ulm university universität
uulm

Ulm University
Faculty of Engineering Sciences, Computer Science and Psychology

Institute of Communications Engineering

Combining Neural Networks with Knowledge for Spoken Dialogue Systems

A doctoral thesis jointly supervised with Southeast University (China) and submitted in fulfilment of the requirements for the academic degree of

Doctor of Natural Sciences
Dr. rer. nat.

By

Waheed Ahmed

born in Sukkur(Pakistan)

2022

Acting Dean:	Prof. Dr. Anke Huckauf
Supervisors:	Prof. Dr. Dr.-Ing. Wolfgang Minker Prof. Guilin Qi
Examiners:	Prof. Dr.-Ing. Stefan Wesner Prof. Xinyu Dai Prof. Zhiqiang Gao Prof. Deyu Zhou Prof. Weiwei Wu
Examination date	June 9, 2022

心於至善

布罗

学校代码: 10286
分 类 号: 000
密 级: 公开
U D C: 000
学 号: 169938

博士学位论文

将神经网络与知识相结合，用于口语对话系统

研究生姓名: 布罗
导 师 姓 名: 漆桂林 教授

申请学位类别 计算机科学博士 学位授予单位 东 南 大 学
一级学科名称 计算机科学与技术 论文答辩日期 2022 年 6 月 9 日
二级学科名称 计算机应用技术 学位授予日期
答辩委员会主席 戴新宇教授 评 阅 人 盲审

2022-07-01

Combining Neural Networks with Knowledge for Spoken Dialogue Systems

A Dissertation Submitted to
Southeast University
For the Academic Degree of Doctor of Engineering

By

Waheed Ahmed

Supervised By

Prof. Guilin Qi
Prof. Wolfgang Minker

School of Computer Science and Engineering
Southeast University
June 2022

Parts of this dissertation have already been published in the following articles:

[1] Abro, Waheed Ahmed, Annalena Aicher, Niklas Rach, Stefan Ultes, Wolfgang Minker, and Guilin Qi. ”Natural Language Understanding for Argumentative Dialogue Systems in the Opinion Building Domain.” Knowledge-Based Systems 242 (2022): 108318.

[2] Abro, Waheed Ahmed, Guilin Qi, Muhammad Aamir, and Zafar Ali. ”Joint Intent Detection and Slot Filling using Weighted Finite State Transducer and BERT.” Applied Intelligence (2022): 1-15.

[3] Abro, Waheed Ahmed, Guilin Qi, Zafar Ali, Yansong Feng, and Muhammad Aamir. ”Multi-turn intent determination and slot filling with neural networks and regular expressions.” Knowledge-Based Systems 208 (2020): 106428.

[4] Abro, Waheed Ahmed, Guilin Qi, Huan Gao, Muhammad Asif Khan, and Zafar Ali. ”Multi-turn intent determination for goal-oriented dialogue systems.” In 2019 international joint conference on neural networks (IJCNN), pp. 1-8. IEEE, 2019.

东南大学学位论文独创性声明

.

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

研究生签名：_____日期：2022-06-24

东南大学学位论文使用授权声明

东南大学、中国科学技术信息研究所、国家图书馆、《中国学术期刊（光盘版）》电子杂志社有限公司、万方数据电子出版社、北京万方数据股份有限公司有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括以电子信息形式刊登）论文的全部内容或中、英文摘要等部分内容。论文的公布（包括以电子信息形式刊登）授权东南大学研究生院办理。.

研究生签名：_____导师签名：  _____日期：2022-06-24

摘要

自然语言理解、口语对话系统、深度神经网络、正则表达式、人机交互

口语对话系统旨在与用户交流并帮助他们使用自然语言。例如，面向任务的对话系统旨在帮助用户完成特定任务，例如预订航班、导航到特定目的地或寻找餐厅。通常，传统的对话系统是高度手工制作的，具有复杂的逻辑和少量的规则。这些系统由四个部分组成：自然语言理解 (NLU)、对话状态跟踪 (DST)、对话管理器 (DM) 和自然语言生成 (NLG)。尽管自然语言理解和对话学习取得了进步，但这些系统在新领域的鲁棒性和可扩展性方面仍存在着问题。NLU 模块是使用领域特定的规则构建的，因此很难扩展到新的域。由此，诸如深度神经网络的统计模型已经被提出并用于 NLU 任务。然而，这些深度神经网络模型依赖于大量的标记数据，而我们通常只能够得到有限的标记数据。在小型训练数据中，它们无法对测试数据进行泛化，因为它们容易捕获虚假特征，而不是获取到重要的领域语义特征。此外，在许多应用中，获取高质量的标记数据是一项成本高昂且耗时的操作。

在本论文中，我们研究了将神经网络与符号规则知识相结合的不同方法，以此来解决口语对话系统中现有 NLU 模块的局限性，以减少对标记训练数据的需求，并提高对新领域的泛化和可扩展性。本论文的贡献如下：

1. 本文的第一个贡献，是提出了一个自然语言理解 (NLU) 框架，用于信息寻求和意见构建领域的论证对话系统。我们采用预训练的语言模型，即来自 Transformers 的双向编码器表示 (BERT)，用于具有少量域内训练数据的论证对话领域。我们所提出的模型，针对两个 NLU 任务对 BERT 进行了微调，即：意图分类和论证相似性任务。意图分类识别一个论证过程中的意图或用户移动过程，论证相似性任务用于检查用户话语与所呈现论证的关系。实验结果表明，我们提出的方法在不同数据集上的意图分类和论证相似性任务，对比基线模型，具有明显的优势。此外，结果表明该模型对于来自训练期间未见过的主题和不同语言能力的数据具有高而稳定的性能。

2. 本文第二个贡献通过加权有限状态传感器 (WFST) 增强了对类 BERT 架构的微调，以减少对大量监督数据的需求。WFST-BERT 模型利用预训练的 BERT 架构来生成用户句子的上下文表示，并通过将正则表达式 (RE) 规则转换为可训练的加权有限状态转换器来利用它们。然后，BERT 表示与 WFST 被结合起来，并使用梯度下降算法在监督数据上对它们同时进行训练。实验结果表明，当可用的训练样本有限或没有可用的训练样本时，WFST-BERT 可以产生不错的预测。

3. 本文第三个贡献介绍了一种基于神经网络的多任务学习模型，该模型使用上下文信息进行多轮意图检测和槽填充任务。我们采用记忆网络对用户对话中的多轮信息进行建模和优化。该模型从预训练的语言模型中提取上下文词特征，并分别采用 CNN 和 RNN 结构来分别预测用户意图和标记相应的槽。此外，该模型集成了正则表达式以编码领域知识，并且以端到端可训练的方式调节神经网络输出。

我们在公开可用的单轮数据集（如 ATIS、SNIPS、Banking）和多轮数据集（如 Key-Value Retrieval、Frames）上评估了我们提出的方法。实验结果表明，本文所提出的模型在有限训练数据的场景、以及完整训练数据的场景中，均优于基线方法。

关键词：

Abstract

Spoken dialogue systems are designed to communicate with users and help them using natural language. Task-oriented dialogue systems, for example, are designed to help users achieve specific tasks, such as booking a flight, navigating to a particular destination, or finding a restaurant. Typically, conventional dialogue systems are highly handcrafted, with complex logic and few rules. These systems are composed of four components: natural language understanding (NLU), dialogue state tracking (DST), dialogue manager (DM), and natural language generation (NLG). Despite advancements in natural language understanding and dialogue learning, these systems continue to confront significant problems in terms of robustness and scalability to new domains. The NLU module is built using domain-specific rules, making it difficult to expand to new domains. As a result, statistical models such as deep neural networks have been proposed for NLU tasks. However, these deep neural networks models rely on large amounts of labeled data, whereas we often have limited labeled data. In small training data, they fail to generalize over test data as they are prone to capture fake features rather than semantically significant domain features. Additionally, in many applications, obtaining high-quality labelled data is a highly costly and time-consuming operation.

In this thesis, we address the limitations of the existing NLU module of the spoken dialogue system by examining various methods for combining neural networks with symbolic rules knowledge to reduce the need for labeled training data, and for improving generalization and scalability to new domains. Contributions of this dissertation are following:

1. Our first contribution introduces a natural language understanding (NLU) framework for argumentative dialogue systems in the information-seeking and opinion building domain. We employ a pre-trained language model, namely Bidirectional Encoder Representations from Transformers (BERT) for argumentation domains having small in-domain training data. The proposed model fine-tuned BERT for two NLU tasks, namely: intent classification and argument similarity tasks. Intent classification identifies intent or user move in an argument and argument similarity task check the relation of user utterance with the presented arguments. The experimental results show a clear advantage of our proposed approach over the baselines for the intent classification and argument similarity tasks on different datasets. Moreover, the outcomes indicate a high and stable performance of the model for data from topics unseen during training and different language proficiency.

2. The second contribution augments the fine-tuning of BERT-like architecture with weighted finite-state transducer (WFST) to reduce the need for massive supervised data. The WFST-BERT model utilizes pre-trained BERT architecture to generate contextual representations of user sentences and leverage regular expressions (REs) rules by converting them into the trainable weighted finite-state transducer. BERT representation is then combined with WFST and

trained simultaneously on supervised data using a gradient descent algorithm. The experimental results show that WFST-BERT can generate decent predictions when limited or no training examples are available.

3. The third contribution introduces a multi-task learning model based on neural networks with contextual information for multi-turn intent detection and slot filling tasks. We employ the memory network to model and optimize the multi-turn information in user conversation. The model extracts the contextual word features from the pre-trained language model and employs CNN and RNN structures for predicting user intent and tagging corresponding slots respectively. Furthermore, the model integrates regular expressions (REs) to encode domain knowledge and regulate neural network output in an end-to-end trainable manner.

We evaluated our proposed methods on publicly available single-turn datasets such as ATIS, SNIPS, Banking and multi-turn datasets namely Key-Value Retrieval and Frames. The empirical results demonstrate that the proposed models outperform baseline methods in both limited data settings and full data settings.

Keywords: Natural Language Understanding, Spoken Dialogue System, Deep Neural Networks, Regular Expressions, Human−Computer Interaction

Table of Contents

List of Figures

List of Tables

Acknowledgment

All glory be to Almighty Allah, the Most Merciful, Who rewards us with His love, mercy, and assistance, which enabled me to complete my studies. I want to express my heartfelt appreciation to Professor Guilin Qi, my advisor. Being his PhD student has been a privilege. Prof. Guilin Qi mentored me tremendously both academically and in life. Mentally and academically, he has been tremendously helpful, never shying away from critical criticism or patting my back when good results are achieved. I am grateful for his time, thoughts, and support in helping to make my PhD experience productive. I consider myself immensely fortunate to have him as my advisor.

I also want to thank my co-advisor, Professor Wolfgang Minker, who has always been a supporting, encouraging, and caring mentor. His enthusiasm, encouragement, and faith in me throughout have been extremely helpful. He was always accessible to answer my queries, upbeat and generous with his time and knowledge. He was constantly aware of where to go for solutions to problems while guiding me to the appropriate source, theory, and perspective.

I want to express my gratitude to the Chinese Government, the China Scholarship Council in Beijing, and Southeast University in China for providing the necessary learning environment and financial support to accomplish my aim. A special thanks to Southeast University's Knowledge Engineering Lab for their unwavering support and affection during our journey. I would say it produced meaningful results. Throughout my pursuit of this degree, I had the opportunity to work with some of the brightest and most sincere brains and world-class scholars. Additionally, I am grateful to the DAAD-STIBET for providing a research grant and an appropriate work environment throughout my time at ULM University in Germany.

To my proofreaders, Prof. Guilin Qi, Dr Muhammad Aamir, and Dr Zafar Ali, I would like to express my gratitude for tolerating my jargon and going over each line. Your comments have improved the readability of this thesis.

I want to express my heartfelt gratitude to my dear friends Muhammad Aamir, Zafar Ali, Muhammad Musa, Asif Dadra, and my beloved wife for their unending support and encouragement. I'd want to express my gratitude to all of my lab mates, particularly Hua Yuncheng, Bi Sheng, Wu Tongtong, Hu Peng, and Gao Huan, for looking after me during my stay in China. Additionally, I would like to thank Niklas Rach, Annalenna Aicher, Matthias Kraus, Denis Dresvyanskiy, and a large number of others for the fantastic interchange of research experiences and ideas that occurred during my time at ULM in Germany.

Finally, and maybe most importantly, I want to express my gratitude to my parents, wife, and siblings for their reassuring and unwavering love. I owe you a debt far greater than words or scripts can convey. This thesis is dedicated to them.

第一章 Introduction

Spoken dialogue systems also called conversational agents or virtual personal assistants are becoming more prevalent in our personal and professional lives. These systems communicate with users in natural language to assist them in solving particular tasks or to entertain them. The two basic categories of dialogue systems are chit-chat systems and goal-oriented dialogue systems. The chit-chat systems [5-10] communicate with users on a wide range of topics. On these systems, users don't have any specific goal to accomplish, and the main purpose of the conversation is to have casual chat, provide mental support, and entertain the users. The system first requires to understand the user utterance correctly, and then respond to the user with meaningful responses. On the other hand, goal-oriented dialogue systems [11-15] assist users to complete specific goals like send an email, find a restaurant or navigate to a particular location. Goal-oriented dialogue systems process the user request, identify the user need, ask for any required detail, and respond to the user with related information. These systems usually interact with external knowledge sources and take multiple dialogue turns to facilitate user needs.

Chit-chat dialogue systems are divided into three subcategories: rule-based systems, IR-based systems, and generation-based systems. Chit-chat dialogue systems are divided into three subcategories: rule-based systems, information retrieval systems, and generation-based systems. The rule-based systems utilize pre-defined rules such as if-else conditions or statistical classifiers to evaluate user input. After identifying the particular rule for user input, relevant action is performed. The most famous rule-based system in the history are ELIZA[1] [16], PARRY[2] [17], Cleverbot[3] , Alice[4]. The ELIZA system searches for keywords in the user utterance. If the keyword is found the particular rule is applied to transform the user utterance into appropriate response. If the keyword is not found, the system responds to the user with a generic response. The IR-based systems exploit information retrieval [18-21] techniques. Given a user utterance, the IR-based system selects response to the user from training corpus based on two criteria: The dialogue history of current input is similar to the dialogue history of candidate response and candidate response is semantically similar to the current input dialogue history. Various ranking schemes such as TF-IDF, page-rank or personalization techniques can be ensembled to form a ranking function and select the highest ranked response for responding users. The Generation-based systems [22, 23] generate response token by token instead of selecting whole response from training corpus. These systems leverage statistical sequence-to-sequence models for producing coherent and meaningful output sequences for given input utterance.

On the other hand, Goal-oriented dialogue systems are usually composed of a chain of five mod-

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

[2] https://en.wikipedia.org/wiki/PARRY

[3] https://en.wikipedia.org/wiki/Cleverbot

[4] https://en.wikipedia.org/wiki/Artificial_Linguistic_Internet_Computer_Entity

ules: *speech recognition, natural language understanding, dialogue manager, natural language generation* and *speech synthesis*. The user utterances are passed through automatic speech recognition (ASR) which converts speech into a text format. Next, the natural language understanding (NLU) module processes text from the lexical and syntactic levels, an intent recognition takes place, and potentially additional information is analyzed. The output of the NLU component is passed to the dialogue manager which tracks the dialogue state, retrieves information from the external database if necessary, and generates a dialogue act. The dialogue act is utilized by the natural language generation module to produce a response to the user. In this thesis, we will focus on the natural language understanding module.

1.1 Background and Motivation

In recent years, significant progress has been made on natural language understanding tasks such as intent detection, slot filling, and sentence similarity utilizing deep neural networks that learn the robust hidden representation from the training data. However, these models' predicted performance is highly dependent on a large amount of labeled data. They fail to generalize over test data when the training data set is limited, as they are prone to capture fictitious features rather than semantically meaningful domain information. Furthermore, these models operate on the vector representation of words as they replace words with their vector embedding. Therefore, it is difficult to interpret prediction results generated by these models and it is also hard to incorporate domain knowledge for guiding model output. Moreover, in many applications collecting high-quality labeled data is a very expensive and time taking process, which hinders applying recent neural network methods.

In comparison, rule-based models built using hand-crafted rules such as regular expressions do not require labeled training data and frequently achieve a reasonable level of prediction accuracy. Regular expressions (REs) are widely utilized in a variety of applications, including pattern matching [24], entity recognition [25], and information extraction [26]. These models can be used in situations where there are few or no training samples available. However, RE-based systems do not benefit from labeled data when available and generalize poorly on a large dataset containing a lot of synonyms and variations.

To summarize, rule-based models are simple to comprehend; they allow for the addition of new rules or the modification of current ones. They do not require labeled training data and provide a mechanism to encode domain knowledge. On the other hand, neural networks require a large amount of training data and are notoriously difficult to interpret. Additionally, it is challenging to infuse domain knowledge into models to lead them toward capturing desired patterns. Therefore, we want to combine the benefits of both neural networks and rule-based models.

1.2 Problem Definition and Statement

In this thesis, we are investigating the combination of neural networks with external knowledge for the natural language understanding module of goal-oriented dialogue systems. The natural language understanding module first recognizes the intent of the user from the user utterance, the task often known as intent detection. The further tasks of the NLU module extract useful additional semantic information from the user's utterance such as tagging slots corresponding to a semantic frame (slot filling), identifying which sentence the user refers to (sentence similarity).

1.2.1 Intent Detection

Intent detection is often treated as a sentence classification task where we learn the mapping of a sequence of words $w = (w_1, w_2, \cdots, w_n)$ to a corresponding intent label c from the pre-defined set of intent labels, formulated as follows:

$$\hat{c} = arg\max_{c} P(c|w) \tag{1.1}$$

1.2.2 Slot Filling

Slot filling is considered as a sequence labeling task where we learn the mapping of a sequence of words $w = (w_1, w_2, \cdots, w_n)$ to a corresponding output sequence $y = (y_1, y_2, \ldots, y_n)$, formulated as follows:

$$\hat{y} = arg\max_{y} p(y|w) \tag{1.2}$$

Where y_i is the slot label of the word w_i.

1.2.3 Sentence Similarity

For the sentence similarity task, we find the most similar sentence $v \in V$ from the set of given sentences V for given an input sentence u. The input sentences are converted into their vector representations i.e. sentence embeddings. Afterward, the cosine similarity can be used to compare sentences or rank the sentences with respect to a given input sentence. The cosine similarity between two sentence embeddings u and v is calculated as given in Eq. 1.3,

$$Sim(u, v) = \frac{u \cdot v}{\| u \| \times \| v \|} \tag{1.3}$$

1.3 Research Objectives and Contributions

.

1.3.1 Research Objectives

This thesis aims to investigate how to combine neural networks with external knowledge for spoken dialogue systems. Neural networks obtain strong generalization capabilities on natural language understanding tasks such as intent detection, slot filling, and sentence similarity tasks by learning robust word and sentence vector representation. On the other hand, rule-based systems like regular expressions which are the most representative form of symbolic rules allow experts to express external domain knowledge. The knowledge could be encoded in the form of clue words for particular intent or slot labels in RE surface form. For example, the RE rule can be written as "`(show)(temperature|weather?)`" which detects the weather intent, also encodes the clue words "show temperature" and "show weather" for intent weather. Neural networks may need hundreds of training examples to learn these rules. Moreover, symbolic rules can be easily updated, added, or removed from the systems to support new domains and tasks. Furthermore, RE-based systems do not need labeled training data to generate the results. However, RE-based systems rely on domain experts to carefully write a set of REs, which often have high precision but low recall.

The problem we want to address in this thesis is how to combine neural networks with external knowledge for NLU tasks to take advantage of both neural networks and rule-based systems and reduce the need for large amounts of labeled data. The main objectives of this research are listed as follows: The main objectives of this research are listed as follows:

1. To investigate neural network models deal with NLU tasks and models that leverage domain knowledge contained in symbolic rules to improve their performance, particularly in settings with limited training data.

2. To propose a natural language understanding (NLU) framework for argumentative dialogue systems in the information-seeking and opinion building domain by employing a pre-trained language model to mitigate the problem of small in-domain training data.

3. To develop a WFST-BERT model for intent detection and slot filling, which augments the fine-tuning of BERT-like architecture with weighted finite-state transducer (WFST) to reduce the need for a large amount of supervised data.

4. To propose a multi-task learning model using neural networks with contextual information and integrate it with REs to encode domain knowledge and handle cases with limited amounts of labeled data in an end-to-end trainable manner for multi-turn language understanding tasks.

1.3.2 Thesis Contributions

To accomplish the aforementioned objectives, this dissertation makes the following core contributions.

Contribution 1: Analysis of the existing state-of-the-art methods about NLU tasks, and combining neural networks with knowledge. We present background information and relevant work on

approaches to natural language understanding that address intent detection, slot filling, and sentence similarity problems. We comprehensively review both classic techniques and more advanced deep neural models for natural language understanding tasks. Additionally, we discuss known methods for combining deep neural networks with predefined rules.

Contribution 2: Natural language understanding framework for argumentative dialogue systems. We introduce a natural language understanding (NLU) framework for argumentative dialogue systems in the information-seeking and opinion-building domain. The proposed framework consists of two sub-models, namely intent classifier and argument similarity. The intent classifier model stacks BiLSTM with attention mechanism on top of pre-trained BERT model and fine-tunes the model for recognizing the user intent, whereas the argument similarity model employs BERT+BiLSTM for identifying system arguments the user refers to in his or her natural language utterances. The model is evaluated in an argumentative dialogue system that allows the user to inform him-/herself about a controversial topic and build his/her opinion towards the topic. In order to evaluate the proposed approach, we collect user utterances for the interaction with the respective system and label them with intent and reference argument in an extensive online study. The data collection includes multiple topics and two different user types (native speakers from the UK and non-native speakers from China). The evaluation indicates a clear advantage of the utilized techniques over baseline approaches on several datasets, as well as the robustness of the proposed approach against new topics and different language proficiency as well as the cultural background of the user. Furthermore, results show that our intent classifier model outperforms baseline models in few-shot setups (i.e., with 10, 20, or 30 labeled examples per intent) and full data setup.

Contribution 3: The WFST-BERT Model for Joint Intent Detection and Slot Filling. This work proposes a WFST-BERT model which augments the fine-tuning of BERT-like architecture with weighted finite-state transducer (WFST) to lessen the need for massive supervised data. The WFST-BERT employs regular expressions (REs) rules to encode domain knowledge and pre-trained BERT model to generate contextual representations of user sentences. In particular, the model converts REs into the trainable weighted finite-state transducer, which can generate decent predictions when limited or no training examples are available. Moreover, BERT contextual representation is combined with WFST and trained simultaneously on supervised data using a gradient descent algorithm. The experimental results on the ATIS dataset show that the F1-Score of the WFST-BERT improved by around 1.8%, 1.3% for intent detection and 0.9%, 0.7% for slot filling tasks as compared to its counterparts models in limited data settings. Further, in full data settings, the proposed model generates better recall and F1-score than state-of-the-art models.

Contribution 4: Multi-turn Intent Detection and Slot Filling with neural networks and regular expressions.

