledge Graph Properties for Question Answering.”
In: *Proceedings of the 13th International Conference on Semantic Systems*. ACM. 2017, pp. 89–96 (cit. on p. 42).
- [168] K. Singh et al. “Capturing Knowledge in Semantically-typed Relational Patterns to Enhance Relation Linking.” In: *Proceedings of the Knowledge Capture Conference*. ACM. 2017, p. 31 (cit. on p. 42).
- [169] 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* (2018).
DOI: [10.18653/v1/k18-1050](https://doi.org/10.18653/v1/k18-1050).
URL: <http://dx.doi.org/10.18653/v1/k18-1050> (cit. on p. 43).
- [170] D. Sorokin and I. Gurevych. “Mixing Context Granularities for Improved Entity Linking on Question Answering Data across Entity Categories.”
In: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*.
New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 65–75.
DOI: [10.18653/v1/S18-2007](https://doi.org/10.18653/v1/S18-2007).
URL: <https://www.aclweb.org/anthology/S18-2007>
(cit. on pp. 43, 44, 47, 49, 59).
- [171] D. Banerjee et al.
“PNEL: Pointer Network based End-To-End Entity Linking over Knowledge Graphs.”
In: *International Semantic Web Conference*. Springer. 2020, pp. 21–38 (cit. on pp. 44, 59).
- [172] B. Santorini. “Part-of-speech tagging guidelines for the Penn Treebank Project.” In: (1990)
(cit. on p. 46).
- [173] A. Graves, S. Fernández, and J. Schmidhuber.
“Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition.”
In: *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*. Springer-Verlag, 2005 (cit. on p. 47).
- [174] W.-t. Yih et al.
“The value of semantic parse labeling for knowledge base question answering.”
In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2016, pp. 201–206 (cit. on pp. 47, 58).
- [175] J. Berant et al. “Semantic Parsing on Freebase from Question-Answer Pairs.” In: *EMNLP*.
Vol. 2. 2013, p. 6 (cit. on p. 47).

- [176] M. Dubey et al. “LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia.” In: *The Semantic Web – ISWC 2019*. Ed. by C. Ghidini et al. Cham: Springer International Publishing, 2019, pp. 69–78. ISBN: 978-3-030-30796-7 (cit. on p. 47).
- [177] A. Sakor et al. “Old is Gold: Linguistic Driven Approach for Entity and Relation Linking of Short Text.” 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, June 2019, pp. 2336–2346. DOI: [10.18653/v1/N19-1243](https://doi.org/10.18653/v1/N19-1243). URL: <https://www.aclweb.org/anthology/N19-1243> (cit. on p. 48).
- [178] M. Dubey et al. “EARL: Joint Entity and Relation Linking for Question Answering over Knowledge Graphs.” In: *The Semantic Web – ISWC 2018*. Ed. by D. Vrandečić et al. Cham: Springer International Publishing, 2018, pp. 108–126. ISBN: 978-3-030-00671-6 (cit. on p. 48).
- [179] P. Ferragina and U. Scaiella. “Tagme: on-the-fly annotation of short text fragments (by wikipedia entities).” In: *Proceedings of the 19th ACM international conference on Information and knowledge management*. 2010, pp. 1625–1628 (cit. on p. 48).
- [180] A. Delpuch. “Opentapioca: Lightweight entity linking for wikidata.” In: *arXiv preprint arXiv:1904.09131* (2019) (cit. on pp. 48–50, 59).
- [181] D. B. Nguyen et al. “Query-driven on-the-fly knowledge base construction.” In: *Proceedings of the VLDB Endowment* 11.1 (2017), pp. 66–79 (cit. on p. 48).
- [182] Y. Yang and M.-W. Chang. “S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking.” In: *arXiv preprint arXiv:1609.08075* (2016) (cit. on p. 49).
- [183] A. Sakor et al. *FALCON 2.0: An Entity and Relation Linking Tool over Wikidata*. 2019. arXiv: [1912.11270](https://arxiv.org/abs/1912.11270) [cs.CL] (cit. on p. 49).
- [184] P. Wu et al. “Learning Representation Mapping for Relation Detection in Knowledge Base Question Answering.” In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 6130–6139. DOI: [10.18653/v1/P19-1616](https://doi.org/10.18653/v1/P19-1616) (cit. on pp. 52, 60, 106, 107, 113).
- [185] L. Logeswaran et al. “Zero-Shot Entity Linking by Reading Entity Descriptions.” In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 3449–3460 (cit. on p. 52).
- [186] M. R. A. H. Rony et al. “End-to-End Entity Linking and Disambiguation leveraging Word and Knowledge Graph Embeddings.” In: (2020) (cit. on p. 52).