Existing models for multi-turn NLU employ memory networks to encode dialogue context, which is used by neural networks for determining user intent and associated slots. However, these methods rely on a large amount of labeled data, whereas we often have limited labeled data. To address this problem, we propose a multi-task learning model based on neural networks and REs, to

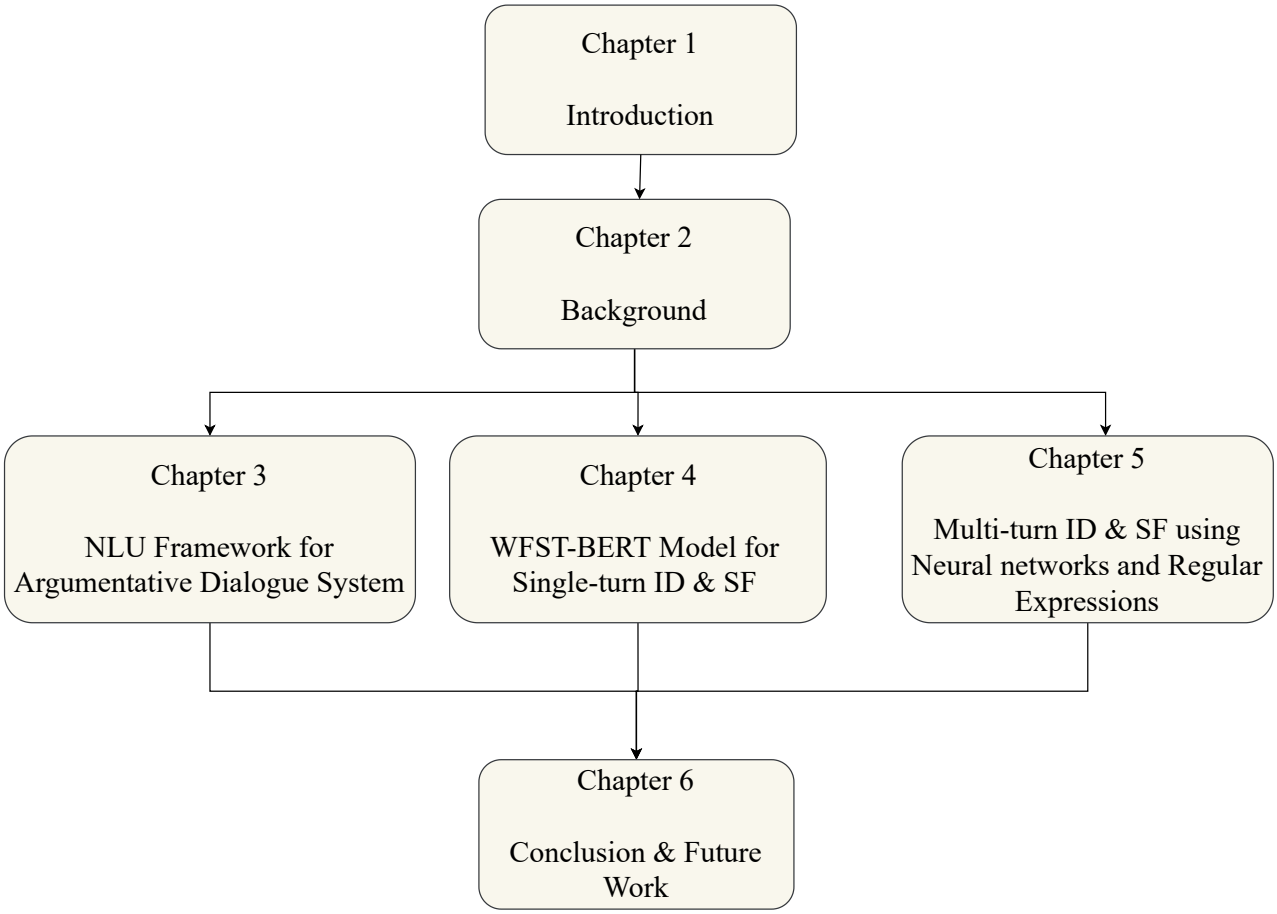


图 1.1: The flow of the dissertation.

jointly perform intent detection and slot filling tasks. The proposed model integrates neural networks with REs to encode domain knowledge and handle cases with a limited amount of labeled data in an end-to-end trainable manner. More specifically, the model employs a pre-trained BERT model to obtain contextual word representations of user utterances. These representations are utilized by a memory network to encode multi-turn information which is shared by the tasks. Furthermore, the convolutional neural network (CNN) and the recurrent neural network (RNN) are applied to contextual word representations and dialogue context for intent detection and slot filling tasks, respectively. These neural networks are then combined with REs which encode domain knowledge about a particular intent or slot value. Finally, the two neural networks are trained simultaneously by minimizing the joint loss. Extensive experiments on Key-Value Retrieval and Frames datasets show that the proposed model outperforms baseline methods in both tasks while requiring modest human effort.

1.4 Dissertation Organization

This dissertation is divided into six chapters that describe each of the critical tasks and contributions as presented in Figure. 1.1.

Chapter 1 discusses the research motivation, problem definition, thesis statement, research objectives, contributions of this research, and organization of the dissertation.

Chapter 2 provides fundamental concepts such as neural network models, regular expressions, and weighted finite-state transducers that will be used throughout this dissertation. Additionally, we review the state-of-the-art research on natural language understanding problems and methods for combining neural networks with symbolic knowledge.

Chapter 3 introduces a natural language understanding (NLU) framework for argumentative dialogue systems in the information-seeking and opinion-building domain. The proposed framework fine-tunes the pre-trained BERT model for intent detection and argument similarity tasks.

Chapter 4 presents a novel WFST-BERT model which augments the fine-tuning of BERT-like architecture with weighted finite-state transducer (WFST) to reduce the need for massive supervised data.

Chapter 5 introduces a multi-task learning model based on neural networks with contextual information for multi-turn intent detection and slot filling tasks. Furthermore, the model integrates REs to encode domain knowledge and regulate neural network output.

Chapter 6 concludes the dissertation with a summary of our main contributions and insights derived from this research. Furthermore, the chapter provides future research avenues on natural language understanding models.

第二章 Background and Literature Review

This chapter provides thorough background information on the dissertation's key approaches. It summarizes state of the art on two topics: natural language understanding models such as RNN, CNN, and SEQ2SEQ models and methods for combining neural networks with Knowledge. The architecture of a task-oriented dialogue system is discussed in Section 2.1. The Section 2.2 explains natural language understanding tasks and subsequent subsections provide their solutions with neural networks such as recurrent neural networks, convolutional neural networks and BERT architecture. The following Section 2.3 refers to earlier work on natural language understanding tasks, specifically intent detection, slot filling, and sentence similarity. Finally, Section 2.4 introduces regular expressions, finite-state transducers, and weighted finite transducers, followed by an overview of previous work integrating neural networks and knowledge. The present chapter includes content previously published in our papers [1-3].

2.1 Task-Oriented Dialogue Systems

As shown in Fig. 2.1, the architecture of a spoken task-oriented dialogue system is composed of a chain of five modules: *speech recognition, natural language understanding, dialogue manager, natural language generation* and *speech synthesis*. The speech recognition module transcribes the user speech into natural language text by exploiting automatic speech recognition (ASR) methods. The text input is passed to the natural language understanding module, which usually performs intent detection, slot filling, and/or sentence similarity tasks. The information obtained from the NLU module is utilized by the dialogue state tracking module to maintain the distribution over all possible dialogue states. The dialogue state is normally expressed with a set of goal slots and probability distributions of possible candidate values for every slot. Based on the dialogue state, dialogue management produces a dialogue act and retrieves information from external knowledge bases (if required). The dialogue act contains a speech act such as inform, request and information presented by slots and val-

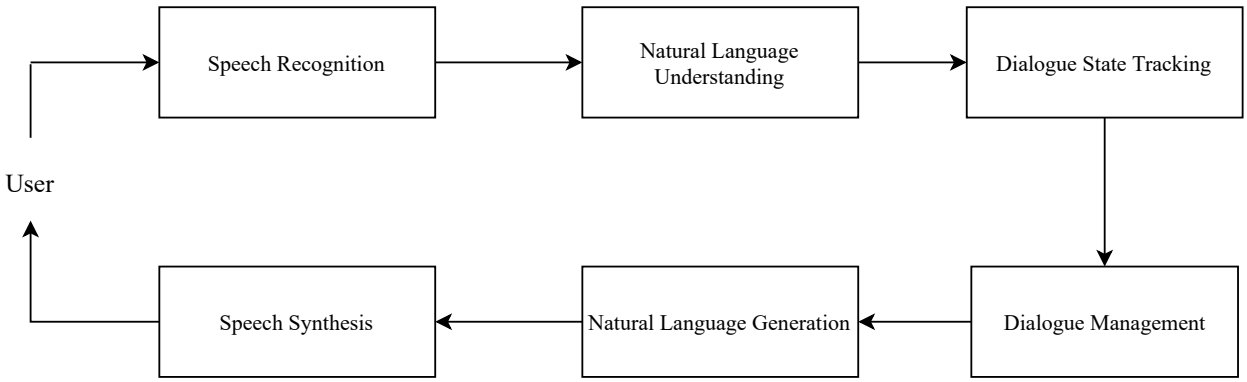


图 2.1: Architecture of a spoken dialogue system.

ues. The dialogue action is given as input to the natural language generation (NLG) module, which generates the natural language formatted response. Finally, speech synthesis converts text to speech and responds to the user. In this thesis, we focus on the natural language understanding module of spoken dialogue systems and review popular NLU methods in the following sections.

2.2 Natural Language Understanding

The natural language understanding (NLU) module is one of the main components of spoken dialogue systems. The NLU module extracts the semantic representations from natural language sentences. Intent detection, slot filling, and sentence similarity are key tasks in the NLU module. The intent detection task [27-29] is normally framed as a sentence classification task. Given an input, the task is to classify the intent label from the pre-defined set of intent candidates. Earlier systems have applied traditional methods such as SVM [30] and more advanced systems have used deep learning models such as recurrent neural networks [28, 29], convolutional neural networks [31-33], and pre-trained sentence encoders [34, 35]. Slot filling [36, 37] is frequently viewed as a sequence labeling problem in which semantic elements are extracted from utterances. Slots can be thought of as variables in an utterance filled from the input text by determining their semantic value. Conditional random fields[38], hidden markov model [39], and recurrent neural networks [37, 40-42] are all prominent methods for slot filling. On the other hand, the sentence similarity task identifies the most similar sentence from the candidate sentences for an input utterance. The recent models for sentence similarity apply pre-trained language models [43-46], convolutional neural networks[47, 48], and recurrent neural networks [49-51]. In the following subsections, we provide review of neural models that are used in our proposed model architectures.

2.2.1 Recurrent Neural Network

Recurrent neural networks (RNNs) are a type of neural network that use feedback connections to store information. RNNs are extremely strong architectures that are capable of capturing long-range dependencies through the use of time-connection feedback [52]. As depicted in Figure 2.2, the RNNs use the output of previous hidden states $t-1$ as input to the current time step t. Therefore, the information presented in the hidden state at time step t is the summarized information of the sequence up to time t. The non-linear activation function such as hyperbolic tangent or sigmoid is applied to the weighted sum of previous hidden states and input vectors as given in Equation 2.1.

$$h_t = f(Wx_t + Uh_{t-1} + b) \tag{2.1}$$

where f represents a non-linear activation function, W and U are the learnable weight matrices for input vector and hidden states, respectively and b is the bias vector. During RNN training, both W and U weight matrices are jointly learned by minimizing a standard loss function.

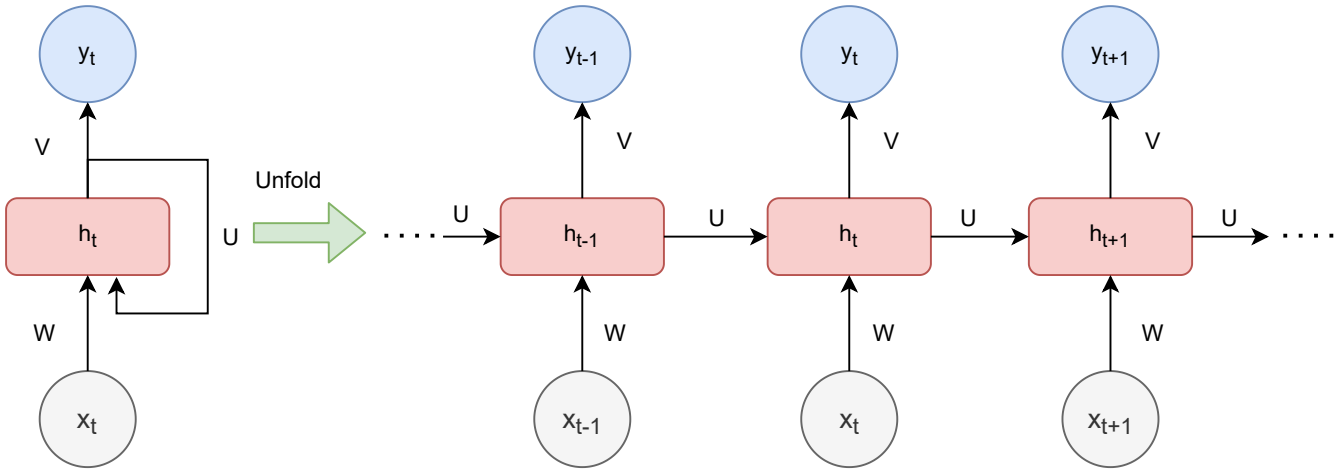


图 2.2: Basic structure unfolded recurrent neural network.

2.2.2 Long Short Term Memory

The standard RNNs suffer from two problems namely vanishing gradient and exploding gradient [53]. In the vanishing gradient, the gradient becomes smaller at every backpropagation step and eventually becomes zero at the end. While in gradient exploding, the gradient becomes too large after backpropagation over time. To address RNNs' vanishing gradient and exploding gradient challenges, long short-term memory (LSTM) [54] and gated recurrent unit (GRU) [55] were developed. LSTMs are similar to RNNs except that memory cells substitute the hidden layer updates. The fundamental concept behind LSTMs is to add different gates to control the flow of information after each time step. The typical LSTM architecture has three gates; input gate, forget gate, and output gate. The input gate specifies the amount of information that should be retained during the current step. The forget gate determines how much information about past states should be maintained and how much should be forgotten. The output gate specifies the amount of information that should be transmitted to the next step from the present output. Figure 2.3, illustrates the basic structure of the LSTM memory cell used in the model. Formally, the memory cell update process in the LSTM network is defined as follows:

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= tanh(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot tanh(c_t),
\end{aligned}
\tag{2.2}
$$

where σ and $\odot$ denote sigmoid function and element-wise multiplication respectively. At time step t, there is an input gate i_t, a forget gate f_t, an output gate o_t, a cell state c_t, a candidate cell state $\tilde{c}_t$, and a hidden state h_t. The learnable parameters for input gate are $[W_i, U_i, b_i]$. The parameters for forget gate and update gate are $[W_f, U_f, b_f]$, $[W_o, U_o, b_o]$, respectively. The candidate cell state parameters are represented by $[W_c, U_c, b_c]$. During LSTM training all parameters are learned jointly.

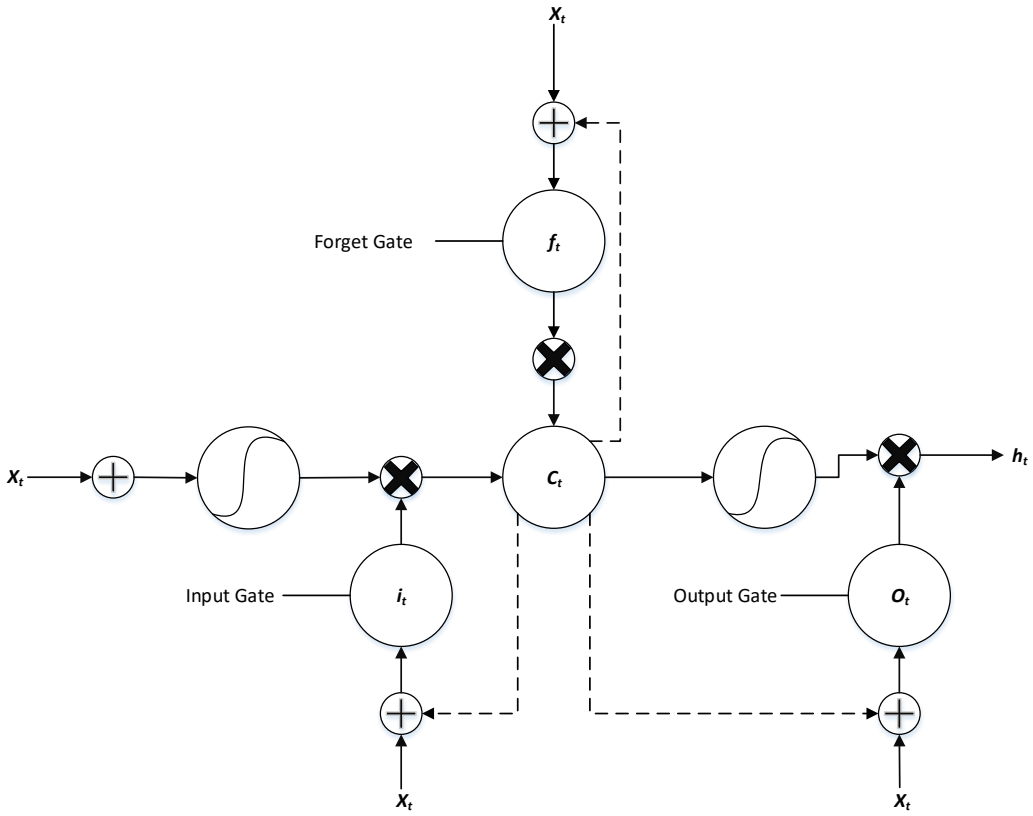


图 2.3: Basic structure of LSTM memory cell. This figure is referred from our paper [3]. Reprinted with permission from Elsevier.

2.2.3 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) [55] can be thought of as a simpler version of the Long Short-Term Memory (LSTM), with the same goal of capturing long-term dependencies and resolving the vanishing and exploding gradient difficulties. In contrast to LSTM, GRU is based on two gates, the update gate and the reset gate, as illustrated in Figure 2.4. The update gate specifies how much information about previous states should be maintained. The reset gate determines whether the prior cell state is retained or reset with the new cell state. The gates in the GRU network are computed informally as follows:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \end{aligned} \tag{2.3}$$

where z_t, r_t represents update and reset gate at time t, respectively and σ denotes sigmoid activation function. The parameters for update are W_z, U_z and parameters for reset gate are W_r, U_r. The candidate activation $\tilde{h_t}$ is calculated by regular recurrent unit, while final activation is the interpolation of candidate activation $\tilde{h_t}$ and previous activation h_{t-1} as given in Equation 2.4.

$$\begin{aligned} \tilde{h_t} &= tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \tilde{h_t} \end{aligned} \tag{2.4}$$

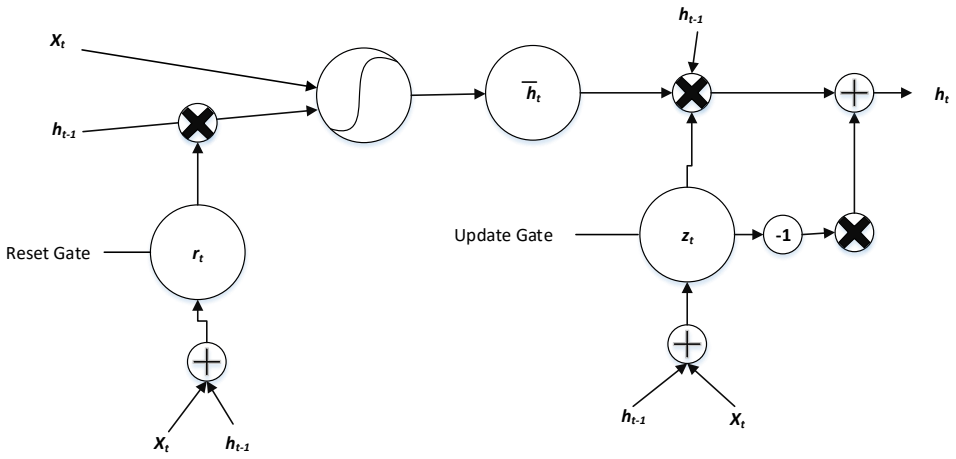


图 2.4: Basic structure of GRU cell.

2.2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are type of neural networks originally designed for computer vision tasks [56-60] have shown impressive results on natural language understanding tasks [31, 61-63]. CNNs use convolutional filters which are applied on top of words representations for incorporating local word context. The 1-dimensional convolution operation convolves a filter W over the window of m words in the input sequence to produce a feature map. For instance, the feature p_i is generated by applying filter W to the window of words $[x_i, \cdots, x_{i:i+m-1}]$ as:

$$\begin{aligned} p_i &= f(W \cdot x_{i:i+m-1} + b), \\ x_{i:i+m-1} &= x_i \oplus x_{i+1} \oplus \cdots \oplus x_{i+m-1}, \end{aligned} \tag{2.5}$$

where b is a bias, $\oplus$ is the vector concatenation operator, and f represents a non-linear activation function.

Traditional convolutional models incorporate local word context information into word representation, where the filter width specifies the local context size used. In order to increase effective context size, several convolutional layers are stacked in a hierarchical structure. However, this setting can incorporate context size linear to the entire depth of the network, therefore makes it harder for tasks that require longer history. To overcome such problems, dilated convolutions are employed that can capture large context by skipping the number of nearby words in subsequent convolutional passes. In particular, a dilated convolution, also known as atrous convolution is a convolution operation in which the receptive field of the network grows exponentially with linear parameter accretion [64, 65]. Besides, to enlarge the receptive field, dilation convolution inserts zeros between filter weights in the standard convolutional filter. For instance, the generic dilated convolution for dilations $1, 2$ and 4 is shown in Figure 2.5.