-
- [187] A. Bordes et al. “Translating Embeddings for Modeling Multi-relational Data.” In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., 2013, pp. 2787–2795.
URL: <http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf> (cit. on pp. 53, 57).
- [188] P. Zhou et al. “Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification.” In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 207–212.
DOI: 10.18653/v1/P16-2034.
URL: <https://www.aclweb.org/anthology/P16-2034> (cit. on p. 55).
- [189] E. Jang, S. Gu, and B. Poole. “Categorical reparameterization with gumbel-softmax.” In: *arXiv preprint arXiv:1611.01144* (2016) (cit. on p. 55).
- [190] L. Duong et al. “Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser.” In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 845–850.
DOI: 10.3115/v1/P15-2139.
URL: <https://www.aclweb.org/anthology/P15-2139> (cit. on p. 57).
- [191] D. Sorokin and I. Gurevych. “Modeling semantics with gated graph neural networks for knowledge base question answering.” In: *arXiv preprint arXiv:1808.04126* (2018) (cit. on pp. 58, 59).
- [192] D. Lukovnikov, A. Fischer, and J. Lehmann. “Pretrained transformers for simple question answering over knowledge graphs.” In: *International Semantic Web Conference*. Springer, 2019, pp. 470–486 (cit. on p. 58).
- [193] Y. J. Kim and H. H. Awadalla. “FastFormers: Highly Efficient Transformer Models for Natural Language Understanding.” In: *arXiv preprint arXiv:2010.13382* (2020) (cit. on p. 58).
- [194] A. Sakor et al. “Falcon 2.0: An Entity and Relation Linking Tool over Wikidata.” In: *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. CIKM ’20. Virtual Event, Ireland: Association for Computing Machinery, 2020, pp. 3141–3148. ISBN: 9781450368599. DOI: 10.1145/3340531.3412777 (cit. on p. 59).
- [195] P. Ferragina and U. Scaiella. “Fast and accurate annotation of short texts with wikipedia pages.” In: *IEEE software* 29.1 (2011), pp. 70–75 (cit. on p. 59).
- [196] D. Chaudhuri et al. “Improving Response Selection in Multi-Turn Dialogue Systems by Incorporating Domain Knowledge.” In: *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 497–507 (cit. on pp. 63, 87).

- [197] K. Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation.” English (US).
In: *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. 2014 (cit. on p. 65).
- [198] D. Bahdanau et al. “Learning to Compute Word Embeddings On the Fly.” In: (2018) (cit. on pp. 68, 71).
- [199] D. C. Uthus and D. W. Aha. “Extending Word Highlighting in Multiparticipant Chat.” In: *FLAIRS Conference*. 2013 (cit. on p. 69).
- [200] Y. Wu et al. “Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots.” In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2017, pp. 496–505 (cit. on pp. 69–71).
- [201] P. Bojanowski et al. “Enriching Word Vectors with Subword Information.”
In: *TACL 5* (2017), pp. 135–146.
URL: <https://transacl.org/ojs/index.php/tacl/article/view/999>
(cit. on pp. 70, 71, 94).
- [202] A. Paszke et al. “Automatic differentiation in PyTorch.” In: (2017) (cit. on p. 70).
- [203] Y. Xu and D. Reitter.
“Information density converges in dialogue: Towards an information-theoretic model.”
In: *Cognition* 170 (2018), pp. 147–163 (cit. on p. 72).
- [204] A. Mohammed et al.
“Jointly learning word embeddings using a corpus and a knowledge base.”
In: *PLoS ONE* 13(3): e0193094. (2018) (cit. on pp. 76, 78).
- [205] A. Graves. “Generating Sequences with Recurrent Neural Networks.” In: (2015) (cit. on p. 77).
- [206] J. Pennington, R. Socher, and C. D. Manning.
“Glove: Global Vectors for Word Representation.” In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2014, pp. 1532–1543. URL: <http://aclweb.org/anthology/D/D14/D14-1162.pdf>
(cit. on pp. 78, 83).
- [207] K. Papineni et al. “BLEU: a method for automatic evaluation of machine translation.”
In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2002, pp. 311–318 (cit. on pp. 82, 95, 105).
- [208] C.-W. Liu et al. “How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation.”
In: *arXiv preprint arXiv:1603.08023* (2016) (cit. on p. 82).
- [209] W. Zhu et al. “Flexible end-to-end dialogue system for knowledge grounded conversation.”
In: *arXiv preprint arXiv:1709.04264* (2017) (cit. on p. 87).

-
- [210] P. Agrawal, A. Suri, and T. Menon. “A Trustworthy, Responsible and Interpretable System to Handle Chit-Chat in Conversational Bots.”
In: *The Second AAAI Workshop on Reasoning and Learning for Human-Machine Dialogues*. Nov. 2018 (cit. on p. 87).
- [211] S. Akasaki and N. Kaji. “Chat Detection in an Intelligent Assistant: Combining Task-oriented and Non-task-oriented Spoken Dialogue Systems.” In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1308–1319. doi: [10.18653/v1/P17-1120](https://doi.org/10.18653/v1/P17-1120) (cit. on p. 87).
- [212] J. Dodge et al. “Evaluating prerequisite qualities for learning end-to-end dialog systems.” In: *ICLR* (2016) (cit. on p. 88).
- [213] M. Buhrmester, T. Kwang, and S. D. Gosling.
“Amazon’s Mechanical Turk: A New Source of Inexpensive, Yet High-Quality, Data?”
In: *Perspectives on Psychological Science* 6.1 (2011). PMID: 26162106, pp. 3–5.
doi: [10.1177/1745691610393980](https://doi.org/10.1177/1745691610393980).
eprint: <https://doi.org/10.1177/1745691610393980>.
URL: <https://doi.org/10.1177/1745691610393980> (cit. on p. 88).
- [214] V. Rieser and O. Lemon. “Learning effective multimodal dialogue strategies from Wizard-of-Oz data: Bootstrapping and evaluation.”
In: *Proceedings of ACL-08: HLT* (2008), pp. 638–646 (cit. on p. 88).
- [215] T. Bergmann et al. In: *I-SEMANTICS*. Ed. by M. Sabou et al. ACM, 2013, pp. 146–149 (cit. on p. 90).
- [216] J. Gu et al. “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, 2016, pp. 1631–1640. doi: [10.18653/v1/P16-1154](https://doi.org/10.18653/v1/P16-1154) (cit. on pp. 91, 92).
- [217] M.-T. Luong, H. Pham, and C. D. Manning.
“Effective approaches to attention-based neural machine translation.”
In: *arXiv preprint arXiv:1508.04025* (2015) (cit. on pp. 92, 103).
- [218] S. Merity et al. “Pointer sentinel mixture models.”
In: *arXiv preprint arXiv:1609.07843* (2016) (cit. on p. 92).
- [219] R. J. Williams and D. Zipser.
“A learning algorithm for continually running fully recurrent neural networks.”
In: *Neural computation* 1.2 (1989), pp. 270–280 (cit. on p. 94).
- [220] N. Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting.”
In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.
URL: <http://dl.acm.org/citation.cfm?id=2670313> (cit. on p. 94).
- [221] D. Chaudhuri, M. R. A. H. Rony, and J. Lehmann. “Grounding Dialogue Systems via Knowledge Graph Aware Decoding with Pre-trained Transformers.” In: (2020) (cit. on p. 99).
- [222] H. Rosales-Méndez, B. Poblete, and A. Hogan. “What should entity linking link?” In: *AMW*. 2018 (cit. on p. 99).

- [223] C. Tillmann and H. Ney. “Word reordering and a dynamic programming beam search algorithm for statistical machine translation.”
In: *Computational linguistics* 29.1 (2003), pp. 97–133 (cit. on p. 103).
- [224] D. Diefenbach et al.
“Core techniques of question answering systems over knowledge bases: a survey.”
In: *Knowledge and Information systems* 55.3 (2018), pp. 529–569 (cit. on p. 105).
- [225] S. Banerjee and A. Lavie. “METEOR: An automatic metric for MT evaluation with improved correlation with human judgments.” In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 2005, pp. 65–72 (cit. on p. 105).
- [226] S. Sharma et al. “Relevance of Unsupervised Metrics in Task-Oriented Dialogue for Evaluating Natural Language Generation.” In: *arXiv preprint arXiv:1706.09799* (June 2017).
URL: <https://www.microsoft.com/en-us/research/publication/relevance-unsupervised-metrics-task-oriented-dialogue-evaluating-natural-language-generation/>
(cit. on p. 105).
- [227] *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017. ISBN: 978-1-5386-0457-1.
URL: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8097368>.