The dilated convolution operator of filter W_p and dilation factor d applied to m-gram $x_{i+m.d}$ with

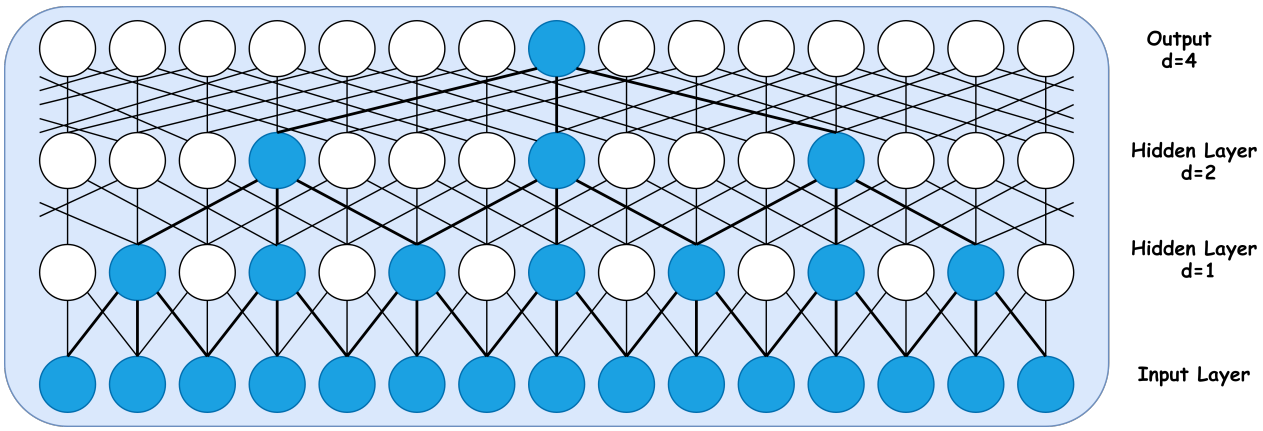


图 2.5: A dilated convolution with dilation factor d = 1,2,4 and filter width = 3. This figure is taken from our paper [3]. Reprinted with permission from Elsevier.

output p_i is defined as:

$$\begin{aligned} p_i &= f(W_p \cdot x_{i+m.d} + b), \\ x_{i+m.d} &= x_i \oplus x_{i+1.d} \oplus \cdots \oplus x_{i+m.d}, \end{aligned} \tag{2.6}$$

where $\oplus$ is the vector concatenation operator, d is the dilation factor, b is the bias, and f is the activation function. The hyperbolic tangent (tanh) is normally used as the non-linear activation function. When the size of the dilation factor is 1, it's equivalent to a regular convolution. Using larger dilation incorporates wider context into the representation of an input token p_i.

2.2.5 Bidirectional Encoder Representations from Transformers (BERT)

The BERT architecture is composed of multiple bi-directional Transformer [66] encoder layers. Each transformer encoder has a multi-head self-attention mechanism and feed-forward networks sub-layers with residual connections. For every word in the user utterance, each head of the self-attention layer computes query, key, and value vectors of dimension d. The model then calculates the dot product of the query with each key, divide all by $\sqrt{d_k}$ followed by the softmax function as given in equation 2.7. The output of each head is concatenated and passed through a linear layer to produce the final weighted representation of the utterance as given in equation 2.8:

$$Att(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \tag{2.7}$$

Where Q, K, V are the query, key, and values matrices.

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(hd_1, hd_2, \cdots, hd_n)W^O \\ hd_i &= Att(QW_i^Q KW_i^K VW_i^V). \end{aligned} \tag{2.8}$$

Where W_i^Q, W_i^K, W_i^V and W^O are parameter matrices.

BERT Utterance Representation

The utterance representations from the BERT encoder are generated as follows: The utterances are tokenized through WordPiece [67] tokenizer that splits the utterance into tokens. Each token representation is obtained by adding its token, position, and segment embeddings. Further, for each utterance special `[CLS]` classification token is added at the start of the utterance. Similarly, a special `[SEP]` separator token is added at the end of each utterance. The BERT encoder then calculates the utterance representation i.e., vector representation for each token denoted as in equation 5.1:

$$H_t^n = BERT(x_{[CLS]}, x_1, \cdots, x_t, x_{[SEP]}) \tag{2.9}$$

where $H_t^n = (h_{[CLS]}^n, h_1^n, \cdots, h_t^n, h_{[SEP]}^n)$, h_t denotes the contextual vector representation of token t, and n represents the number of encoder layer. The last layer hidden state H_t^{12} of every token is given as input to fully connected layers for detecting user intent and corresponding slots.

Up to now, all essential background information for training neural network models for NLU tasks has been provided. The next section reviews state-of-the-art NLU methods.

2.3 Related Work on Natural Language Understanding tasks

Deep learning models such as CNNs and RNNs have been extensively adopted in a wide range of NLU tasks, including intent detection [27-29, 68, 69], slot filling [36, 37, 40], and sentence similarity [45, 46, 49, 50]. In the following subsection, we describe some typical works on intent classification, slot-filling, sentence similarity tasks. The discussion of related work on intent classification, slot-filling, sentence similarity is previously published on our papers [1-3].

2.3.1 Intent classification

Previous works on intent classification use deep learning models such as convolutional neural networks [31, 32] and recurrent neural networks [4, 70] to encode sentence into the fixed-sized vector representation. A classifier is then applied on top of this representation to classify user intent. To generate features from unlabeled data, deep belief networks (DBNs) [68] have been employed. SVM then uses these features to complete the classification task. While deep belief networks have demonstrated impressive feature extraction capabilities for classification tasks, training them has been computationally expensive due to the difficulty of parallelizing DBN training across computers for large datasets. To tackle the scalability issue of DBNs, Deng and Yu have proposed a deep convex network (DCN) [71] which is convex optimization based on batch mode instead of stochastic; therefore it can be parallelized across multiple machines. DCN has proven to be better than DBN in terms of both the accuracy and training scalability of the model. In this connection, Tur et al. [27] have used DCNs for semantic utterance classification and achieved higher accuracy than baseline boosting-based classifier. Recently, RNNs architectures have shown excellent performance on the intent detection task, they take each word of a sentence and use a series of hidden units to temporally model an entire sentence [28, 29]. Furthermore, Lin and Xu [72] proposed SofterMax and deep

novelty detection (SMDN), a post-processing method for detecting unknown intent via pre-trained deep neural network classifiers. On the other hand, Howard and Cambria [73] discussed the importance of the intention-awareness in the human-centric environment where the intentions of actors' are integrated into the surrounding environment. Intention-aware (IA) based systems lessen the informational burden on humans without degrading effectiveness. In this connection, Liu and Zhang [74] have proposed a method called WebIA, which infers human intentions through web queries. The WebIA method establishes an active knowledge database for robots to perform context-specific intention awareness.

Furthermore, Goo et al. [41] proposed a slot-gated mechanism that leverages the intent context vector for modeling slot-intent relationships to improve semantic frame results. Recently, capsule-based architectures have produced excellent results for the intent detection task, these models aggregate word-level features to predict sentences using dynamic routing-by-agreement [75-77]. On the contrary, Bunk et al. [78] proposed DIET (Dual Intent and Entity Transformer) model, which obtains dense features from pre-trained word embedding models such as BERT [43], ConveRT [35], and combine these with sparse word features. These features are then used by 2-layer transformer for detecting user intent. In this connection, Casanueva et al. [34] proposed dual sentence encoders and employed pre-trained Universal Sentence Encoder (USE) [51] and Conversational Representations from Transformers (ConveRT) [35] for predicting user intent. Furthermore, Minaee et al. [79] provides a detailed survey of the pre-trained models for text classification. The survey presents a quantitative analysis of deep learning models performance on text classification benchmarks. Recently, attention based deep neural models [80-82] produce state-of-the-art results on sentiment and emotions intensities classification tasks.

2.3.2 Slot filling

The slot filling is often considered as the sequence labeling task. The RNN architecture and its variants LSTM and GRU perform well for sequence labeling tasks [37, 40]. In this direction, Mesnil et al. [37] proposed RNNs models to produce slot tags sequentially by reading words one by one and efficiently modeling temporal dependencies. The RNN-based model significantly outperforms CRF models on the ATIS benchmark. However, architecture can produce number of slots equivalent to words in the sentence. To tackle this problem, encoder-decoder architecture was proposed which utilizes one RNN to encode input and another RNN to decode output [40]. The encoder-decoder architecture allows systems to match semantic tags and input words of different lengths without aligning input and output.

2.3.3 Joint Tasks

The joint models for intent detection and slot filling were shown to improve the performance of both tasks. In [83] authors modeled domain identification, intent prediction, and slot filling via a single joint RNN model. The RNN reads the sequence of words and sequentially produces slots tags.

The final state of RNN was used for domain detection and intent prediction. The joint model was shown to reduce the overall frame error rate. Liu and Lane [40] further proposed the RNN encoder-decoder model for jointly modeling both tasks. First, hidden states information from slot filling RNN is consolidated, and then the attention model is used to generate its intent. Moreover, Goo et al. [41] proposed a slot-gated mechanism, which introduces an additional gate in LSTM that leverages the intent context vector for modeling slot-intent relationships to improve slot filling performance. Recently, the hierarchical multi-task model [84] was proposed which captures contextual and spatial information of the sentence using CNNs and RNNs. The representations generated through CNNs and RNNs are utilized by the joint model to predict intent and slots.

Moreover, models encoding contextual information from dialogue history have shown higher generalization ability [85-88]. For instance, Chen et al. [85] have used RNNs to encode user and system previous utterances and store them in memory. These memory embeddings are then matched with a current utterance vector using cosine similarity to obtain attention distribution over dialogue history. These attention distributions are then aggregated with memory embeddings to incorporate information from history. Bapna et al. [86] further extended the memory network by adding a session encoder to embed the context in chronological order. Encoding contextual information in sequential order results in a reduction of frame error rate. In the same direction, Su et al. [87] introduced a time-aware attention mechanism to pay more attention to recent utterances of dialogue by decaying attention weights over time. On the other hand, Kim et al. [88] employed two separate memories for the encoding system and user utterances differently depending on the speaker. Dual Memory Networks obtained significant performance improvement over the state-of-the-art contextual slot tagging models. Recently, Firdaus et al. [84] proposed a hierarchical convolutional neural network and hierarchical convolutional recurrent neural network for jointly modeling of intent detection and slot filling tasks. The related work on joint models for intent detection and slot filling first appeared in our papers [2, 3].

2.3.4 Pre-trained Language Models

The pre-trained language models such as ELMo [89], ULMFiT [90], GPT [91], BERT [43], XLNet [44], and more recently UNILM [92], MT-DNN [93], and ERNIE [94] trained on large unlabeled corpora are dominating in natural language understanding leaderboards such as GLUE [95] and SuperGLUE [96]. The ELMo [89] model learns deep contextualize word representation by training two-layer bidirectional LSTM with language modeling objective on the large text. The downstream model combines representations from all BiLSTM outputs using downstream task-specific weights. On the contrary, GPT [91] model adopted a discriminative fine-tuning approach when it is applied on downstream tasks. For every task, the task-specific layer is added on top of the pre-trained model and the whole model is fine-tuned with labeled data. In the same direction, ULMFiT [90] model employed discriminative fine-tuning with triangular learning rates. The ULMFiT uses gradual unfreezing of pre-trained parameters to retain the previous information and obtains state-of-the-art results for text

classification tasks. An extensive study of the pre-trained models for text classification was done by Minaee et al. [97]. The study provides a quantitative analysis of the performance of different deep learning models for text classification on popular benchmarks. On the other hand, Peters et al. [98] explored how to adapt pre-trained models for down streaming tasks. They focused on feature extraction and fine-tuning of pre-trained model adaptation methods and performs experiments on diverse NLP tasks. Peters et al. [98] concluded that fine-tuning and feature extraction approaches have comparable performance in most cases and their relative performance depends on the similarity between the pre-training and the target tasks. In this thesis, we extract contextual representations of the words from the pre-trained BERT model which are utilized by the RNN and CNN network for determining user intent and associated slots. The related work on pre-trained language models was based on our paper [3].

2.3.5 Sentence Similarity

For the textual similarity task, Skip-Thoughts [49] model was proposed which extends the word2vec [99] skip-gram approach from the word-level to sentence-level. The Skip-Thoughts model employs encoder-decoder architecture to learn the vector representation of the sentences. Similarly, FastSent [100] was proposed which replaced the RNN encoder with word embedding summation of the skip-thoughts model. In contrast, InferSent [50] model utilizes supervised data of the Stanford Natural Language Inference (SNLI) dataset to train a siamese BiLSTM network with max-pooling over the output. The InferSent model outperforms unsupervised methods like Skip-Thoughts. The Universal Sentence Encoder (USE) [51] model extends the InferSent model by training transformer architecture and augmenting unsupervised learning with supervised training objectives.

In parallel, pre-trained models such as ELMo [89], GPT [91], BERT [43], XLNet [44], ERNIE [94], and MT-DNN [93] trained on large unsupervised corpora, shown significant improvement for intent classification, semantic textual similarity, and various natural language understanding tasks [95]. Such pre-trained models allow the downstream task model to be fine-tuned without training from scratch. Furthermore, fine-tuning of pre-trained models produced state-of-the-art results for text classification [90, 101] and textual similarity [45, 46]. Inspired by the recent works, we have employed these pre-trained models for the natural language understanding of the arguments. The related work on textual similarity was first published in our paper [1].

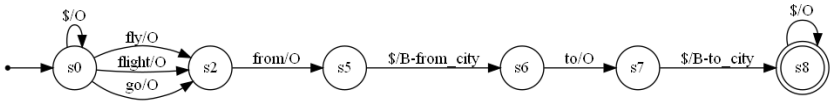
2.4 Combining neural network with Knowledge

This section and the subsequent section include content previously published in our paper [2].

2.4.1 Regular Expressions

Regular Expressions (REs) are extremely strong techniques for describing text patterns and are frequently used for pattern matching, entity recognition, information extraction, and sentence classi-

表 2.1: A regular expression for detecting the intent "flight" and associated slots label of the matched sentence. Here '$' denotes a wildcard pattern that matches any single word, '$*$' asterisk represents the Kleene star operator, and '|' is the OR operator. This table is taken from our paper [2] with kind permission from Springer.

Intent	flight
Slots	[O O O B-from_city O B-to_city O]
RE	`$ * ( flight | go | fly ) from $ to $ $ *`
Matched Text	<BOS> i want to **fly from Baltimore to Dallas** <EOS>
FST	

fication [24, 25, 102]. REs enable experts to inject domain knowledge into deep neural models and regulate them to comply with desired patterns [3]. REs can be employed at the sentence level and/or word level. At the sentence level, REs scan the user input for key phrases and return the user's specific purpose to the matched sentence. At the word level, REs find word sequences in user text and assign matched word-level RE tags. Table 2.1 demonstrates the example of RE matching a sentence asking for intent "flight" and identifying the slots B-from_city and B-to_city as Baltimore and Dallas, respectively. We write n REs $R = \{r_1, r_2, \cdots, r_n\}$, each RE has sentence-level label l and word-level m labels $\{s_1, \cdots, s_m\}$.

2.4.2 Finite State Transducer (FST)

A finite-state transducer (FST) is a type of finite automata (FA) that consists of a finite number of states, an input label, and an output label. The finite-state acceptor (FSA) is the most frequently seen automaton, which accepts or rejects input sequences based on whether the FSA has a path from the beginning to the final state. The FST inherits the properties of the FSA, but in addition to returning the accepted or rejected value, it returns the output symbol sequence. The FST can be seen as a directed graph, its vertices represent states and edges represent input/output pair (labeled transitions). The FST transits from one state to another state by reading the input alphabet and producing an output label. An FA can be constructed for each RE using Thompson's construction algorithm [103], which first converts RE into Non-deterministic finite automata (NFA) and from NFA to deterministic finite automata (DFA). Automation is called DFA if it has deterministic transitions which means given input on the current state there is a unique next state.

2.4.3 Weighted Finite State Transducer (WFST)

A weighted finite-state transducer (WFST) extends an unweighted finite-state transducer by assigning weights to each transition. A WFST is a 6-tuple $\mathcal{T} = (\Sigma, \Delta, Q, T, \lambda, \rho)$ where

- Σ is a finite non-empty set of input alphabets.

- Δ is a finite non-empty set of output labels.
- Q is a finite non-empty set of states.
- $T \in \mathbb{R}^{|\Sigma| \times |Q| \times |Q|}$ is the weighted transition tensor. $T(x, q_a, q_b)$ is the weight of transiting from state $q_a \in Q$ to state $q_b \in Q$ on receiving input $x \in \Sigma$.
- $\lambda \in \mathbb{R}^{|Q|}$ is the weight vector of initial states.
- $\rho \in \mathbb{R}^{|Q|}$ is the weight vector of final states after reading all input.

Let $p = (q_1, \cdots, q_{n+1})$ be a path from the initial state q_1 to the final state q_{n+1} after consuming sequence x where q_n is the $n - th$ state, score of the path p is defined as follow:

$$S(\mathcal{T}, x) = \lambda[q_1] \cdot \left(\prod_{t}^{n} T[x_t, q_t, q_{t+1}] \right) \cdot \rho[q_{n+1}] \tag{2.10}$$

The maximum score from all accepting paths $\pi(x)$ starting from initial state q_1 and reaching the final state q_f after consuming input x is calculated by using the Viterbi [104] algorithm as follow:

$$S(\mathcal{T}, x) = \max_{p \in \pi(x)} \lambda \cdot \left(\prod_{t}^{n} T[x_t] \right) \cdot \rho \tag{2.11}$$

The Viterbi algorithm returns accepting path and output labels with maximum score. The Weighted Finite-State Acceptor (WFSA) [105, 106] can be viewed as a special case of WFST and can also be represented similarly by omitting output labels.

2.4.4 Related Work on Combination of Neural Networks with Knowledge

The combination of neural networks with pre-defined rules has been an active research topic [102, 107, 108]. In this connection, Hu et al. [107] used first-order logic rules to adjust the output probability of a neural network and then train the student network from the rule-regularized teacher network. Alashkar et al. [108] trained a neural network from examples and knowledge base rules by minimizing a joint loss of examples-based and rules-based network. Similarly, Awasthi et al. [109] injected rules into the neural network via multitask training that jointly denoises rules using latent coverage variables. Furthermore, Xu et al. [110] incorporated symbolic knowledge in the neural networks via semantic loss function that constraints neural networks output to comply with rules. Li and Srikumar [111] introduced declarative knowledge as first-order logic rules into neural networks by constraining the output logits or guide the network using attention scores. On the other hand, Waqas et al. [112] combined hand-crafted features obtained from local binary pattern network with features generated by a neural network for tuberculosis analysis. Similarly, Luo et al. [102], incorporated knowledge of regular expressions into the training of a neural network as an input feature, as an attention guide, and as a regularization of neural network output. In contrast, Rule-Guided Embedding (RUGE) [113] was proposed that inject soft logic rules into learned knowledge graphs (KGs) embeddings.

On the other hand, Zhou et al. [114] proposed a model that leverages commonsense knowledge using a knowledge graph in neural dialogue systems. The model retrieves knowledge graphs for each user utterance and then encodes the graphs with an attention mechanism to augment the semantic information and thus supports a better understanding of utterances. Similarly, Guan et al. [115] have proposed the neural-based model that utilizes commonsense knowledge by multi-source attention to generate the story ending for a given story context. Recently, Young et al. [116] augmented the Seq2Seq framework with audio features of the user message for neural conversation generation and outperformed the audio-free models. The latent intention dialogue model (LIDM) [117] was proposed that employs a discrete latent variable to learn the complex distribution of dialogue intentions. The latent variables represent the dialogue intentions, based on which a dialogue agent generates appropriate responses. The LIDM model was trained using supervised using reinforcement learning manners. Likewise, Xu et al. [118] proposed a hierarchical encoder-decoder that exploited a discrete latent variable to learn dialogue intentions. The model augments latent intention inferred from the user utterance and trained end-to-end manner under unsupervised, semi-supervised, or reinforcement learning. In contrast, Jiang et al. [106] converted REs rules into trainable neural networks for the text classification task. Zhang et al. [25] leveraged REs to generate weak labels for entity mentions from unsupervised data and these weak labels are predicted by deep learning models during training. Locascio et al. [119] proposed the LSTM model to generate regular expressions from the natural language specifications of REs. On the other hand, Zhang et al. [25] employed regular expressions for generating weak labels of the entity mentions from unlabeled data and then trained a neural network to predict those RE-generated weak labels.

The above discussion of related work on the combination of neural networks with knowledge was published in our papers [2, 3].

第三章 Natural Language Understanding for Argumentative Dialogue Systems

3.1 Introduction

In the following natural language understanding is discussed with respect to Argumentative Dialogue Systems. The following chapter elaborates the work that has been conducted within the scope of a joint PhD program between Southeast University and Ulm University, Germany and has been published in Knowledge-Based Systems Journal [1].

The vast amount of often contradicting information in online sources has raised the need for technologies and applications that assist humans in processing and evaluating them. While recent developments in the field of argument mining [120] provided the tools to automatically retrieve and structure such information from various sources, the resulting data structures are still large and not necessarily intuitive to humans. Argumentative dialogue systems and conversational agents on the other hand can process these structures [121, 122] and provide a natural and intuitive interface to the data. However, the capabilities of such systems are limited by their ability to understand and process user responses to the presented arguments, especially if they are presented by means of natural language.

Within this chapter we introduce a natural language understanding (NLU) approach for the information seeking and opinion building scenario discussed above that extracts the required information from the user input:

a) Direct user commands to the system (for example end discussion, provide more information)

b) System arguments referenced in the user utterance,

c) User sentiment on the referenced system arguments.

In order to do so, we first recognize the general user intent, i.e. the type of utterance or speech act. In a second step, we use semantic similarity measures to identify the argument that the utterance refers to (if this is required).

Natural language understanding in the argumentation domain in general is quite challenging [123] which can be attributed to the complexity of the domain and the comparatively small amount of conversational training data [124]. Consequently, NLU components in argumentative systems often suffer from small and/or domain-specific training data that hinders the generalization capability.

One main challenge is to design an NLU component that works in low-data scenarios where only several examples available per system-specific intent (i.e., so-called few-shot learning setups). Recently language models, such as Bidirectional Encoder Representations from Transformers (BERT) [43] trained on large-scale unlabeled corpora have achieved state-of-the-art performance on natural language processing tasks after fine-tuning. These large-scale pre-trained language models generate

contextualized word embeddings and also encodes transferable linguistic features such as parts of speech and syntactic chunks [125].

Considering the benefits of the pre-trained language models, we have utilized the BERT model for the herein discussed NLU approach. More precisely, we have fine-tuned BERT for two NLU tasks, namely intent classification and argument similarity. The proposed intent classifier model stack Bidirectional-LSTM (BiLSTM) with an attention mechanism on top of pre-trained BERT model and fine-tune the whole model. However, fine-tuning BERT on a small dataset may result in overfitting which leads to performance degradation. Therefore, the proposed intent classifier combines sentence representation from the argument similarity model with the representation of BERT+ BiLSTM to improve intent classification performance on a small dataset and few-shot setups. The argument similarity model uses a large supervised Semantic Textual Similarity (STS) [126] benchmark dataset for training and produce high-quality sentence representation. The argument similarity model combines word features from the BERT layer and the BiLSTM layer to produce high-quality sentence representations which can be compared using cosine similarity. We test the presented approach in the argumentative dialogue system BEA [127] that assists the user in building an opinion on a specific topic by providing incremental information and tracking the preference of the user towards it. In order to train and evaluate the proposed model, we collect user utterances labeled with intent and referenced arguments for the interaction with the BEA system for three different topics in an extensive user study. Apart from testing our model in the BEA system, we evaluate the proposed intent classifier and argument similarity models on the publicly available Banking77 and STS benchmark datasets. The results are used to evaluate our approach in four different categories:

a) We compare intent classifier and argument similarity models separately to baseline approaches on the User Study, Banking77, and STS benchmark datasets to test the robustness of the proposed model on different domains.

b) We look at a few-shot intent classification scenario where only 10, 20, or 30 training examples for each intent are sampled from full training data to test model performance in absence of sizeable training data.

c) We train and evaluate the complete pipeline (intent classifier and argument similarity) on separate topics of the User Study dataset to assess the robustness of the model against topic changes.

d) We collect a separate test data set from non-native speakers with a different cultural background (Chinese students) in order to test the robustness of the proposed model against different levels of language proficiency and cultural diversity.