List of Publications

10.1 Accepted Papers

- Mohnish Dubey, Debayan Banerjee, **Debanjan Chaudhuri**, and Jens Lehmann, EARL: joint entity and relation linking for question answering over knowledge graphs, International Semantic Web Conference(ISWC), 2018.
- Debayan Banerjee, **Debanjan Chaudhuri**, Mohnish Dubey, and Jens Lehmann, PNEL: Pointer Network based End-To-End Entity Linking over Knowledge Graphs, International Semantic Web Conference(ISWC), 2020.
- **Debanjan Chaudhuri**, Augustinus Kristiadi, Jens Lehmann, and Asja Fischer, Conference on Natural Language Learning (CoNLL), 2018.
- Firas Kassawat, **Debanjan Chaudhuri**, and Jens Lehmann, Extended Semantic Web Conference (ESWC), 2019.
- **Debanjan Chaudhuri**, MRAH Rony, Simon Jordan, and Jens Lehmann, Using a KG-Copy Network for Non-Goal Oriented Dialogues, International Semantic Web Conference (ISWC), 2019.
- **Debanjan Chaudhuri**, Md Rashad Al Hasan Rony, and Jens Lehmann, Grounding Dialogue Systems via Knowledge Graph Aware Decoding with Pre-trained Transformers, 2021.

10.2 Papers in Review

- Md Rashad Al Hasan Rony, **Debanjan Chaudhuri**, Rostislav Nedelchev, Asja Fischer, and Jens Lehmann, End-to-End Entity Linking and Disambiguation leveraging Word and Knowledge Graph Embeddings, 2021.

List of Figures

1.1	Example where a Conversation System can benefit from several knowledge sources in order to have knowledge-grounded conversation with the user.	2
1.2	Hierarchical Structure of General Intelligence as defined by the CHC model. The last strata representing general intelligence as defined by [12]. Diagram adapted from [17]	3
1.3	The sub-research questions contributing to the main Research Question.	5
2.1	Knowledge Base and corresponding Knowledge Graph	11
4.1	EARL Architecture: In the disambiguation phase one may choose either Connection Density or GTSP. In cases where training data is not available beforehand GTSP works better.	34
4.2	Using GTSP for disambiguation : The bold line represents the solution offered by the GTSP solver. Each edge represents an existing connection in the Knowledge Graph. The edge weight is equal to the number of hops between the two nodes in the Knowledge Graph. We also add the index search ranks of the two nodes the edges connect to the edge weight when solving for GTSP.	36
4.3	Connection Density with example: The dotted lines represent corresponding connections between the nodes in the knowledge base.	37
4.4	Adaptive E/R learning	40
4.5	The red and green dots represent entity candidate vectors for the given question. The green vectors are the correct entity vectors. Although they belong to the same entity they are not the same dots because they come from different n-grams. At each time step the Pointer Network points to one of the input candidate entities as the linked entity, or to the END symbol to indicate no choice.	44

4.6	The word "Francisco" is vectorised in the following manner: 4 n-grams represented by the underlines are considered and searched against an entity label database. The top 50 search results are depicted for each of the n-grams resulting in 200 candidates. For the entity Q72495, for example, we fetch its TransE embedding, add its text search rank, n-gram length, word index position as features. Additionally we also append the fastText embedding for "Francisco" and the entire fastText embedding for the sentence (average of word vectors) to the feature. We then append the fastText embeddings for the label and description for this entity. Hence we get a 1142 dimensional vector V_{k120} corresponding to entity candidate Q72495. For all 200 candidate entities for "Francisco", we have a sequence of two hundred 1142 dimensional vectors as input to the pointer network. For the sentence above which has 7 words, this results in a final sequence of $7 \times 200 = 1400$ vectors each of length 1142 as input to our pointer network. Any one or more of these vectors could be the correct entities.	45
4.7	K=Number of search candidates per n-gram. On the left: K vs F1 score on a set of 100 WebQSP test questions, with average word length of 6.68. F1 is maximum for K=40 and 50. On the right: K vs time taken for PNEL to return a response. The relationship appears to be close to linear.	47
4.8	Architecture of ELiDi.	54
5.1	Our proposed way to incorporate domain knowledge into the model. β_t and $1 - \beta_t$ represent the (multiplicative) weights for the description embedding and the word embedding respectively. The resulting combination, \hat{w}_t^r acts as an input of the encoder.	67
6.1	Sequence-to-Sequence based module with multi-task learning of user intent regularized with additional Entity Loss	79
6.2	An example of how KVL works to replace predicted entities with correct ones from the local KG.	81
7.1	A conversation about the football club Arsenal and the Knowledge Graph involved.	89
7.2	AMT setup for getting conversations over soccer.	90
7.3	Schema of the proposed Knowledge Graph for Arsenal.	91
7.4	KG-Copy Model Encoder-Decoder Architecture for Knowledge Grounded Response Generation.	94
7.5	Response Generation during Decoding for KG-Copy Model.	96
8.1	Example of a knowledge grounded conversation.	100
8.2	KGIRNet model diagram.	101
8.3	Sub-Graph Encoding using Graph Laplacian.	104
9.1	A Figure depicting the overall contribution of this Thesis towards Answering the main Research Question(RQ).	112