The experimental results show a clear advantage of our proposed approach over the baselines for the intent classification and argument similarity tasks on different datasets. Moreover, the outcomes indicate a high and stable performance of the model for data from topics unseen during training and different language proficiency.

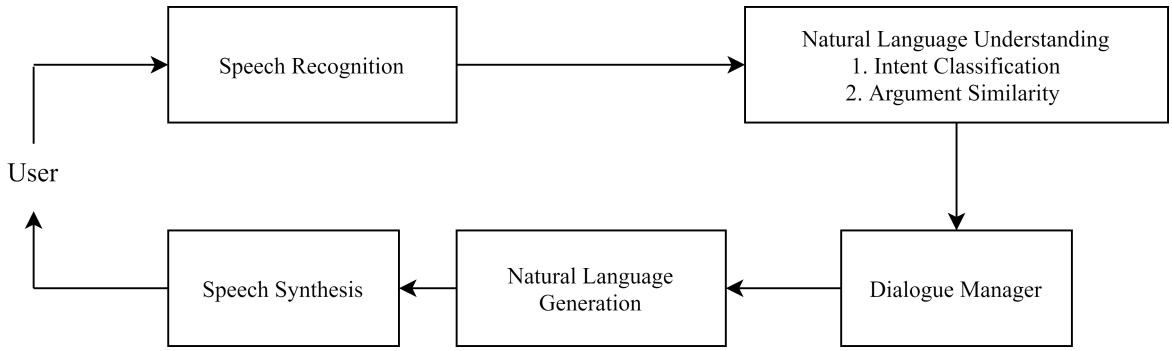


图 3.1: Architecture of a spoken argumentative dialogue system. First published in our paper [1]. Reprinted with permission from Elsevier.

3.2 Existing Argumentative Dialogue Systems

A variety of different argumentative dialogue systems have been introduced in the past years. In [128] a dialogue system to enable a computer to engage its users in debate on a controversial issue was introduced. In [127] the BEA system is proposed which is an argumentative dialogue system that helps a user to form his or her opinion on a certain topic by providing arguments in a spoken dialogue. Rach et al. [121] proposed EVA System to discuss controversial topics with the users. Hunter [129] discusses formal models of dialogues involving arguments and counterarguments of user models, and strategies, for automated persuasion system (APS). Ma et al. [130] provides a review of empathetic dialogue systems that respond to users in an empathetic way.

On the other hand, argumentative systems that include natural language input like the IBM Debater[1] are mainly focused on the exchange of arguments in competitive setups like debates, discussions or persuasion [124, 131-133]. For instance, Rosenfeld and Kraus [134] proposed a methodology for persuading people through argumentative dialogues to invoke an attitude and behavior change. In contrast, we look at a cooperative setup and focus explicitly on the exchange of information and opinions *about* the arguments presented by the system rather than a recognition of user arguments. Discussion on existing argumentative dialogue systems was first published in our paper [1].

3.3 Natural Language Understanding Framework

As shown in Fig. 3.1, the architecture of a spoken human-computer dialogue system is composed of a chain of five modules: *speech recognition, natural language understanding, dialogue manager, natural language generation* and *speech synthesis*. In the following, we will focus on the second module, the Natural Language Understanding. It processes text from the lexical and syntactic levels, converting it into dialogue action at the discourse level. Therefore in general an intent recognition takes place and potential additional information is analyzed. In our setup, the identified user intents are referred to as *speech acts*, which are determined by the used dialogue model. In order to understand and process the user utterance correctly, *what* the user wants (intent classification) and *which*

[1] https://www.research.ibm.com/artificial-intelligence/project-debater/

argument the user refers to (argument similarity) have to be identified. Thus, the herein presented NLU framework consists of two components, an intent classifier model and an argument similarity model. The user intent might consist of direct commands to the system (like "End the conversation") but can also include expressed opinions, sentiments, or preferences towards arguments presented by the system. Given a successful recognition of the user intent, the referred content of the latter has to be identified. This is accomplished by comparing the user utterance to the known arguments by the means of semantic similarity measures. Further details of both NLU models, as well as the dialogue system utilized throughout this work are given in the following sections.

3.3.1 BEA: An Application Scenario

The herein described NLU framework is tailored to be integrated in the argumentative dialogue system BEA introduced by Aicher et al. [127] which conditions the tasks the NLU addresses. Therefore, a short overview is given over BEA's dialogue model and framework. BEA incrementally presents arguments on a controversial topic, allows users to express preferences towards and between these arguments and utilizes the responses to model and monitor the user opinion in view of the discussed topic throughout the interaction. The goal of BEA is to engage the users into an intuitive and natural dialogue allowing them to explore different arguments with diverging stance and various subtopics. In contrast to competitive systems BEA does not pursue a persuasive approach but tries to provide pro and con aspects on a controversial topic to help the user to build a balanced opinion.
In order to navigate through a large amount of arguments and divide the discussed information in reasonable and logically consistent parts, the system utilizes an argument structure based on the argument annotation scheme introduced by Stab et al. [135]. This scheme was originally introduced for annotating argumentative discourse structures and relations in persuasive essays and meets our purpose to offer the user a fair chance to decide unprejudiced which side (pro/con) to prefer or reject. According to Stab et al. an argument consists of several argument components (*major claim*, *claim* and *premise*) and two relations (*support* and *attack*) between them. Usually a single *major claim* formulates the overall topic of the debate, representing the root node in the tree graph structure. Thus, it is the only component without target. Likewise to Aicher et al. [127] in the following we use sample debate from the *Debatabase* of the idebate.org[2] website with the *major claim Marriage is an outdated institution*.
Claims are allegations which formulate a certain opinion targeting the *major claim* but still need to be justified by further arguments, *premises* respectively. Hence, a *claim* (parent node) is either supported or attacked by at least one other premise (child node). For the remainder of this work, we refer to a single node, i.e., an argument component in the structure as *argument*.
We only focus on non-cyclic graphs, meaning that each premise only targets one other component, leading to a strictly hierarchical structure. Furthermore, the annotation scheme distinguishes two di-

[2]https://idebate.org/debatabase (last accessed 16 September 2021).

表 3.1: Available speech acts and their corresponding user and system action. The table is taken from our paper [1]. Reprinted with permission from Elsevier.

Speech Act	**User Action**	**System Action**
stance	Request for the current overall stance of the user.	WBAGs are used to calculate and returns current user stance on the overall topic.
exit	Request to end the interaction with the system.	System terminates interaction with complimentary closing.
level-up	Request to return to the previous argument (parent).	Changes the current state and switches to the corresponding parent node (one level up).
why(argument)	Request for further information on an argument.	Provides information about the current argument by introducing all of its child nodes.
prefer(argument)	States preference towards the referenced argument over all its siblings.	Calculates new stance according to the preference model and updates tree.
reject(argument)	Rejects an argument.	The argument and all its corresponding child nodes are rejected, thus, a new stance is calculated according to the preference model and the tree is updated.

rected relations a premise can have towards a claim (*support* and *attack*). Between sibling nodes, there exists no explicit relation.

Due to the generality of the annotation scheme, the system is not restricted to certain data and generally every argument structure that can be mapped into the applied scheme can be processed by the system.

In the interaction, BEA introduces sibling arguments related to the same parent argument in the tree simultaneously. In particular all available argument components attacking or supporting the parent node are introduced. Thus, the user is able to express preferences[3] between the siblings or navigate to another sub-structure depending on his/her interest. If a user expresses a preference, it is crucial the system can identify the user intent and the sibling which is preferred. This expressed preference is incorporated into a calculation that determines the user's overall opinion on the topic of the discussion and is updated in real-time during the interaction.

The interaction is divided in turns such that each user action ('move') is followed by a system

[3]For ease of reading, 'preferences' here denotes both a negative (rejecting) and positive (approving/preferring) attitude towards an argument.

response and vice versa. The six different user moves the user is able to choose from and the corresponding system actions are shown in Table 3.1. These moves are equivalent to the intents that have to be recognized by the NLU.

Three moves (*prefer*, *reject*, and *why*) refer to a specific argument and require the NLU to identify this argument. Whereas *prefer*, *reject* allow the user to express his or her opinion towards the argument, the *why* move can be used to ask the system for further information on the argument. Thus, the selected argument the *why* move refers to, becomes the parent node and its attacking and supporting children are displayed. Since BEA introduces only siblings related to one parent node at the same time, the list of siblings serves as the list of possible reference arguments for the NLU.

In addition to that, the user is able to request the calculated opinion (weight) on an argument (*stance*). To calculate the user stance the system uses the preference statements and determines the respective stance by utilizing weighted bipolar argumentation graphs (wBAGs) [136]. Therefore the weights are iteratively determined following the tree structure and updated after every preference statement. The user can navigate through the tree, by using the *why* move to descend and the *level up* move to ascend. The latter takes him/her back to the previous argument. The conversation with BEA can be terminated by the user at any time by performing the *exit* move.

The natural language generation of the system is based on the original textual representation of the argument components. As discussed in [127] the annotated sentences were slightly modified to form a stand-alone utterance which serves as a template for the respective system response. Additionally, a list of natural language representations for each type of move was defined. During the generation of the utterances, the explicit formulation and introductory phrase is chosen from this list randomly. E.g. regarding the *why* move the relation (*support* or *attack*) of the presented children is clearly stressed by specific formulations, such as *This claim is supported by the argument that...* and *A contrary indication is the fact that....*

In the following, an exemplary dialogue, taken from [127] is shown to demonstrate the targeted user system interaction. The presented arguments are taken from a sample debate from the *Debatabase* of the idebate.org[4] website.

In claim[5] *"Marriage undermines same-sex couples and single parent families as legitimate ways of raising children."*, which is addressed by the user with a *why* move.

System: *"This claim is supported by the argument that marriage is seen as the best way to raise children (C1). Another argument in favor is the existence of marriage is essentially saying that same-sex couples and single parents are less able of raising children than heterosexual couples (C2). A contrary indication is the fact that the idea that the existence of marriage undermines other methods of raising children is ridiculous (C3)."*

[4] https://idebate.org/debatabase (last accessed 09 January 2018).

[5] For the sake of simplicity we define this argument as the *major claim* of this subdialogue.

User: *"I do not believe that this is the best way to raise children."*

System: *"It is interesting that you rejected this claim (C1)."*

In order to realize this dialogue the system has to fullfill two tasks. First, the intent of the user has to be identified (rejection) and second, it has to determine which of the three claims the user wants to reject ("marriage is seen as the best way to raise children"). Therefore, the intent classification model addresses the first problem, introduced in Subsection 3.3.2. The second part represents the argument similarity model described in Subsection 3.3.3. Both taken together form the complete system pipeline.

3.3.2 Intent Classifier Model

The architecture of the proposed intent classifier model is depicted in Fig. 4.5. The model consists of two main components namely: BERT Transformer Encoder and BiLSTM classifier. Further details of these components are given in the following subsections.

BERT Transformer Encoder

The BERT model computes user utterance representation by splitting the utterance into a list of tokens and then combines token, position, and segment embeddings for producing a fixed-length vector. Moreover, the special classification `[CLS]` token is added at the start of each utterance. Similarly, special `[SEP]` token is inserted at the end of each sentence as a final token. The BERT encoder then computes the user utterance representations i.e. hidden states for each token x_t as shown in equation 5.1.

$$H_t^n = BERT(x_{[CLS]}, x_1, .., x_t, x_{[SEP]}) \tag{3.1}$$

where $H_t^n = (h_{[CLS]}^n, h_1^n, .., h_t^n, h_{[SEP]}^n)$, n denotes number of BERT encoder layer, and h_t is the contextual representation of token t. The final hidden state $H_t^N (N = 12)$, of each token is passed to a task-specific LSTM layer.

Bidirectional-LSTM on BERT

The Long short-term memory (LSTM) [54] is a powerful architecture capable of capturing long-range dependencies via time-connection feedback. Our proposed model stacks a Bidirectional-LSTM (BiLSTM) on top of the final BERT encoder layer. The forward LSTM and a backward LSTM of BiLSTM read in the final hidden states of all the words $H_t^N (N = 12)$, produced by BERT in two opposite directions and generates output sequences $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$. The two outputs are then concatenated to access both past and future context for a given word as given in equation 3.2.

$$\begin{aligned} \overrightarrow{h_t} &= \overrightarrow{LSTM}(H_1^N, ..., H_t^N) \\ \overleftarrow{h_t} &= \overleftarrow{LSTM}(H_1^N, ..., H_t^N) \\ h_t &= [\overrightarrow{h_t}, \overleftarrow{h_t}] \end{aligned} \tag{3.2}$$

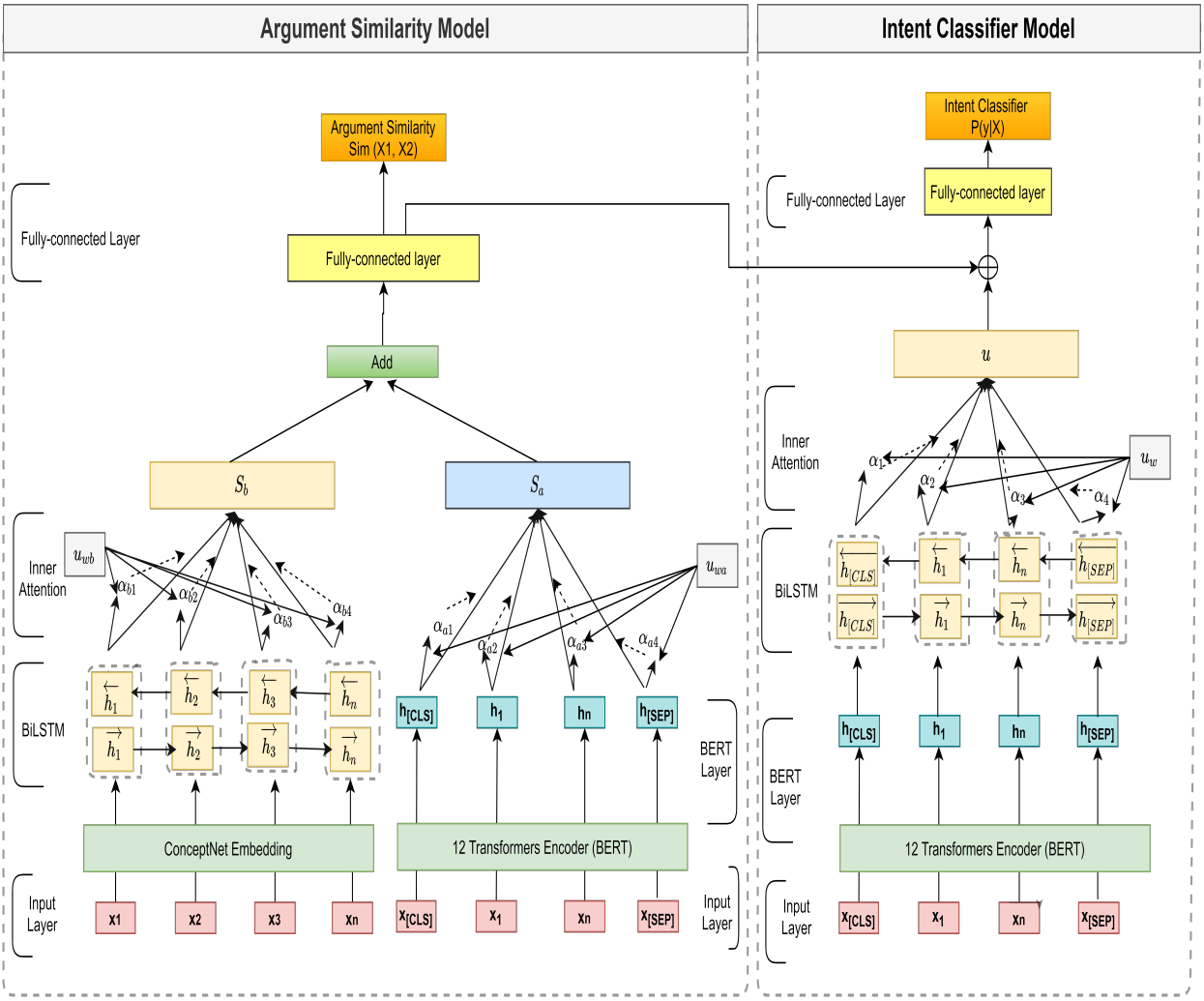


图 3.2: An illustration of argument similarity and intent classifier model. Argument similarity model generates sentence representations S_a, S_b by applying inner attention on BERT encoder and BiLSTM encoder, respectively. Intent classifier model obtains sentence representation u by BERT+BiLSTM encoder. The final representation is produced by concatenating sentence representation u and s and passing through a fully-connected layer. The figure is taken from our paper [1] with permission from Elsevier.

where h_t is the representation of the given word obtained by concatenating the forward hidden state $\overrightarrow{h_t}$ and backward hidden state $\overleftarrow{h_t}$.

Inner-Attention

To encode the variable-length sentence into a fixed-sized vector representation, we employ an attention mechanism. The attention mechanism is used to give more focus on the important information of the sentence. The attention mechanism is applied to the whole hidden states $H = (h_1, h_2, ..., h_t)$ of BiLSTM to generate vector representation of the sentence. As suggested in [137], we use multiple attention views to focus on a different part of the sentence. The attention mechanism is defined as follows:

$$\begin{aligned} u_w &= W_2(tanh(W_1 H^\intercal + b_1) + b_2 \\ \alpha &= softmax(u_w) \\ u &= H\alpha^\intercal \end{aligned} \tag{3.3}$$

where $W_1 \in \mathbb{R}^{d_w \times 2d}$ and $W_2 \in \mathbb{R}^{r \times d_w}$ are the trainable weight metrics; $b_1 \in \mathbb{R}^{d_w}$ and $b_2 \in \mathbb{R}^r$ are the trainable bias, here r is the number of attention heads, d represents the number of hidden units of LSTM, d_w is the size of vector parameters we can set arbitrarily. The softmax function is applied along the second dimension of its input, which ensures computed weights sum up to 1. We then compute the weighted average of context vectors r by multiplying the attention matrix α and LSTM hidden states H to generate the sentence representation u. The final representation u along with sentence representation s obtained from the argument similarity model is given as input to a fully connected layer for predicting the corresponding user intent (speech act).

$$\hat{y} = \text{softmax}(W[u, s] + b) \tag{3.4}$$

where $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}}$, $W \in \mathbb{R}^{d \times |I|}$ and $b \in \mathbb{R}^{|I|}$ are the weights and bias of the fully connected layer respectively, s is the sentence representation as defined in Eq 3.8, and I denotes the user intent vocabulary. Furthermore, the model minimizes the cross-entropy loss between true user intent and predicted user intent $\hat{y}$.

$$\mathcal{L} = -\sum_{i=1}^{l} y^i log(\hat{y}^i) \tag{3.5}$$

where i is the number of intents while y^i is actual user intent and $\hat{y}^i$ is predicted user intent.

3.3.3 Argument Similarity Model

The argument similarity model operates on the contextual word features obtained from the BERT model and word features with common sense knowledge obtained from training BiLSTM on top of ConceptNet Numberbatch [138]. At a high level, our model consists of two parts: BERT Encoder and BiLSTM.

In the first part, each word of the sentence is passed through BERT encoder layers and output vectors are given as input to the inner-attention layer. Attention mechanism provides summation vectors which are dotted with BERT output vectors to generate a sentence vector. This yields

$$\begin{aligned} H_a &= BERT(x_{[CLS]}, x_1, .., x_t, x_{[SEP]}) \\ u_{wa} &= W_{a2} tanh(W_{a1} H_a + b_{a1}) + b_{a2} \\ \alpha_a &= softmax(u_{wa}) \\ S_a &= H_a \alpha_a^\intercal \end{aligned} \tag{3.6}$$

where $W_{a1} \in \mathbb{R}^{d_w \times d}$ and $W_{a2} \in \mathbb{R}^{r \times d_w}$ are the weight metrics; $b_{a1} \in \mathbb{R}^{d_w}$ and $b_{a2} \in \mathbb{R}^r$ are the bias, where r is the number of attention heads, d represents the size of BERT output vectors. We compute weighted average of context vectors r by multiplying the attention matrix α_a and BERT output vectors H_a to generate the sentence representations_{1a}.

In the second part, each sentence is passed through the ConceptNet Numberbatch embedding layer to obtain a semantic vector for each word. ConceptNet Numberbatch embedding combines embeddings from word2vec [139], GloVe [140], and structured knowledge from ConceptNet [141], which provide common sense knowledge along with surrounding word context. These word embeddings are given as an input to the BiLSTM layer for modeling the temporal relationship between word embeddings. Then, the model utilizes an attention mechanism on the hidden states of BiLSTM to generate a vector representation of the sentence.

$$\begin{aligned} \overrightarrow{H_b} &= \overrightarrow{LSTM}(x_1, .., x_t) \\ \overleftarrow{H_b} &= \overleftarrow{LSTM}(x_1, .., x_t) \\ H_b &= [\overrightarrow{H_b}, \overleftarrow{H_b}] \\ u_{wb} &= W_{b2} tanh(W_{b1} H_b + b_{b1}) + b_{b2} \\ \alpha_b &= softmax(u_{wb}) \\ S_b &= H_b \alpha_b^\intercal \end{aligned} \tag{3.7}$$

where H_b represents all hidden states of BiLSTM; $W_{b1} \in \mathbb{R}^{d_w \times 2d}$ and $W_{b2} \in \mathbb{R}^{r \times d_w}$ are the weight metrics; $b_{b1} \in \mathbb{R}^{d_w}$ and $b_{b2} \in \mathbb{R}^r$ are the bias, d represents the number of hidden units of LSTM, r is the number of attention heads. The weighted sum of hidden state based on attention score weights α_b are used to generate final representation s_b. The outputs of two components are added together and passed through the fully-connected layer to generate final sentence embedding as shown in Fig. 4.5 and given in equation 3.8.

$$s = W(s_a + s_b) + b \tag{3.8}$$

where W and b are the weight matrix and bias for a fully-connected layer.

In order to produce semantically meaningful sentence embeddings, we train the argument similarity model on the STS benchmark dataset. The model takes sentence-pair as input. For each sentence

of sentence-pair, the model calculates sentence embedding with the BERT encoder part and BiLSTM part. The sentence embeddings generated by each part are added together and passed through a fully connected layer. The final embeddings of two sentences are compared using cosine similarity. The model minimizes the mean squared error between the predicted cosine similarity score and the labeled similarity score. We choose the STS dataset for training the model as it fits best for argument similarity task of determining the similarity between two sentences. At inference time, the model generates vector representations for user utterance and possible arguments. Cosine distance between these vectors is then calculated and closest argument vector to a user utterance vector is identified as a reference argument.