List of Tables

4.1	Comparison of GTSP based approach and Connection density for Disambiguation	37
4.2	Empirical comparison of Connection Density and GTSP: n = number of nodes in graph; L = number of clusters in graph; N = number of nodes per cluster; top K results retrieved from ElasticSearch.	41
4.3	Evaluation of joint linking performance	41
4.4	Evaluating EARL's Entity Linking performance	42
4.5	Evaluating EARL's Relation Linking performance	42
4.6	Evaluation on KBPearl split of LC-QuAD 2.0 test set	48
4.7	Evaluation on LC-QuAD 2.0 test set	49
4.8	Evaluation on WebQSP	49
4.9	Evaluation on SimpleQuestions	50
4.10	Entity Candidates available post label search	50
4.11	Ablation test for PNEL on WebQSP test set features	51
4.12	Time taken per question on the WebQSP dataset of 1639 questions	51
4.13	Comparison of PNEL's performance with respect to number of entities in a question.	52
4.14	Entity Linking Accuracy over Freebase Knowledge Graph	58
4.15	Entity Linking on SimpleQuestions over Wikidata Knowledge Graph	59
4.16	Entity Linking on WebQSP over Wikidata Knowledge Graph	59
4.17	Ablation Study	60
4.18	Span Detection Error (Green - correct span, blue - detected span).	60
5.1	Illustration of a multi-turn conversation with domain specific words (UNIX commands) in italics.	64
5.2	Evaluation results of our models compared to various baselines on Ubuntu Dialogue Corpus.	69
5.3	Ablation study with different settings.	71
5.4	Visualization of attention weight in utterance samples, darker shade means higher attention weight.	72
5.5	Sample context utterances from UDC's test set whose correct response is the first utterance in Table 5.4.	72
5.6	Examples on the error our model made. We observed that our model's predictions are biased towards non-generic responses.	73
6.1	User query and respective intents	76
6.2	Statistics of the in-car multi-turn, multi-domain, goal-oriented dialogue dataset.	81
6.3	Statistics of the trained Joint Embedding.	82

6.4	Results compared to baseline and State-of-the-art models.	83
6.5	Ablation Study.	83
6.6	Example of generated responses on the Navigation domain explaining the improvement process through the model stages.	84
6.7	KG triples for the dialogue in Table 6.6, for navigation.	84
6.8	Error Analysis.	85
6.9	KG triples and context for the error analysis in Table 6.8 for weather.	85
7.1	Statistics of Soccer Dataset.	89
7.2	KG statistics.	91
7.3	Results on Soccer Dataset.	95
7.4	Results on the In-car Dialogue Dataset.	95
7.5	Human evaluation of generated responses.	97
7.6	KG-copy Response for Factoid and non-Factoid queries.	97
7.7	KG Copy Model’s Knowledge Grounded Responses.	97
7.8	Incorrect opinionated responses from KG-Copy model.	98
7.9	Incorrect factual responses from KG Copy model.	98
8.1	Dataset statistics.	105
8.2	Evaluation on goal and non-goal oriented dialogues.	106
8.3	Relation Linking Accuracy on SQB [184] Dataset.	106
8.4	Analyzing sample predictions.	107
8.5	In-depth evaluation of KGIRNet model.	108
8.7	Analysing fact-fullness of KGIRNet.	109
8.8	A conversation with KGIRNet.	110
8.9	Error analysis of KGIRNet.	110