3.4 Data Collection

In order to evaluate our NLU framework, we collected natural language utterances labeled with intent and (if needed) reference argument in an online survey. To this end, participants were asked to paraphrase possible (pre-defined) user utterances occurring in the interaction with BEA. After emphasizing to formulate all answers in their own words and showing four examples, the survey was conducted by showing three random arguments to the participants and asking them to reformulate different requests for each speech act. In case of *prefer*, *reject*, and *why* it was clearly specified which argument the subjects should refer to. To reduce a potential bias the formulation of the instruction was altered for each speech act and participant. For instance, an instruction was: "Please formulate that you agree with the argument, that 'nuclear weapons had become a source of extreme risk'[6]". To ensure the quality and validate the answers, copy-pasting and skipping answers were not allowed. Furthermore, we required at least five words in each response, which referred to an argument. To make sure that all participants were paying attention to the instructions, a control question was added which asked the user to precisely repeat a sentence. The data collection was divided into two parts with separate user groups: In the first part, we conducted an anonymous survey via clickworker[7] with 200 native English speakers from the UK to collect data for training and testing. In the second part, the group of participants consisted of 15 Chinese Master and Ph.D. students, to test whether there is a measurable effect if the interaction is conducted with non-native speakers. As we aim to evaluate our framework especially with regard to cross-domain applicability, we generated samples on three different topics. Two out of three arguments were taken from an annotated debate on the topic *Marriage is an outdated institution* from the *Debatabase* of the idebate.org[8] website [142]. The other remaining argument was sampled from the IBM corpus on claim and evidence detection [143] for one of the two topics *All nations have a right to nuclear weapons* and *The sale of violent video games to minors*. The data collection resulted in 1616 valid user responses for the first group and 143 responses from the second group.

[6] Copyright IBM 2014. Released under CC-BY-SA.

[7] https://marketplace.clickworker.com (last accessed 06 May 2020)

[8] https://idebate.org/debatabase (last accessed 09 January 2018)

3.5 Experimental Setup

In this section, we define the experimental setup for evaluating the proposed NLU approach. We evaluate the proposed model with respect to four different categories: The first one evaluates intent classification and argument similarity models separately against suitable baselines on different dataset described below:

User-Study: In Section 3.4, we have discussed the data collection of the User Study dataset. For the experiments, we divided the dataset into train and test sets. We train the intent classifier model on the two topics i.e., *All nations have a right to nuclear weapons* and *The sale of violent video games to minors*. The model is then evaluated on a large test set concerned with *marriage is an outdated institution*. We adopt this train and test data split to get an estimate of the model's robustness against topic changes. The statistics of train and test samples for each speech act are given in table 3.2 and table 3.3 provides a sample example of each speech act.

BANKING77: The dataset of Coope et al. [144], dubbed BANKING77, is composed of 13,083 customer service queries annotated with 77 intents. The dataset is divided into train and test sets. The test set contains 3080 examples and the full training set contains 10003 examples.

STS benchmark (STSb): The STS benchmark [126] dataset is a popular dataset for training and evaluating textual similarity task. It comprises 8628 sentence pairs from three categories captions, news, and forums. The dataset is divided into train-set (5749), valid-set (1500), and test-set(1379). We train the argument similarity model on the STSb training set and evaluate its performance on the user study dataset and STSb test dataset by computing cosine-similarity between the sentence embeddings. Furthermore, the user study test dataset comprises 2028 sentence pairs for 501 user utterances. For the user study dataset at prediction time, the model generates vector representations for user utterance and possible arguments. Cosine distance between these vectors is then calculated and closest argument vector to a user utterance vector is identified as a reference argument.

In the second evaluation category, we look at a few-shot intent classification scenario where only 10, 20, or 30 training examples are sampled for each intent from full training data to get an estimate of the model's performance when small training data is available. It is noteworthy that our argument similarity model is trained on STSb and doesn't require task-specific training data whereas the intent classifier model needs task-specific training data to learn the required system-specific intents. Due to this reason, we check few-shot setups for the intent classification task where few task-specific training data is available.

In the third evaluation category, we train and evaluate the complete pipeline (intent classifier and argument similarity) on separate topics of the user study dataset to assess the robustness of the model against topic changes.

The fourth evaluation category compares the results achieved with utterances from native speakers against results achieved with utterances from non-native speakers with a different cultural background to get an estimate of the model's sensitivity towards language proficiency.

表 3.2: Train and test examples statistics for each speech act of user study dataset. Table taken from our paper [1] with permission from Elsevier.

Speech Act	Train	Test (UK)	Test (China)
Exit	72	73	16
Level-up	72	73	16
Stance	71	74	15
Why	189	203	32
Prefer	79	305	33
Reject	110	304	31

表 3.3: Example utterances with annotated labels from user study dataset. The table referred from our paper [1]. Reprinted with permission from Elsevier.

Utterance	**Label**
What is my stance right now?	stance move
I would like to finish.	exit move
Please return to the previous argument.	level up move
Please tell me more why marriage promotes better way to raise child.	why move
I think marriage is good way to raise children	prefer move
I reject argument about marriage is an unreasonable expectation	reject move

3.5.1 Training Setup

For the intent classification, we employ the Bert-Base model[9] with 12 Transformer layers, 768 hidden states, and 12 self-attention heads. The size of the hidden units in uni-direction LSTM is 512 and the number of attention head r is 5. Furthermore, we use Adam optimizer with default values of $\beta_1 = 0.9$ and $\beta_2 = 0.99$, and a learning rate of $1e-4$ and $2e-5$ for training the BiLSTM and fine-tuning whole model respectively. Each update is computed through a batch size of 8 or 16 training examples and the number of epochs per batch are 32, 25, 16, and 8 epochs for 10-shot, 20-shot, 30-shot, and full-data settings, respectively. We apply the dropout as a regularization technique for our model to avoid over-fitting. We set the dropout rate as 0.1 for all dropout layers. We employ the transformers [145] library to train our intent model.

For training argument similarity, we used Adam optimizer with a learning rate of $2e-5$ and a batch size of 16 training samples. Furthermore, the model uses the pre-computed 300-dimensional word embeddings ConceptNet Numberbatch. The number of hidden units in uni-direction LSTM is 512 and the number of attention heads is 5. The model is trained for 8 epochs.

3.5.2 Evaluation Metrics

The evaluation metric used for intent classification is the accuracy metric. For the argument similarity task on the user study dataset, the model performance is measured by the accuracy of identifying user reference arguments. As suggested in [146], we use Spearman correlation for the semantic textual similarity (STS) task. The Spearman's rank correlation is computed between the cosine-similarity of sentence embeddings and gold labels for the STS dataset.

Accuracy: Accuracy calculates the number of intent/argument correctly predicted by the model for all intent/argument test samples, as given in the following equation.

$$Accuracy = \frac{\#true\_positives + \#true\_negatives}{\#true\_positives + \#false\_positives + \#true\_negatives + \#false\_negatives} \tag{3.9}$$

3.5.3 Sequential Training

The proposed approach consists of two sub-models namely: intent classification and argument similarity. We trained these models in a sequential manner. We first train the argument similarity model on the STS benchmark dataset. The detailed training steps are presented in algorithm 3.1. After training the argument similarity model, its weights are fixed. The sentence representation obtained from the argument similarity model is then given as input to a fully connected layer of intent classifier model. The proposed approach then trains the intent classifier model; its training is divided into two stages. In the first stage, we freeze the BERT encoder parameters and only train the task-specific BiLSTM and fully connected layer for four epochs. In the second stage, we unfreeze all BERT encoder parameters and fine-tune all parameters of BERT encoder as well as BiLSTM, and fully connected

[9] https://storage.googleapis.com/bert_models/2020_02_20/uncased_L-12_H-768_A-12.zip

layer in an end-to-end manner. We adopt this training strategy to train the model with different learning rates for different epochs that helps the model to retrain the pre-trained knowledge of the BERT and avoid catastrophic forgetting of this knowledge during fine-tuning [90, 147]. Detail training steps are presented in algorithm 3.2.

算法 3.1 Training procedure of argument similarity model. Adopted from our paper [1].

Input: Training data $D = \{x_n, y_n\}_{n=1}^{N}$

Load pre-trained BERT model parameters θ_1

Load pre-trained ConceptNet Embedding

Initialize BiLSTM parameters θ_2

for *each epoch* **do**

  Sample mini-batch $(x, y) \subset D$

  Construct argument similarity model with Eq 3.8

  Train argument similarity model with a learning rate of $2e-5$

  Update parameters θ_1, θ_2

Output: Model parameters

算法 3.2 Training procedure of intent classifier model. Adopted from our paper [1].

Input: Training data $D = \{x_n, y_n\}_{n=1}^{N}$

Load pre-train BERT parameters θ_3

Initialize BiLSTM parameters θ_4

Freeze parameters θ_3

for *each epoch* **do**

  Sample mini-batch $(x, y) \subset D$

  Construct intent classifier model with Eq 3.4

  Train intent classifier model with a learning rate of $1e-4$

  Update parameters θ_4

Unfreeze parameters θ_3 **for** *each epoch* **do**

  Sample mini-batch $(x, y) \subset D$

  Load trained intent classifier model

  Continue train intent classifier model with a learning rate of $2e-5$

  Update parameters θ_3, θ_4

Output: Model parameters

3.6 Evaluation and Results

3.6.1 Evaluation –Intent classification

We compare the performance of the proposed intent classifier model against the following baseline methods.

1. Embedding Classifier: The embedding intent classifier model from Rasa[10] NLU inspired by StarSpace [148], counts distinct words of the training data and provides these word token counts as input features to the intent classifier. We trained this model using the Rasa framework for 300 epochs.

2. Logistic Regression with BERT (LR + BERT): The model uses features obtained from the pre-trained Bert model. The BERT model processes the user utterances and the final hidden state of the `[CLS]` token of each utterance is passed as features to the logistic regression model. The model is then trained on these extracted features to predict the user intent.

3. Dual Intent and Entity Transformer (DIET) Classifier: The DIET [78] model is multi-task architecture for intent classification and entity recognition. The model obtains dense features from pre-trained word embedding models. These features are then used by 2 layer transformer with relative position attention. The BERT-base model is employed for producing dense features. The model is trained using the Rasa framework for 100 epochs.

4. BERT Classifier: In the pre-trained BERT model [43], we add a fully-connected layer on top of the last encoder layer `[CLS]` token for classifying user intent. The model is fine-tuned for 8 epochs with a batch size of 16 and a learning rate of $2e-5$.

5. DistilBERT Classifier: The DistilBERT model [149] utilizes knowledge distillation during pre-training to reduce the size of the BERT model. We added one fully-connected layer on top of the final encoder `[CLS]` token for predicting user intent. The model is fine-tuned using Adam optimizer with a learning rate of $2e-5$ and a batch size of 16 for 8 epochs.

6. RoBERTa Classifier: The RoBERTa model [150] optimizes the BERT pre-training approach by employing dynamic masking, large mini-batches, and a larger byte-level byte-pair encoding (BPE) for training the robust model. The fully connected layer is applied on top of `[CLS]` token of the final encoder layer for predicting user intent. The model is fine-tuned with Adam optimizer using a learning rate of $2e-5$ and a batch size of 16 for 8 epochs.

[10] https://rasa.com/

表 3.4: Intent classifier performance comparison on Users Study and Banking77 datasets with the different number of training examples i.e., 10-shot (10 training examples per intent), 30-shot (30 training examples per intent), and full training data. Performance is reported in accuracy scores × 100. First published in our paper [1]. Reprinted with permission from Elsevier.

	User Study				Banking77			
Model	10-shot	20-shot	30-shot	Full	10-shot	20-shot	30-shot	Full
Embedding	58.9	67.5	70.3	73.1	59.0	72.2	74.8	86.2
LR+BERT	55.9	68.9	75.1	76.3	56.2	69.0	75.1	86.9
DIET	51.3	69.4	75.6	82.3	55.8	76.4	82.6	90.3
DistilBERT	65.9	80.9	83.4	88.2	79.1	85.5	88.1	92.9
BERT-tuned	71.5	82.1	85.1	89.3	82.0	86.7	88.5	93.2
RoBERTa Classifier	63.5	82.0	85.0	89.3	76.5	87.4	88.6	93.2
Albert Classifier	63.6	83.4	85.2	89.2	75.9	86.5	88.2	92.3
BERT+BiLSTM +ArgSim	**74.6**	**85.0**	**87.3**	**89.7**	**83.5**	**88.1**	**90.2**	**93.9**

7. Albert Classifier: The Albert model [151] improves the BERT model by incorporating factorized embedding parameterization and cross-layer parameter sharing techniques. The fully-connected layer is placed on top of the last encoder layer. The model is fine-tuned using a learning rate of $2e-5$ with the Adam optimizer for 8 epochs.

Intent classification results are presented in Table 3.4 (statistically significant with $p < 0.05$). The results demonstrate that the embedding classifier performs poorly in 30-shot and full data against all baseline approaches because it utilized merely the word counts and does not consider pre-trained language model information. In contrast, LR + BERT model obtained better results than the embedding classifier, because it extracts features from the pre-trained language model for predicting user intent. Furthermore, the DIET classifier employed 2 Transformer encoder layers to learn the contextualized sentence representation and outperformed the former models. The only case where the DIET classifier performs poorly than the embedding classifier is the 10-shot case. The reason behind this is the lack of training data as the pre-trained language model employed by the DIET model requires more training data to generalize well. Besides, DistilBERT, BERT, RoBERTa and Albert models achieved superior results than the DIET classifier, because these models employ 6, 12, and 12 Transformer encoder layers respectively, as compared to 2 Transformer encoder layers used by the DIET classifier, therefore, these models provide better and robust utterance representation. We observe the marginal differences in the performance of the BERT, Albert and RoBERTa classifiers. Furthermore, Albert and RoBERTa classifier perform better than DistilBERT in most cases. The only setup where DistilBERT performs better than Albert and RoBERTa is a 10-shot setup. This indicates that RoBERTa requires a large amount of data to generalize well. Nevertheless, our proposed model outperforms all baseline methods and performance increases by approximately 16%, 13%, and 7% better accuracy score compared to embedding classifier, LR + BERT, and DIET classifier, respectively on the full

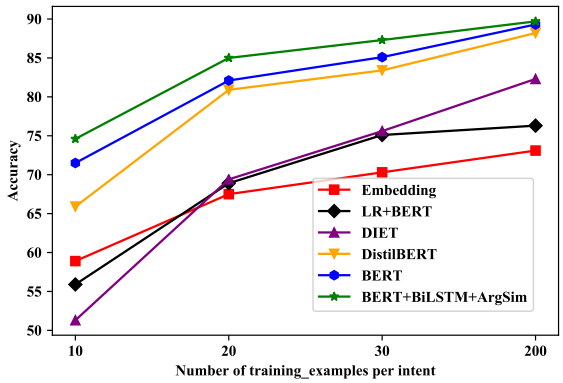


图 3.3: Performance comparison of the intent classifier on the User study dataset with respect to the number of training examples per intent. Plot referred from our paper [1]. Reprinted with permission from Elsevier.

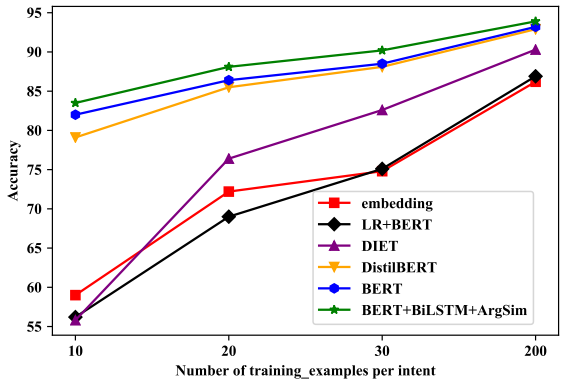


图 3.4: Performance comparison of the intent classifier on the Banking77 dataset with respect to the number of training examples per intent. Plot referred from our paper [1]. Reprinted with permission from Elsevier.

data setup of the user study dataset. Furthermore, the improvement over embedding classifier, LR + BERT, and DIET classifier are 7.7%, 7%, and 3.7% on the full data setup of Banking77 dataset. This is because the proposed BERT+BiLSTM+ArgSim model fine-tuned the BERT model to obtain contextual utterance representation for predicting user intent. Additionally, we observe that stacking BiLSTM on top of the BERT model and concatenating sentence representation from the argument similarity model provides better results than stacking just a single fully-connected layer.

From Fig. 3.3 and Fig. 3.4, we can observe that in the few-shot scenario i.e., 10-shot, 20-shot, and 30-shot; our BERT + BiLSTM+ArgSim model performance gains are more prominent over baselines models. The improvement gains of our model over the state-of-the-arts DistilBERT, RoBERTa, and BERT classifier models are almost 9%, 11%, and 3% for the user study dataset and 4%, 7%, and 1.5% for the Banking77 dataset in a 10-shot setup (10 samples per intent). Similarly, we see the proposed model significantly outperforms all model in the 20-shot and 30-shot setup on both datasets. Overall, our model has a clear advantage over other methods in all setups, and more prominently in few-shot setups. Furthermore, the proposed model is statistically significant as compared to Embedding, LR+BERT, DIET, DistilBERT, RoBERTa, and BERT models with p-value of $1e-05$, $2e-05$, 0.0019, 0.023, 0.044, and 0.049 respectively, on user study dataset.

3.6.2 Evaluation –Argument Similarity

We evaluate the performance of the proposed argument similarity model for identifying the reference argument task and also validate our model performance for the common semantic textual similarity task. We perform experiments on user study and semantic textual similarity benchmark dataset. We compared the results against the following baseline methods.

1. Average of glove embeddings;
2. Average of ConceptNet Numberbatch embeddings;

3. Mean of last layer tokens representations of BERT;

4. InferSent model [50] is trained on a natural language inference dataset using the siamese BiLSTM network structure with max-pooling over the output.

5. Universal Sentence Encoder [51] is a strong sentence-level embedding model trained using transformer architecture and multiple objectives.

6. Sentence-BERT model [45] a state-of-the-art sentence embedding model is trained using a siamese network structure over BERT.

7. Sentence-RoBERTa model [45] SRoBERTa fine-tunes the RoBERTa pre-trained model to produce robust sentence embeddings which can be compared using cosine similarity. We fine-tune the RoBERTa network on the STSB dataset with Adam optimizer using a learning rate of $2e-5$. We report experiments using the mean pooling strategy.

8. Sentence-XLM-Align model fine-tunes cross-lingual language models word alignment (XLM-Align) pre-trained model [152] to generate useful sentence embeddings. We fine-tune the pre-trained XLM-Align model on the STSB dataset using Adam optimizer with a learning rate of $2e-5$. The mean pooling strategy is applied to produce fixed-sized sentence embeddings.

9. Albert model [151] improves the BERT model by incorporating factorized embedding parameterization and cross-layer parameter sharing techniques. The model is fine-tuned on the STSB dataset using a learning rate of $2e-5$ with the Adam optimizer. The results are reported using the mean pooling strategy.

Table 3.5 presents the results corresponding to the argument similarity task. The results generated by our model significantly outperform InferSent, SXLM-Align, and SBERT by achieving approximately 4%, 2%, and 2% better accuracy on the user study dataset, respectively. The SXLM-Align model produces worse results than SBERT and SRoBERTa on both the user study dataset and STSB dataset. On the other hand, the SRoBERTa model performs slightly better than the SBERT model on both datasets. Albert model performs better on STSB datasets but produces average results on the user study dataset. Also, USE performs better than InferSent and SBERT on the user study dataset, as it is pre-trained on question answering data in addition to NLI data, which is related to the classification task. Our model performance matches the performance of the state-of-the-art USE model on the user study dataset. Furthermore, the mean of last layer tokens representations of BERT embeddings, average glove embeddings, and an average of ConceptNetNumberbatch embeddings perform poorly on the STSB dataset. However, for the user study dataset, these methods produced better results than the supervised InferSent model trained using the siamese structure on NLI data. The reason behind this is that for identifying the reference argument task, we calculate the cosine-similarity between candidate arguments and current utterance, and the closest argument to the current utterance embedding is selected as a reference argument. This allows two-sentence embeddings to have high

表 3.5: Argument similarity model performance comparison on User Study and STS datasets. SBERT-STSb-base, SBERT-STSb-large, and ArgSim models are trained on the STS-B dataset. Performance on user study is reported in the accuracy scores × 100, and performance on STS is reported in SPEARMAN × 100. Table referred from our paper [1]. Reprinted with permission from Elsevier.

Model	User Study Accuracy	STSB SPEARMAN
Avg. GloVe	93.2	61.5
Avg. ConceptNet	94.0	65.1
Avg. BERT	93.2	47.2
InferSent - Glove	90.2	75.8
USE	**95.2**	78.2
SBERT-STSb-base	94.0	84.6
SBERT-STSb-large	94.0	84.4
SRoBERTa-STSb-base	94.4	84.8
SXLM-Align-STSb-base	93.8	80.5
Albert-STSb-base	94.1	79.7
ArgSim (Ours work)	**95.2**	**85.1**

and low similarity on certain dimensions and still correctly identify a reference argument. In contrast, the STS task is a regression task, which estimates the similarities between two-sentence embeddings by cosine-similarity and treats all dimensions equally. This indicates average word embeddings are infeasible for the STS task. Nonetheless, our model trained on the STS benchmark yields better sentence representation for argument similarity task and STS task as it combines contextualized word representation with common sense knowledge obtained from ConceptNet Numberbatch embeddings.

3.6.3 Evaluation- Complete Pipeline

We evaluate the performance of the complete pipeline consists of intent classification and argument similarity modules for native English speakers and non-native English speakers. The performance of intent classification and argument similarity module is measured in F1 score and accuracy, respectively. The complete pipeline performance is measured in an accuracy matrix. The overall accuracy of the complete pipeline is the percentage of utterances where the pipeline correctly predicts both intents and presented arguments. The model is trained on the native speakers dataset and evaluated on native and non-native speakers test-set. The statistics of train and test samples are given in table 3.2. The results are shown in table 3.6. We can observe from the results that both intent classifier and argument similarity models perform better on the native speaker dataset. The respective improvements are around 1% and 5% for intent classifier and argument similarity models. However, we do not observe a significant difference in accuracy between native speakers and non-native speakers on the complete pipeline. Overall, this proves that the proposed framework is robust to the different

表 3.6: Complete pipeline performance of the proposed framework on native speakers and Non-native speakers datasets. Table taken from our paper [1]. Reprinted with permission from Elsevier.

Model	Intent F1	Sim Acc.	Overall Acc.
Native Speakers	89.8	95.2	87.7
Non-native Speakers	88.8	89.7	87.3

language proficiency of the users.

3.6.4 Impact of hyper-parameters

The learning rate, batch size, and hidden size are the most important hyper-parameters of the proposed model. The model performance is analyzed concerning these parameters. We choose learning rate from $[0.001, 0.0001, 1e-5, 2e-5, 5e-5]$. In Fig. 3.5 we can see that the learning rate has a significant impact on model performance for both intent classification and argument similarity tasks. Furthermore, we observe that the proposed model performs poorly on learning rates of 0.001 and 0.0001, which shows that the model fails to converge on high learning rates. In most cases, a lower learning rate produces better results. Especially when the learning rate is $2e-05$ model achieves the highest accuracy on the user study dataset for both tasks. Furthermore, we choose a batch size from a range of $[4, 8, 6, 32, 64, 128]$. Fig. 3.6 reveals that change in batch size has less impact on model performance for argument similarity task and full data setup of intent classification. The model produces a higher accuracy on batch sizes of 16 and 32 for the full data setup of intent classification. However, batch size has a significant impact on model performance for the 10-shot intent classification setup. The model performance increases with a small batch size of 4 and 8. Especially when batch size is 8, the model achieves the highest accuracy of 74.6%. As the value of batch size increases, the number of mini-batches decreases, and model performance decreases. We explore the following values $h = [4, 8, 6, 32, 64, 128]$ for hidden dimensions of LSTM. From Fig. 3.7 we can observe that the performance of the model does not change significantly for different sizes of hidden dimensions of LSTM. Results suggest that the proposed model is robust to hidden dimensions of LSTM for intent classification and argument similarity tasks. Furthermore, a lower learning rate and smaller batch size yield better results especially for the 10-shot intent classification setup.

3.6.5 Ablation Study

To demonstrate the effectiveness of different aspects of the intent classifier model, we conduct an ablation study on the two datasets. The results are shown in table 3.7. It shows that adding BiLSTM and SimBERT representation improves the performance of the plain fine-tuned BERT model. Adding BiLSTM provides performance improvement of 1.3%, 1.1%, 0.6%, and 0.2% for 10-shot, 20-shot, 30-shot, and full-data respectively on user study dataset, and 0.6%, 0.7%, 0.7%, and 0.2% improvement on the Banking77 dataset. Furthermore, adding representation from the argument similarity

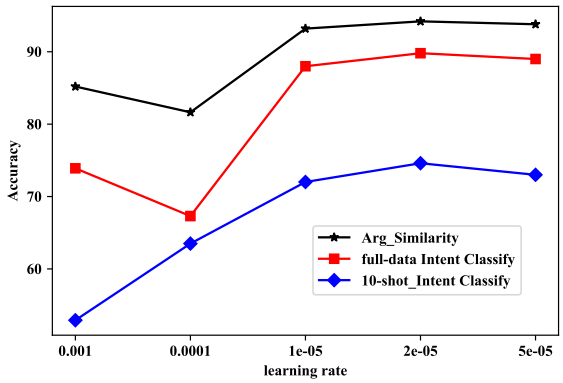


图 3.5: Effect of learning rate on model performance for intent classification and argument similarity tasks on User study data set. Plot adopted from our paper [1]. Reprinted with permission from Elsevier.

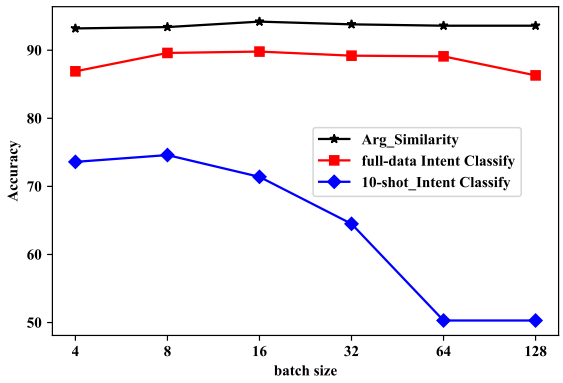


图 3.6: Effect of batch size on model performance for intent classification and argument similarity tasks on User study data set. Plot taken from our paper [1]. Reprinted with permission from Elsevier.

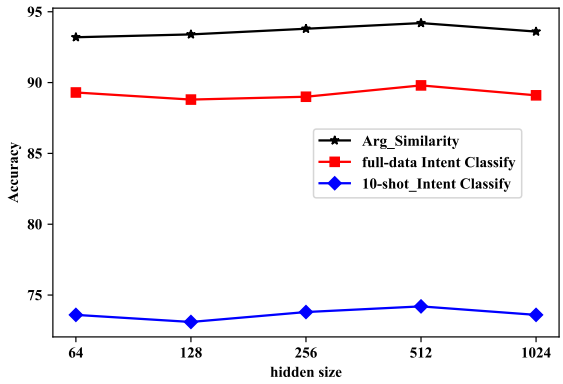


图 3.7: Effect of hidden size on model performance for intent classification and argument similarity tasks on User study data set. Plot referred from our paper [1]. Reprinted with permission from Elsevier.

model further improves the performance of the proposed model, especially in the 10-shot, 20-shot, and 30-shot scenarios, and corresponding improvements on the user study dataset are 1.8%, 1.8%, and 1.6% respectively. The overall performance gains over BERT-tuned are 3.1%, 2.9%, 2.2%, and 0.4% respectively for the user study dataset, and 1.5%, 1.4%, 1.2% and, 0.7% improvement for the Banking77 dataset.

We start the ablation of the argument similarity (ArgSim) model by removing different components in our model to get an understanding of their importance. We first remove the BERT part (-BERT) and then remove the BiLSTM part (-BiLSTM). Table 3.8 shows the ablation results on the user study and STS dataset. From the results, we observe that the BERT part plays a significant role in model performance, and removing the BERT part decreases the model performance by around 2% in terms of accuracy on the user study dataset and 14.1% in terms of spearman correlation on the STSB

表 3.7: Evaluation results of intent classifier ablation on the Users Study and Banking77 datasets for 10-shot, 20-shot, 30-shot, and full training data setups. The table referred from our paper [1]. Reprinted with permission from Elsevier.

	User Study				Banking77			
Model	10-shot	20-shot	30-shot	Full-data	10-shot	20-shot	30-shot	Full-data
BERT-tuned	71.5	82.1	85.1	89.3	82.0	86.7	88.5	93.2
BERT+BiLSTM	72.8	83.2	85.7	89.5	82.6	87.4	89.2	93.4
BERT+BiLSTM + ArgSim	**74.6**	**85.0**	**87.3**	**89.7**	**83.5**	**88.1**	**90.2**	**93.9**

表 3.8: Evaluation results of argument similarity model ablation on User Study and STSB datasets. Table referred from our paper [1]. Reprinted with permission from Elsevier.

Model	User Study Accuracy	STSB SPEARMAN
- BERT	93.2	71.0
- BiLSTM	94.0	84.0
ArgSim	**95.2**	**85.1**

dataset. Next, we show that the BiLSTM part is also an important contributor to model performance, and removing the BiLSTM part decreases the model performance by around 1.2% in terms of accuracy on user study dataset and 1.1% in terms of spearman correlation on the STSB dataset. These results indicate both BERT and LSTM parts are important for the argument similarity model.

3.7 Discussion and Summary

Throughout this chapter, we introduced an NLU approach for argumentative dialogue systems in the domain of information seeking and opinion building. Our approach detects arguments addressed by the user within his or her utterance and distinguishes between multiple intents including user preferences towards the respective arguments. Our approach was applied and tested in an actual argumentative dialogue system on data collected in an extensive user study. Additionally, we evaluated the proposed intent classifier and argument similarity models on the Banking77 and STS benchmark datasets. Throughout the evaluation, we assessed the performance of the NLU components against state-of-the-art baselines on different datasets, the robustness of the proposed approach against new topics, and the robustness of the approach against different language proficiency and cultural diversity. Besides, the performance of the intent classifier model is assessed in full data and few-shot setups. Our results show a clear advantage of our model against baselines approaches for intent classification in both full data and few-shot setups. Furthermore, results show the superior accuracy of

the proposed model against baselines models for argument similarity tasks as well as the accuracy of 87.7% in complete pipeline testing. Moreover, no significant difference between utterances from UK users and Chinese users was detected. The results indicate that our model has to be trained only once for each system in order to learn the required system-specific intents but does not require pre-training for new topics or user groups which ensures high flexibility of the respective system.

第四章 Single turn Intent Detection and Slot Filling

4.1 Introduction

In this chapter, we present a neural model for single turn intent detection and slot filling as introduced in our publication [2].

Task-oriented dialogue systems interact with users via natural language to perform specific tasks, such as send an email, find a restaurant, or navigate to a particular location. The natural language understanding (NLU) module is the main component of these systems. The NLU module extracts the semantic representations from natural language sentences. Intent detection and slot filling are key tasks in the NLU module [153]. Intent detection is framed as a sentence classification task that classifies the intent of the user. Slot filling is viewed as a sequence tagging task, it tags the slots related to semantic frames. For example, "show me the flights from dallas to san francisco" as depicted in Fig. 1. In the example, the user's intent is to find a flight, "dallas'' is origin city slot value and "san francisco" is the destination city slot value. The words which do not belong to any slot are assigned the null "O" label. The dialogue manager utilizes results generated by intent detection and slot filling tasks to provide a response to the user.

Deep neural models have achieved impressive performance for intent detection and slot filling tasks [40, 41, 77, 154]. These models can learn the robust word and sentence representations from training data. However, training of these models requires a large number of labeled examples. Therefore, scaling neural network models to support new intent and slots is a challenging and resource-intensive process [34, 155]. The pre-trained language models trained on large-scale unlabeled corpora such as Bidirectional Encoder Representations from Transformers (BERT) [43], robustly optimized BERT [156] have reduced the bottleneck of scarce in-domain data. However, directly applying the BERT model to intent detection and slot filling may be sub-optimal. Fine-tuning BERT still needs a sufficient number of training examples as it expects the adaptation to a full large model [157]. Furthermore, fine-tuning BERT on small in-domain data may result in overfitting. In Contrast, rule-based models build on handcrafted rules like regular expressions do not need labeled data and often obtain decent prediction accuracy. Regular expressions (REs) are widely used for tasks like pattern matching [24], entity recognition [25], and information extraction [26]. These models can be utilized when limited or no training examples are available. However, RE-based systems do not benefit from labeled data when available and generalize poorly on a large dataset containing a lot of synonyms and variations.

Combining the advantages of rule-based systems and neural networks is an active research topic. One line of research is to incorporate task-specific knowledge into the neural networks by imposing soft constraints on neural network output using posterior regularization [107, 158], and by minimizing a joint loss of multi-task model [108, 109], or by injecting symbolic knowledge via semantic loss

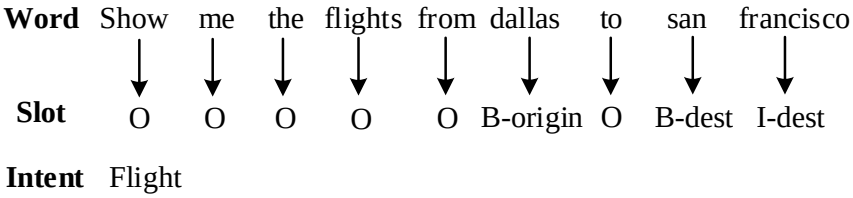


图 4.1: Illustration of example utterance contains intent and slot annotation using IOB format. This figure is referred from our paper [2].)

function [110]. Another line of research is to employ regular expressions for encoding knowledge at different levels inside neural networks [102], or by directly converting regular expressions into neural networks [106]. Although these models reduce the need for large training data, these models fail to produce state-of-the-art results in both limited data and rich data scenarios.

Based on the above intuitions, this work propose a WFST-BERT model, which integrates WFST with BERT for joint intent detection and slot filling tasks. The WFST-BERT combines the strength of large-scale pre-trained models and rule-based models. The WFST encodes domain knowledge into the model by converting RE rules into the trainable neural model. Furthermore, the model leverage pre-trained BERT for incorporating contextual information. The WFST-BERT trained simultaneously on labeled data using a gradient descent algorithm to improve model prediction performance over the original RE. Furthermore, our experiments find that the proposed model produces significant improvements in both a limited data setting and a full-data setting.

Our key contributions of this work are given as follows:

- We propose a WFST-BERT model for intent detection and slot filling, which reduces the need for large labeled data by generating robust utterance representation through a pre-trained language model and encoding domain knowledge using finite automata.
- We show that the proposed model has a clear advantage in both a few-shot setup and a full-data setup.
- We perform extensive experiments on the ATIS and SNIPS datasets and compare the results of WFST-BERT against state-of-the-art counterparts using precision, recall, and F1 score evaluation metrics.

4.2 Model Architecture

The illustration of WFST-BERT architecture is given in Fig. 4.4. The WFST-BERT architecture consists of 2 main modules, the WFST module that encodes domain knowledge into the system, and the pre-trained BERT module which incorporates contextual information into the model. The model receives the user sentence as an input, which is passed through the FST layer and BERT transformer layers. The sentence representation s_1 containing information of matched REs is obtained from WFST. Furthermore, contextualize sentence representation s_2 is generated from the BERT module.

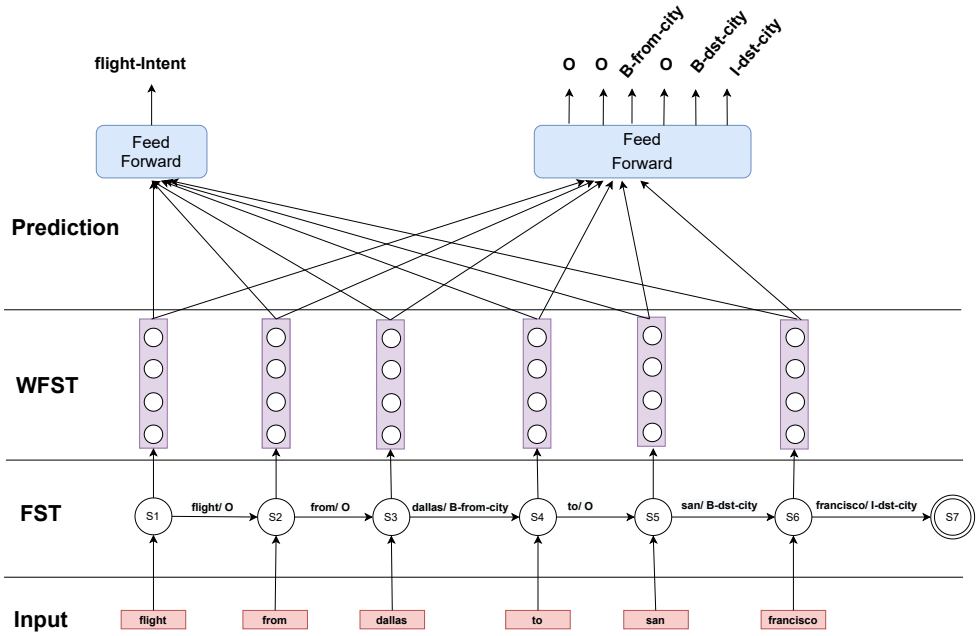


图 4.2: Illustration of WFST module for generating vector representation of intent and slots. The handwritten REs are converted into unweighted FST. The feed-forward layers is used to generate final representation from the state vectors of WFST containing REs matching information. Illustration adopted from our paper [2]. Reprinted with permission from Springer.

Our method combines s_1 and s_2 sentence representations to produce a final sentence representation that predicts user intent and slot labels. The detailed working process of the WFST and BERT modules is provided in the subsequent subsections.

4.2.1 Weighted Finite State Transducer (WFST)

From REs to Trainable WFST

As discussed in section 2.4.2, An FA can be constructed for each RE using Thompson's construction algorithm [103]. For every RE there exists a unique DFA with the minimum number of states. The REs are first converted into NFAs and then the subset construction algorithm [159] is applied to them for obtaining DFAs. Furthermore, the model obtains minimize DFA by running the minimization algorithm [160]. Specifically, our method starts by writing a set of REs $R = \{r_1, r_2, \cdots, r_n\}$. Each RE corresponds to some intent class and slot label at the sentence and word level, respectively. In particular, each RE r_i is converted into unweighted FST with Q_i states, initial state q_0, and final state q_n. The FST is turned into WFST by assigning 0/1 weights at each transition, initial state, and final state. $T(x, q_a, q_b)$ is 1 if WFST transits from state $q_a \in Q$ to state $q_b \in Q$ on receiving input $x \in \Sigma$ and 0 otherwise. $\lambda(q_a) = 1$ if q_a is initial state 0 otherwise. Similarly $\rho(q_a) = 1$ if q_a is final state. We combine these individual WFSTs into a single WFST having total states $Q = \sum_i Q_i$ and multiple start and end states. We then run this WFST on sentence x to get REs matching results. The feed-forward layers are applied to matching results to produce intent and slots representations. The complete procedure is illustrated in Fig. 4.2.

Let $h_t \in \mathbb{R}^{|Q|}$ be the score vector after reading t words of input sentence x. The Viterbi score that uses the max-product semiring can be formulated using a recurrent form, as follows:

$$\begin{aligned} h_0 &= \lambda, \\ h_t &= h_{t-1}.T[x_t], \\ S(\mathcal{T}, x) &= h_N \cdot \rho \end{aligned} \tag{4.1}$$

where λ and ρ are the initial and final weights of WFST respectively. Moreover, $T \in \mathbb{R}^{|\Sigma| \times |Q| \times |Q|}$ is the weighted transition tensor.

As suggested in [106], we decompose T into three matrices $E_R \in \mathbb{R}^{|\Sigma| \times r}, D_1 \in \mathbb{R}^{|Q| \times r}, D_2 \in \mathbb{R}^{|Q| \times r}$ for reducing the number of parameters where r is any arbitrary number. The recurrent form of h_t in Eq. 4.1 can be written as follows:

$$\begin{aligned} c &= (h_{t-1} \cdot D_1) \odot x_t \\ h_t &= c \cdot D_2^T \end{aligned} \tag{4.2}$$

where $\odot$ represents element-wise product. Now WFST is parameterized by $\theta = (E_R, D_1, D_2, \lambda, \rho)$.

Prediction of Intent and Slots via WFST

To predict intent class and slots label, we run WFST on sentence x that returns the output vectors h_t and final output vector h_N containing information of all matching REs. The feed-forward layer is applied to the final output vector h_N to generate intent representation.

$$I^{FA} = (W_{in}^{FA}(h_N \cdot \rho) + b_{in}^{FA}) \tag{4.3}$$

where $W_{in}^{FA} \in \mathbb{R}^{d \times |Q|}$ represents the weight matrix, $b_{in}^{FA} \in \mathbb{R}^{|I|}$ are the represents the bias vector, and d denotes dimension of intent representation vector.

To detect the tag sequence y for the input word sequence x, we apply another feed-forward layer on all output vectors h_t and defined as follow:

$$y_t^{FA} = (W_{slot}^{FA} h_t + b_{slot}^{FA}) \tag{4.4}$$

where h_t denotes output vector of word x_t, W_{slot}^{FA} is the weight matrix, and b_{slot}^{FA} is the bias vector.

4.2.2 BERT

The BERT architecture for joint intent detection and slot filling is depicted in Fig. 4.3. It composes of two parts bidirectional Transformer encoder and the Intent & Slot prediction part. Details of these parts are provided in subsequent subsections.

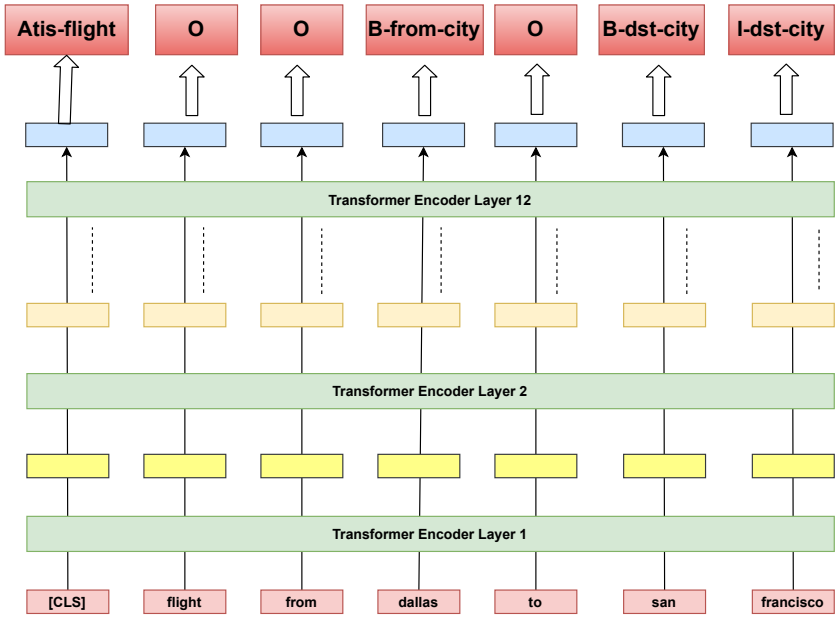


图 4.3: Illustration of the BERT model for joint intent detection and slot filling. This figure is taken from our paper [2] with permission from Springer

BERT Utterance Representation

The BERT encoder calculates the utterance representation i.e., vector representation for each token denoted as in equation 4.5:

$$H_t^n = BERT(x_{[CLS]}, x_1, \cdots, x_t, x_{[SEP]}) \tag{4.5}$$

where $H_t^n = (h_{[CLS]}^n, h_1^n, \cdots, h_t^n, h_{[SEP]}^n)$, h_t denotes the contextual vector representation of token t, and n represents the number of encoder layer. The last layer hidden state H_t^{12} of every token is given as input to fully connected layers for detecting user intent and corresponding slots.

Intent Detection and Slot Filling via BERT

To predict intent class, we apply a fully-connected layer on the first special token $h_{[CLS]}$ of final hidden state H_t^{12}, which is represented as follow:

$$I^B = W_{in}^B h_{[CLS]} + b_{in}^B \tag{4.6}$$

For the slot filling task, we feed all other tokens $h_1, \cdots, h_t$ of final hidden state H_t^{12} into another fully connected layer to generate slot representation.

$$y_t^B = W_{slot}^B h_t + b_{slot}^B \tag{4.7}$$

where h_t represents hidden state of first sub token of word x_t, W_{slot}^B denotes the weight matrix, and b_{slot}^B is the bias vector.

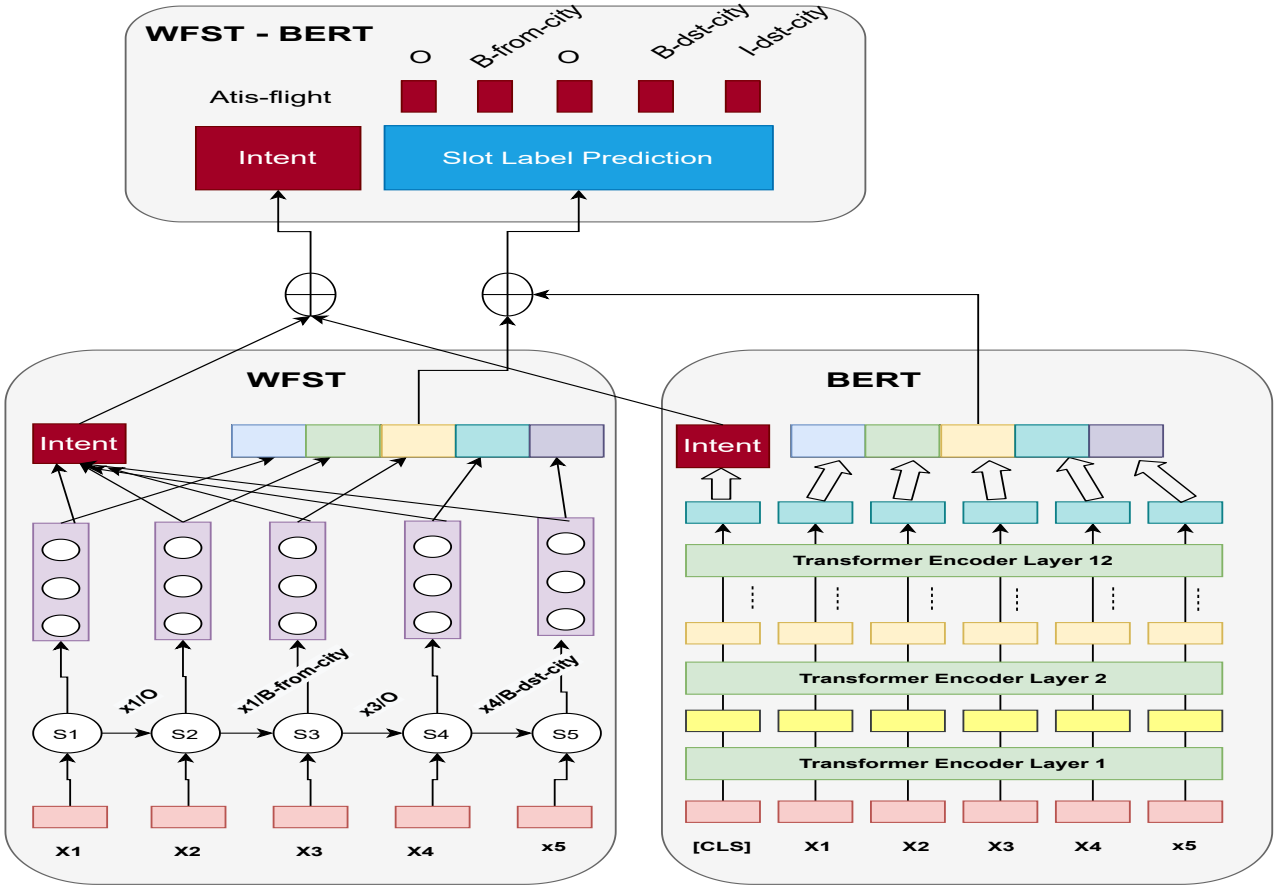


图 4.4: Illustration of the proposed WFST-BERT model for joint intent detection and slot filling. This figure is referred from our paper [2].)

4.2.3 WFST-BERT

Fig. 4.4 shows the structure of the proposed WFST-BERT model in detail. First, the model applies WFST on sentence x to generates intent representation I^{FA}. Besides, the model finetunes the BERT model for generating intent representation I^B from `[CLS]` of the last hidden layer. We then concatenate the intent representation from WFST containing information of matching REs and intent representation from BERT containing contextual information. The linear layer is applied to this final representation followed by the softmax activation function for producing intent labels.

$$\hat{I} = \text{softmax}(W_{in}(I^{FA} \oplus I^B) + b_{in}) \tag{4.8}$$

where $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}}$, $W_{in} \in \mathbb{R}^{2d \times |I|}$ represents weight matrix of fully connected layer, $b \in \mathbb{R}^{|I|}$ is the bias vector, $\oplus$ denotes the vector concatenation operator, and I represents the user intent vocabulary.

Similarly, for the slot filling task, we obtain y_t^{FA} from WFST and y_t^B from BERT last task-specific layer. We concatenate these word-level representations and passed through the fully connected layer. The IOB slot labels are produced by applying a softmax layer on top of the word level outputs y_t.

$$\hat{y_t} = \text{softmax}(W_{slot}(y_t^{FA} \oplus y_t^B) + b_{slot}) \tag{4.9}$$

Here W_{slot} and b_{slot} denote weights and bias of fully connected layer.

4.2.4 Joint Optimization

For joint training of both intent detection and slot filling tasks the objective function is formulated as:

$$\begin{aligned} &\mathcal{P}(y^{in}, y^{slot}|x) \\ &= \mathcal{P}(y^{in}|x_1, \cdots, x_t) \prod_{t=1}^{T} \mathcal{P}(y_t^{slot}|x_1, \cdots, x_t) \end{aligned} \tag{4.10}$$

where $\mathcal{P}(y^{in}, y^{slot}|x)$ is the conditional probability of detecting intent of user and corresponding slots for given input x. Furthermore, we use the cross-entropy loss for intent detection ($\mathcal{L}_{in}$) and slot filling ($\mathcal{L}_{slot}$) tasks defined as follows:

$$\begin{aligned} \mathcal{L}_{in} &= -\sum_{i=1}^{I} y_{in}^{i} log(\hat{y}_{in}^{i}) \\ \mathcal{L}_{slot} &= -\sum_{t} \sum_{s}^{S} y_t^s log(\hat{y_t}^s) \\ \mathcal{L} &= \mathcal{L}_{in} + \mathcal{L}_{slot}, \end{aligned} \tag{4.11}$$

where I, S represent the number of intent and slot labels, and y_{in}, y_t^n are the ground truth of intent and slot labels, respectively. The model minimizes the joint loss which is the sum of both individual losses.

4.3 Experiments

4.3.1 Datasets

We perform experiments on two datasets: ATIS [161] and SNIPS [162] datasets which are commonly used for NLU tasks like intent detection and slot filling. The ATIS dataset contains information regarding flight reservation requests. We used the ATIS data format as in [40], [163]. The training dataset consists of 4978 utterances and the test set consists of 893 utterances. The number of distinct intent labels is 18 and the number of slots labels is 127. The SNIPS dataset contains queries to voice assistants asking for commonly used tasks like playing music, booking restaurants, queries regarding weather. The SNIPS training dataset contains 13084 utterances and the test set contains 700 utterances. The number of distinct intent labels is 7 and the number of slot labels is 72.

4.3.2 Evaluation Metrics

To evaluate the results of the proposed model, We have utilized the most commonly used evaluation metrics such as precision, recall, and F1 score.

表 4.1: Basic statistics of datasets.

	ATIS	SNIPS
Training Samples	4978	13084
Validation Samples	500	700
Test Samples	893	700
intent type	18	7
Slot types	127	72

Precision (P): Precision calculates the number of intent/slots predicted by the model as positive are actually positive, as given in the following equation.

$$Precision = \frac{\#true\_positives}{\#true\_positives + \#false\_positives} \tag{4.12}$$

Recall (R): Recall calculates the number of actual positive intent/slots the model capture through labeling as positive.

$$Recall = \frac{\#true\_positives}{\#true\_positives + \#false\_negatives} \tag{4.13}$$

F1 Score (F1): F1 Score computes the harmonic mean of precision and recall, defined as follows:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.14}$$

4.3.3 Training Setting

Our model utilizes pre-trained Bert-Base model[1] which has 12 Transformer encoder layers, 768 hidden states, and 12 self-attention heads. The BERT is pre-trained on Books Corpus [164] and English Wikipedia. The Adam optimizer [165] is employed to optimize the training loss using a fixed learning rate of $5e-5$. The sentence maximum length is set to 50. Furthermore, we trained the model using a mini-batch size of 32 training examples. Number of iterations per mini-batch is 128. To avoid gradient exploding, gradient clipping is used and the maximum norm is set to 5. The tensor decomposition rank r is set to 200.

4.3.4 Baseline Models

We compare the model performance against the following baseline models.

- Attention-based Encoder-Decoder (Att-ED): The Att-ED model [40] employs bidirectional RNN to embed user sentences into vector space. The model then employs another uni-directional RNN for predicting the slot label. Additionally model utilizes an attention mechanism to exploits information from different parts of user sentences for predicting slot labels.

[1] https://storage.googleapis.com/bert_models/2020_02_20/uncased_L-12_H-768_A-12.zip

As suggested by the authors we used 128 units in LSTM. To avoid overfitting, the dropout rate of 0.5 is used. We used the online available code[2].

- SlotGated-NLU: The model [41] employs bidirectional RNN-LSTM with an attention mechanism to predict the intent of the user. The weighted sum of BiLSTM hidden states is utilized for tagging slot labels. The model learns slot-intent relationships by employing an intent context vector as the additional gate in LSTM. The hidden unit in the LSTM cell is configured to 64. We utilized online available code[3] given by the authors.

- Capsule-NLU: The model [76] is based on capsule neural networks, it performs intent detection and slot filling by making use of a dynamic routing schema. It employs capsule neural networks to model the hierarchical relationship between words, slots, and intents of user utterances. We utilized the code provided by the authors [4].

- Stack-Propagation: The Stack-Propagation model [166] employs a shared self-attentive encoder to exploit shared knowledge between two tasks. The intent-detection decoder and slot filling decoder are used to perform intent detection and slot tagging. The model utilizes the intent detection output as input to the slot filling module to improve the slot filling performance. We have used the code provided by the authors[5].

- RE-NN: The RE-enhanced Neural Networks [102] employ bi-directional RNN-LSTM to encode user utterances. Furthermore, the model utilizes RE matching results in an RNN-LSTM as additional input features, to guide attention, or tune the output logits.

- JointBERT: The Joint BERT model [167] fine-tune the BERT model for joint intent detection and slot filling. The special `[CLS]` token of the last hidden layer is used for intent detection and all other tokens are passed to a fully connected network with softmax activation for slot tagging. The model is tuned using Adam optimizer for 10 epochs. The fixed learning rate of $2e-5$ is used for optimizing training loss.

- FA-RNN: The FA-RNN model [106] convert RE into FA and then the MLP layer is applied to matched RE rules for converting the representation to label logits. The model is trained on labeled data via Adam optimizer using a learning rate of 0.001 for 128 epochs. We have used the online available code [6].

4.4 Evaluation and Results

[2]https://github.com/HadoopIt/rnn-nlu

[3]https://github.com/MiuLab/SlotGated-SLU

[4]https://github.com/czhang99/Capsule-NLU

[5]https://github.com/LeePleased/StackPropagation-SLU

[6]https://github.com/jeffchy/RE2RNN

表 4.2: The proposed WFST-BERT model performance comparison against the baseline methods on limited training data setup. Taken from our paper [2]

Dataset	Model	Intent			Slot		
		P	**R**	**F1**	**P**	**R**	**F1**
ATIS	Att-ED [40]	88.94	88.17	88.52	92.48	91.94	92.34
	SlotGated-NLU [41]	89.03	88.73	88.89	92.85	92.39	92.63
	Capsule-NLU [76]	89.35	89.14	89.22	93.78	93.31	93.26
	Joint-BERT [167]	89.67	91.48	90.49	93.42	94.50	93.81
	Stack-Propagation [166]	92.64	91.25	90.95	93.30	94.09	93.69
	RE-NN [102]	89.12	91.26	89.95	93.13	94.45	93.62
	FA-RNN [106]	90.44	91.39	90.85	-	-	-
	WFST-BERT	90.94	**92.23**	**91.76**	93.35	**95.77**	**94.48**
SNIPS	Att-ED [40]	92.96	93.04	92.75	60.35	67.39	63.68
	SlotGated-NLU [41]	93.26	93.59	93.38	61.09	69.21	64.90
	Capsule-NLU [76]	93.86	93.55	93.66	66.75	74.46	70.39
	Joint-BERT [167]	96.78	96.73	96.75	87.64	91.17	89.37
	Stack-Propagation [166]	96.57	96.54	96.51	89.49	89.94	89.71
	RE-NN [102]	95.13	95.48	95.30	80.74	82.45	81.58
	FA-RNN [106]	95.26	95.71	95.48	-	-	-
	WFST-BERT	96.95	97.28	**97.12**	89.45	90.82	**90.15**

In this section, we assess the WFST-BERT model concerning two different evaluation scenarios: The first evaluation category assesses model performance in a limited data scenario. In this case, we conduct experiments on a limited data setting (Section 4.4.1), that utilizes 10% of the training data. In the second scenario, we perform experiments on complete training data (Section 4.4.2). Furthermore, the proposed model results are compared against state-of-the-art baseline models. The two different setups are used to find an estimation of the model performance on different amounts of training data.

4.4.1 Limited Data Training

Table 4.2 presents the results of a limited data training in which the model is trained on 10% labeled data of ATIS and SNIPS datasets. It can be viewed from the results that WFST-BERT outperforms all the baseline models for both intent detection and slot filling tasks. These results from two datasets reveal that the WFST-BERT model performs effectively in limited training data and reduces the burden of having a high amount of training data. Furthermore, it can be seen in Fig. 4.5, 4.6 that RE-NN and FA-RNN models achieved superior results than Att-ED, SlotGated-NLU, and Capsule-NLU models. The reason behind the superior performance is their ability to incorporate external knowledge in form of RE rules which is not present in Att-ED, SlotGated-NLU, and Capsule-NLU

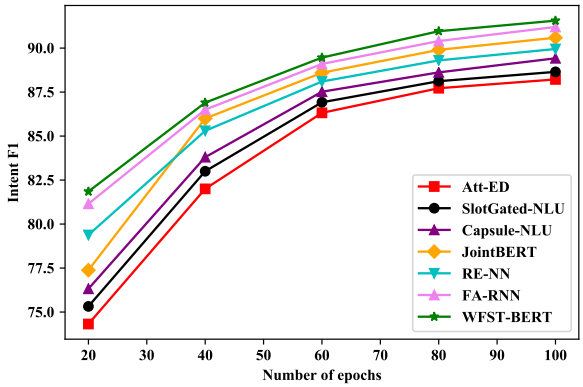


图 4.5: Limited data training results for intent classification task. Plot referred from our paper [2].

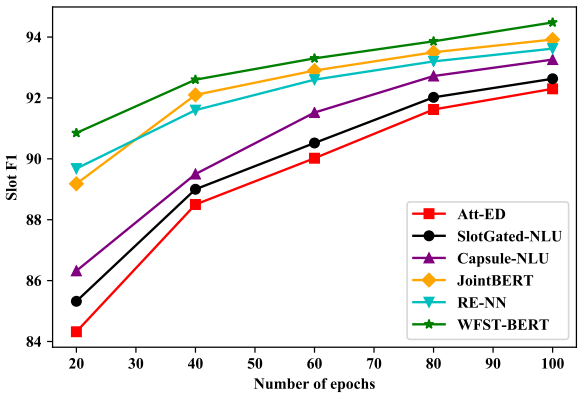


图 4.6: Limited data training results for slot filling task. Plot taken from our paper [2]. Reprinted with permission from Springer

models. Furthermore, the JointBERT model performs better than RE-NN on the intent detection task as it utilizes the contextualized word representation which lacks in the RE-NN model. The Stack-Propagation model also produced better results SlotGated-NLU, and Capsule-NLU models as it employed a self-attention mechanism to capture the contextual information of each word. However, the proposed WFST-BERT model improved intent detection task by 1.27% and 0.9% in terms of F1 score as compared to JointBERT and FA-RNN models, respectively on ATIS dataset. On SNIPS dataset WFST-BERT model improved intent detection task by almost 0.4% and 1.6% in terms of F1 score as compared to JointBERT and FA-RNN models, respectively. This is because the WFST-BERT model incorporates external knowledge using RE rules and exploits contextualized language representation by finetuning the BERT model. Moreover, WFST-BERT jointly trains both tasks to improve their performance. The WFST-BERT outperforms the state-of-the-art RE-NN and JointBERT model on both ATIS and SNIPS datasets on the slot filling task.

4.4.2 Full Data Training

Table 4.3 presents results on the full training data for intent detection and slot filling tasks on ATIS and SNIPS datasets. It can be observed that SlotGated-NLU produces superior results compared to the Att-ED model, its significance is attributed to its ability to learn slot-intent relations with slot gate mechanism and improve global optimization of the joint model. Capsule-NLU beats SlotGated-NLU by encapsulating the relationships between words, slots, and intents with dynamic routing-by-agreement schema. Results on two datasets shows that RE-NN and FA-RNN models obtain superior performance than former baseline models by incorporating the external knowledge via RE rules. The JointBERT model produces second superior performance and better generalization capability as it is pre-trained on large- scale English corpus. Nevertheless, the WFST-BERT produces state-of-the-art results by injecting domain knowledge using RE rules and exploiting contextualized word representation through a pre-trained BERT model. The WFST-BERT outperforms the best result by 0.55%, 1.07% F1-score for intent detection and slot filling task, respectively on ATIS dataset.

表 4.3: The proposed WFST-BERT model performance comparison against the baseline methods on full training data setup. Table adopted from our paper [2] with permission from Springer.

Dataset	Model	Intent			Slot		
		P	**R**	**F1±std**	**P**	**R**	**F1±std**
ATIS	Att-ED [40]	91.64	91.06	91.18	93.96	94.63	94.25
	SlotGated-NLU [41]	93.47	93.25	93.36	94.48	95.13	94.83
	Capsule-NLU [76]	96.12	94.21	95.02	95.06	95.42	95.28
	Stack-Propagation [166]	97.35	96.51	96.82	95.78	95.93	95.85
	Joint-BERT [167]	96.92	97.76	97.57	95.27	97.33	96.14
	RE-NN [102]	95.52	97.59	96.54	94.74	95.86	95.35
	FA-RNN [106]	95.47	97.44	96.32	-	-	-
	WFST-BERT	97.62	98.36	**98.12**	95.58	98.12	**97.21**
SNIPS	Att-ED [40]	96.78	96.67	96.72	89.24	86.45	87.81
	SlotGated-NLU [41]	97.28	97.39	97.28	90.83	86.95	88.85
	Capsule-NLU [76]	97.52	97.41	97.38	92.45	91.19	91.81
	Stack-Propagation [166]	98.01	98.05	98.00	94.78	93.65	94.21
	Joint-BERT [167]	98.11	98.03	98.06	95.84	96.59	96.21
	RE-NN [102]	97.69	97.57	97.62	94.03	91.86	92.93
	FA-RNN [106]	97.51	97.44	97.48	-	-	-
	WFST-BERT	98.51	98.24	**98.35**	96.82	96.68	**96.75**

Furthermore, on SNIPS dataset WFST-BERT obtains an improvement of 0.31% and 0.54% F1-score for intent detection and slot filling task, respectively.

4.4.3 Impact of hyper-parameters

The learning rate α and tensor decomposition rank r are important hyper-parameters of the model. We analyze the model performance with regard to these parameters. he random search [168] approach is employed for finding hyper-parameters randomly from the subset of hyper-parameters. We choose learning rate to optimize the model from range $[0.00001, 0.00002, 0.00005, 0.0001, 0.0002, 0.001, 0.002, 0.01]$. We can see from Fig. 4.7 that a learning rate of 0.00005 produces the best F1-score for both intent detection and slot filling tasks. Furthermore, results show that the model performance decreases on high learning rates of $0.01, 0.002, 0.001$ as the model fails to converge. On the contrary, the proposed model takes more time to converge on small a learning rate of 0.00001. Moreover, we choose tensor decomposition rank r from $[50, 100, 150, 200, 300]$. The tensor decomposition not only helps in reducing the number of parameters of WFST but also improves overall model performance. It is evident from Fig. 4.8 when rank size is increased, the model performance also increases. The model produces the best results when setting the value of r to 150.

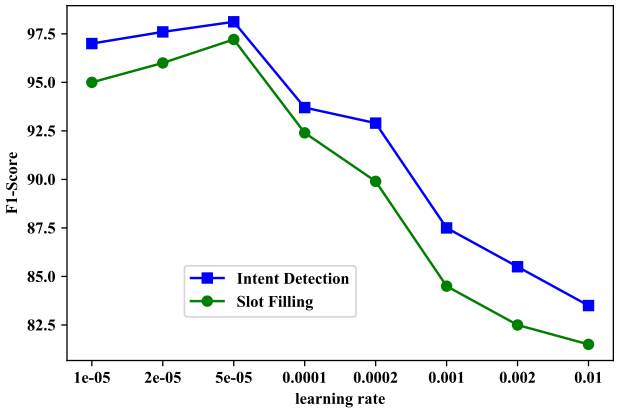


图 4.7: Impact of learning rate. Plot referred from our paper [2].

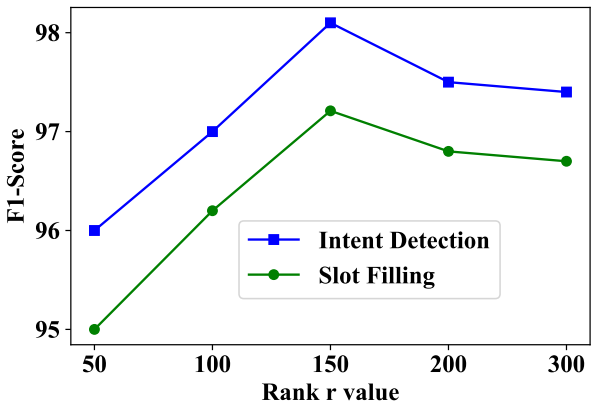


图 4.8: Impact of tensor decomposition rank r. Plot adopted from our paper [2].

4.4.4 Ablation Analysis

We conducted an ablation study to show the effectiveness of different aspects of the WFST-BERT model. We start the ablation by removing different components in the WFST-BERT model, we first remove the BERT part (-BERT) and then remove the WFST part (-WFST). Table 4.4 summaries ablation results on ATIS and SNIPS dataset. We can observe from the results that the BERT plays a major role in model performance. Removing the BERT encoder from the model decreases its performance by 1.8% in terms of F1 score for intent detection task and almost 3% F1 score for slot filling task on ATIS dataset. On SNIPS dataset removing BERT encoder from the model decreases intent

表 4.4: Evaluation results of WFST-BERT model ablation on ATIS & SNIPS datasets. Adopted from our paper [2]

Dataset	Model	Intent F1	Slot F1
ATIS	- BERT (WFST only)	96.34	94.22
	- WFST (BERT only)	97.65	96.16
	WFST+BERT	**98.12**	**97.21**
SNIPS	- BERT (WFST only)	96.72	93.17
	- WFST (BERT only)	98.00	95.65
	WFST+BERT	**98.35**	**96.75**

表 4.5: Independent training evaluation results of WFST-BERT model on intent detection and slot filling tasks. Table referred from our paper [2]. Reprinted with permission from Springer.

Dataset	Model	Intent F1	Slot F1
ATIS	WFST+BERT (Intent detection only)	97.53	-
	WFST+BERT (Slot filling only)	-	96.58
	WFST+BERT	**98.12**	**97.21**
SNIPS	WFST+BERT (Intent detection only)	98.18	-
	WFST+BERT (Slot filling only)	-	96.27
	WFST+BERT	**98.35**	**96.75**

detection performance by 1.63% and performance drop on slot filling task is 3.5% F1 score. Next, we demonstrate that the WFST part also contributes significantly to model performance. Removing the WFST part decreases the model performance by around 0.5% F1 score for intent detection and 1.1% F1 score for slot filling. On the SNIPS dataset, removing the WFST decreases performance of the model by 0.35% and 1.1% F1 score for intent detection and slot filling task, respectively. These results reveal that both WFST and BERT parts are important for the WFST-BERT model. Furthermore, we report ablation results on task level in Table 4.5. From the table we can see that the joint training of the model improves the performance of each intent detection and slot filling tasks on both ATIS and SNIPS datasets. On the ATIS dataset joint training of Intent detection and slot filling improves the performance of slot filling tasks by 0.63%. Similarly, on the SNIPS dataset slot filling performance improved by almost 0.5%.

4.5 Conclusion

In this chapter, we proposed a novel WFST-BERT model that combines a BERT architecture with WFST. The WFST-BERT employs REs rules to encode domain knowledge and converts them into the trainable weighted finite-state transducer. Furthermore, the model utilizes the language representation power of BERT to generate contextual representations and improve the generalization

capability. BERT representation is combined with WFST and trained simultaneously on supervised data using a gradient descent algorithm. Throughout the evaluation, we evaluate the performance of the WFST-BERT with baselines models on the ATIS dataset. Additionally, the performance of the WFST-BERT is assessed in limited data and full data setups. Our experiment results on intent detection and slot filling tasks show that the proposed model outperforms baselines neural approaches in both limited data and full data setups. Moreover, the WFST-BERT requires fewer training examples to learn user intents and associated slots.

第五章 Multi-Turn Intent Detection and Slot Filling with Neural Networks and Regular Expressions

5.1 Introduction

Content in this chapter first appeared in our publication [3].

Goal-oriented dialogue systems help users to solve a given task using natural language, such as finding a particular location, booking a ticket, or sending a message. Natural Language Understanding (NLU) module is a critical component of such systems, which converts the user utterance into a task-specific semantic representation. The main tasks of NLU are intent detection and slot filling [153]. intent detection predicts the user intent, and slot filling fills the set of arguments or slots corresponding to a semantic frame. For instance, *"Please book a trip to New York from Mannheim"*. In this example, the intent of the user is to book a trip, *"New York"* and *"Mannheim"* fills the associated destination and origin city slot, respectively. Other words not corresponding to slots are tagged with the null label "O". The results from intent detection and slot filling are then used by the dialogue manager to form a query for the back-end and respond to the user accordingly.

Some prior studies [37, 40, 41, 65, 83] have focused on single-turn NLU in which the system receives only one utterance at a time and predicts user intent and slot labels. However, goal-oriented dialogue systems require both user and system to take multiple turns of back-and-forth interactions to accomplish a specific user goal. In multi-turn NLU, the user, as well as the system can refer to the entities of previous dialogue turn, resulting in contextual ambiguity [169]. For example, "all three" in an utterance can represent a number of activities, days, or tickets, generating ambiguity in understanding the semantics behind.

To overcome this problem, incorporating prior dialogue history has shown effective results in resolving such ambiguities of the utterances [85-87]. For instance, Chen et al. [85] employed a memory network to incorporate prior dialogue turns. In contrast, research studies [86, 87] exploited the temporal order of the utterances to encode dialogue history in chronological order. The aforementioned methods generate promising results when there is a large amount of labeled data. Unfortunately, such labeled data is not always available. On the other hand, large-scale pre-trained language models, such as BERT (Bidirectional Encoder Representations from Transformers) [43] and XLNet [44] have achieved the state of the art performance on NLU tasks after simple fine-tuning. For instance, BERT is pre-trained with the transformer encoder on the large-scale unlabeled text and encodes the context of words from both forward and backward directions. As a result, the pre-trained BERT model can be then fine-tuned with comparatively small and task-specific training data on downstream tasks such as text classification [90]. Therefore, these large scale pre-trained language models have supported the improvement of language understanding tasks, even in the absence of sizeable data sets. Furthermore,

regular expressions that are based on the hand-crafted rules, do not require training data to generate output [25].

Regular expressions (REs) are widely used for information extraction, named entity recognition, sentence classification, and slot tagging tasks [24, 25]. REs consists of human-defined rules which are concise, tunable, and do not require much training data to generate. Due to the vast diversity in the user utterances, it is impossible to create a RE to identify particular slots and sentences with perfect accuracy. Therefore, instead of abandoning REs in favor of neural network approaches for sentence classification and slot tagging tasks, we can combine REs with neural networks. REs can complement the neural networks and can regulate the output of neural networks, especially in the absence of enough training data. More importantly, REs provide a mechanism for domain experts to encode domain knowledge and guide neural network models to capture desired patterns. REs can be applied at sentence-level and token-level. At the sentence-level, REs can be applied to search for key phrases in user utterance for particular user intent. At the token-level, REs can be employed to detect word sequence in user utterance and assign word-level RE label.

In light of the above intuition, we presents a Neural Network with Regular Expression (NN-RE) model, which exploits the expressive power of regular expressions along with the robustness of neural networks to predict user intents and slots labels. Furthermore, the correlation between intent detection and slot filling tasks makes them suitable for multi-task learning. The proposed model obtains contextual word representations of current utterances from the pre-trained BERT model. These representations along with dialogue context are utilized by dilated CNN to capture local features with a wider context for intent detection. At the same time, the model employs RNN to model the temporal relationship between word representations for slot filling task. More specifically, the model first employs a memory network to encode a dialogue context. The dialogue context vector is then shared by CNN and RNN for optimizing the objectives of slot filling and intent detection simultaneously via joint learning. Finally, regular expressions are utilized to complement CNN and RNN by encoding domain knowledge and handling cases with a limited amount of labeled data.

The main contributions of this study are summarized as follows:

- We propose a model that exploits the domain knowledge encoded in regular expressions to improve the performance of neural networks, especially in limited training data setting.
- We present a multi-task architecture that extracts the contextual word features from the pre-trained language model and employs CNN and RNN structures for modeling intent detection and slot filling tasks, simultaneously.
- We conducted extensive experiments on two real-world datasets, and evaluated the results of the proposed model against baseline competitors in terms of precision, recall, and F1 score metrics.

Our model differs from the above-mentioned works as we propose a multi-task model employing dilated CNN for capturing local features and RNN for modeling temporal relationships for intent

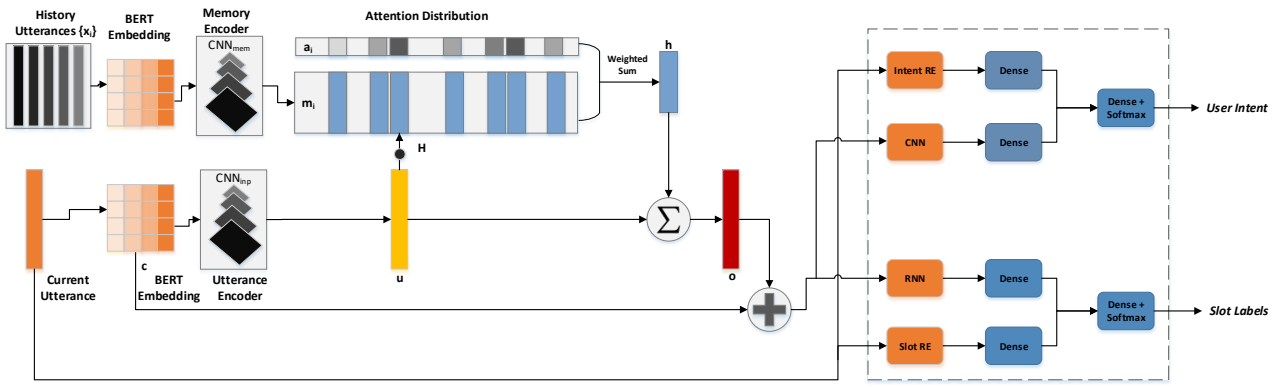


图 5.1: Architecture of the proposed model for multi-turn intent detection and slot filling. A dialogue context vector and current utterance is shared by RNN and CNN network. REs are integrated with RNN and CNN to predict user intent and slot labels. First published in our paper [3].

detection and slot filling tasks, respectively. Furthermore, the proposed model uses CNN for encoding a dialogue context instead of RNN, which is computationally expensive because the next output depends on the previous time step. The dialogue context representation is shared between the tasks. Moreover, the proposed model leverages REs to encode domain knowledge about a particular intent or slot value into the training of the model.

5.2 Model Architecture

The proposed model architecture is depicted in Fig. 5.1. The model is divided into two main modules, namely: memory network encoder (Section 5.2.1), and intent detection and slot-filling module (Section 5.2.2). The memory network encoder module generates a vector representation of the dialogue history, which is later fed into the input of intent detection and slot-filling tasks. The intent detection and slot-filling tasks module use dilated CNN and RNN to determine user intent and extract associated slots of given utterance simultaneously, and it employs regular expression to complement neural networks.

5.2.1 Memory Network Encoder

In this section, we discuss how to generate a dialogue context vector from the prior dialogue turns. The proposed model encodes previous user and system utterances into vector space, and stores these vectors in the memory. Similarly, a current utterance is also encoded into a vector space, which is then compared with memory vectors for encoding knowledge using cosine attention. Further details of the memory network encoder working process are given in the following subsections.

Contextual Word Representation from BERT

To extract contextual word representations of user utterances, our model employs the pre-trained BERT [43] model, which has 12 transformers [66] encoders, 768 hidden states, and 12 self-attention

heads. The BERT model is pre-trained on large-scale unlabeled text with training objectives of masked language modeling and the next sentence prediction. The BERT encoder computes the user utterance representation, i.e., hidden states for each token x_t as shown in equation 5.1:

$$H_t^n = BERT(x_{[CLS]}, x_1, .., x_t, x_{[SEP]}) \tag{5.1}$$

where $H_t^n = (h_{[CLS]}^n, h_1^n, .., h_t^n, h_{[SEP]}^n)$, n denotes number of BERT encoder layer, and h_t is the contextual representations of token t. For every word in the utterance, we obtain the word representations by summing the hidden states of last 4 layers $H_t = H_t^9 + H_t^{10} + H_t^{11} + H_t^{12}$ as suggested by Devlin et. al. [43]. The contextual representations of each token are passed to CNN and LSTM layers to predict the corresponding user intent and slot labels. It is noteworthy that WordPiece tokenizer split tokens into sub tokens. For instance, 'quickest' split into two tokens 'quick', and '##est'. To tackle this problem in the slot filling task, we use the first sub-token representation of a word as input to the RNN model.

Utterance Encoder

The utterance encoder uses a convolutional neural network (CNN) to encode current utterance c consists of n words: $c = w_1, w_2, .., w_n$, into continuous space with dimension d. The current utterance is passed through BERT embedding layer to extract fixed sized features. Furthermore, we apply a 1-dimensional convolution on top of these features. Next, max pooling operation on each filter is performed to obtain a fixed length output d. The representation of current utterance u can be defined as:

$$\begin{aligned} c &= BERT(c), \\ u &= CNN_{inp}(c), \end{aligned} \tag{5.2}$$

where CNN_{inp} represents 1-dimensional convolution followed by max pooling operation for input utterance.

The 1-dimensional convolution operation convolves a filter W over the window of m words in the input sequence to produce a feature map. For instance, we generate feature p_i by applying filter W to the window of words $[x_i, ..., x_{i:i+m-1}]$ as:

$$\begin{aligned} p_i &= f(W \cdot x_{i:i+m-1} + b), \\ x_{i:i+m-1} &= x_i \oplus x_{i+1} \oplus \cdots \oplus x_{i+m-1}, \end{aligned} \tag{5.3}$$

where b is a bias, $\oplus$ is the vector concatenation operator, and f represents non-linear activation function. We apply this filter to all locations of the current utterance c, producing feature maps: $p = [p_1, p_2, \cdots, p_{n-m+1}]$. The model then applies max-pooling operation to extract the maximum value from each feature map into a single fixed d-dimension vector: $u = max(p)$.

Memory Encoder

To store and incorporate dialogue history, the model takes previous dialogue turns i.e., $x_1, x_2, .., x_i$, passes them through the BERT layer to obtain the word-level features for each turn. These features are given as input to CNN for generating memory vectors $m_1, m_2, .., m_i$. Each memory vector possess same dimension d as the current utterance, as shown in equation 5.4.

$$\begin{aligned} x_i &= BERT(x_i), \\ m_i &= CNN_{mem}(x_i). \end{aligned} \tag{5.4}$$

The primary aim is to store previous turns knowledge into memory for future usage. To retrieve information from the memory, attention mechanism is applied to selectively pay attention to a specific history utterance. To do so, the model computes attention distribution a_i over the memory vectors $m_1, m_2, ..., m_i$ by taking the cosine similarity between each memory vector m_i and current utterance representation u followed by the softmax activation as shown in equation 5.5.

$$a_i = softmax(u^T m_i), \tag{5.5}$$

where $softmax(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$.

To accumulate knowledge from history, the history vector h is obtained by taking the weighted sum over the memory vectors m_i by the attention distribution a_i as presented in equation 5.6.

$$h = \sum_i a_i m_i \tag{5.6}$$

Finally, the history vector h and current utterance u are added to generate dialogue context vector o,

$$o = (h + u). \tag{5.7}$$

The dialogue context vector o is then passed to the intent detection and slot-filling module discussed in the following section.

5.2.2 Intent detection and Slot Filling Module

For both intent detection and slot-filling tasks, the model leverages the dialogue context. More specifically, the model concatenates dialogue context vector o with user utterance representation obtained from the BERT embedding layer c; this input is then shared by both tasks. The intent detection is usually framed as a sentence classification problem; therefore, we use a convolutional neural network. Convolutional neural networks are designed to produce local and position-invariant features, and they capture local key-phrases in user sentences; hence perform well on sentence classification tasks [61, 170]. On the other hand, the slot filling task is treated as a sequence labeling problem; therefore, we use a recurrent neural network (RNN). Recurrent neural networks can model temporal relationships, hence suitable for modeling sequential information [37, 40]. Furthermore, we employ regular expressions to encode domain knowledge and handle cases with a limited amount of training data.

Utterance	Slot RE:	Match	RE-Label assigned
Trip to New York from Mannheim.	/to dst_city from or_city/	to New York from Mannheim	O B-dst_city I-dst_city O B-or_city
We are 3 adults.	/([0-9]+) adults/	3 adults	B-n_adult I-n_adults

图 5.2: Slot RE examples with word-level labels assigned to the matched phrase. This figure is referred from our paper [3]. Reprinted with permission from Elsevier.

Slot Filling via Recurrent Neural Network

Recurrent neural networks (RNNs) are powerful architectures capable of capturing long-range dependencies via time-connection feedback [52]. Therefore, the proposed model uses a recurrent neural network (RNN) to model long-range dependencies. To overcome the vanishing gradient problem of RNNs, long short-term memory (LSTM) was designed [54]. LSTMs are the same as RNNs except that the hidden layer updates are replaced by memory cells. We place LSTM in forward and backward directions and concatenate two LSTMs outputs to access both past and future information for a given time. Input to LSTM is the dialogue context vector obtained from equation 5.7 and contextual word representation of current utterance as defined in equation 5.8.

$$\begin{aligned} \overrightarrow{h_t} &= \overrightarrow{LSTM}(o, c, \overrightarrow{h}_{t-1}) \\ \overleftarrow{h_t} &= \overleftarrow{LSTM}(o, c, \overleftarrow{h}_{t+1}) \\ s_r &= [\overrightarrow{h_t}, \overleftarrow{h_t}]. \end{aligned} \tag{5.8}$$

Where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the hidden states of forward and backward passes in BLSTM, while c represents current utterance and o demonstrates dialogue context vector. To obtain a per-class score for slot filling, the representation s_r is passed through a fully connected network.

$$logit_{slot} = W_r(s_r), \tag{5.9}$$

where W_r is the weight matrix for a fully-connected network and $logit_{slot}$ is the unnormalized predictions of a model.

Regular Expressions for Slot Filling

Regular Expressions (REs) are an algebraic notation that is used to describe search patterns we want to match. More specifically, REs search through the user sentences and return all texts which meet its search pattern. Existing research studies have exploited the significance of REs in different natural language processing tasks namely information extraction [26], sequence tagging [24] and named entity recognition [25] tasks. REs are often used to extract patterns from text based on predefined rules. However, REs can be combined with neural models instead of being just simple pattern matching. REs complement the robustness of neural networks by providing control of a rule-based system, especially in the absence of enough training data. Furthermore, REs facilitate domain experts to encode domain knowledge about particular user intent or particular structure of slot value. The

Utterance	Intent RE:	Match	RE-Label assigned
set reminder for doctor's appointment	/set (a) ? reminder /	set reminder	schedule
give me directions to the nearest shopping center	/(give)(?:\w+){0,3}?(directions)/	give me directions	navigate

图 5.3: Intent RE examples with the sentence-level label assigned to the matched utterance. This figure is taken from our paper [3]. Reprinted with permission from Elsevier.

proposed model incorporates knowledge expressed by REs into the training of CNN and RNN for the intent detection and slot filling.

For the slot filling task, we write regular expressions to match the specific structure of the slot being extracted. In Fig. 5.2, we illustrate some examples of slot REs. In first slot RE, "`or_city`" and "`dst_city`" represent word list of city names in the dataset. This RE matches "to New York from Mannheim" phrase. Slot RE module then assigns the RE label to each word of the matched phrase. The word-level labels are also converted into BIO format. For instance, `B-dst_city` and `I-dst_city` for New York destination city. In the second example, slot RE matches "3 adults" phrase of the utterance, "B-n_adults" and "I-n_adults" are the labels assigned to it.

To generate the slot REs score, a fully connected layer is applied to the REs search outcome. The output produced by slot REs is used to amend the output of RNN, which is fed as an input to a softmax layer for predicting IOB slot labels as defined in equation 5.10.

$$\hat{y}_{slot} = softmax(W_{slot}(logit_{slot} + W_{reg}R^k)). \tag{5.10}$$

Where $logit_{slot}$ is the logit obtained from equation 5.9 and W_{reg} represents a trainable weight matrix for REs while R^k is the REs search output containing 0-1 values for each slot, which indicates the absence or presence of at least one matched RE results in slot label k.

intent detection using Dilated CNNs

Traditional convolutional models incorporate local word context information into word representation, where the filter width specifies the local context size used. The standard CNNs can incorporate context size linear to the entire depth of the network, therefore makes it harder for tasks that require longer history. To overcome such problems, we use dilated convolutions that can capture large context by skipping the number of nearby words in subsequent convolutional passes. The Dilated convolution operator of filter W_p and dilation factor d applied to m-gram $x_{i+m.d}$ with output p_i is defined as:

$$\begin{aligned} p_i &= f(W_p \cdot x_{i+m.d} + b), \\ x_{i+m.d} &= x_i \oplus x_{i+1.d} \oplus \cdots \oplus x_{i+m.d}, \end{aligned} \tag{5.11}$$

where $\oplus$ is the vector concatenation operator, d is the dilation factor, b is the bias, and f is the activation function. We use hyperbolic tangent (tanh) as the non-linear activation function. We apply this filter with an exponentially increasing dilation $1, 2, 4$ and 8 to the concatenated vector of dialogue context

o and current utterance representation c for extracting feature vectors. A max-pooling layer is then applied to feature vectors to induce final indicative feature vector s_c for intent detection as defined:

$$\begin{aligned} p &= [p_1, p_2, \cdots, p_{n-m.d}], \\ s_c &= max(p), \end{aligned} \tag{5.12}$$

The main idea is to apply 1-dimensional convolution with dilated filters for capturing important m-gram with the wider context. Then, apply max pooling over time to extract the most important m-grams for determining user intent. Next, the final state s_c of dilated CNN is passed through a fully connected network to obtain per-class scores for each intent.

$$logit_{in} = W_c(s_c), \tag{5.13}$$

where W_c is the weight matrix for the fully connected network.

Regular expressions for intent detection

For the intent detection task, we write regular expressions to search for key phrases in user utterance and assign RE intent label on the basis of key phrases. In Fig. 5.3, we demonstrate some examples of intent REs. In the first example, intent RE finds clue words i.e., "set reminder" in user utterance and assign *schedule* RE label. In the second example, intent RE matches "give me directions" keywords and assign *navigate* RE label.

To generate an intent REs score, a fully connected layer is applied to the REs assigned labels. The output produced by intent REs then is used to amend the output of dilated CNN which is utilized for determining the user intent as defined follows.

$$\hat{y}_{in} = softmax(W_{in}(logit_{in} + W_{reg}R^k)). \tag{5.14}$$

Where W_{in} represents a trainable weight matrix, $logit_{in}$ is the logit obtained from equation 5.13 and W_{reg} represents a trainable weight matrix for REs while R^k is the REs intent labels containing 0-1 values for each intent, which indicates the absence or presence of at least one matched RE results in intent label k.

5.2.3 Joint Optimization

For joint optimization, we use categorical cross-entropy loss for both intent detection ($\mathcal{L}_{in}$) and slot filling ($\mathcal{L}_{slot}$) tasks. The loss function for intent detection is $\mathcal{L}_{in}$, and for slot filling is $\mathcal{L}_{slot}$.

$$\begin{aligned} \mathcal{L}_{in} &= -\sum_{i=1}^{l} y_{in}^i log(\hat{y}_{in}^i) \\ \mathcal{L}_{slot} &= -\sum_{t}\sum_{n} y_t^n log(\hat{y_t}^n) \\ \mathcal{L} &= \mathcal{L}_{in} + \mathcal{L}_{slot}. \end{aligned} \tag{5.15}$$

Where l is the number of intent types, t is the number of words in the utterance, n is the number of slot labels, and y_{in}, y_t^n denote the ground truth of user intent and slot label for the t-th word, respectively. The loss value minimized by the model is the sum of both individual losses. The overall optimization of the model is summarized in the Algorithm 5.1.

5.3 Experiments

5.3.1 The Dataset

The ATIS [161] dataset is commonly used in literature for single turn intent detection and slot filling tasks. In this work, our problem is multi-turn NLU, thus, the ATIS dataset does not fit the experimental setup. Hence, we considered two publicly available datasets namely Frames [171] and Key-Value Retrieval [13] for multi-turn NLU problems. The statistics of these datasets are given in table 5.2 and example dialogues are given in table 5.1.

Key-Value Retrieval dataset:

The Key-Value Retrieval (KVRET) dataset is composed of multi-turn dialogue between a driver and an In-car assistant[1]. A dialogue between the user and assistant starts with a user sentence that creates a dialogue domain using which the requirements of a user is achieved accordingly. This dataset contains three intents: calendar scheduling, point-of-interest navigation, and weather information. There are 15 slot types, each slot is stored with its value. Before using the dataset, we performed the following preprocessing steps: 1) annotating intent from session to sentence level by copying intent values. 2) obtaining word-level annotation by matching the slot values in the manual annotations with the corresponding sentence. We used IOB (Inside, Outside, Beginning) tag format, which is a common word-level annotation for slot filling tasks. Furthermore, we used 2425, 302, 304 dialogues for training, validation, and testing respectively.

[1] https://nlp.stanford.edu/blog/a-new-multi-turn-multi-domain-task-oriented-dialogue-dataset/

算法 5.1 Integrating RE into NN training. First Published in our paper [3].

Input: Training data (x_t, U, C, R_0, R_1)

// x_t: set of history utterances

// U: current utterances surface form

// C: current utterances word representations

// R_0: expert defined regular expressions for slots

// R_1: expert defined regular expressions for intent

Initialize neural network parameter θ

foreach *((u,c) ∈ (U,C))* **do**

- $o \leftarrow$ DialogueContext(x_t, c)
- $s_r \leftarrow$ RNN (o, c)
- $s_c \leftarrow$ DCNN (o, c)
- $logits_{slot} \leftarrow Dense(s_r)$
- $logits_{in} \leftarrow Dense(s_c)$
- **foreach** *($r_o \in R_0$)* **do**
  - match $\leftarrow$ re.search(r_o, u)
  - SlotLabel $\leftarrow$ match
- RESlot $\leftarrow$ Dense(SlotLabel)
- **foreach** *($r_1 \in R_1$)* **do**
  - match $\leftarrow$ re.search(r_1, u)
  - IntentLabel $\leftarrow$ match
- REIntent $\leftarrow$ Dense(IntentLabel)
- slots $\leftarrow$ softmax (Dense($logits_{slot}$, RESlot))
- intent $\leftarrow$ softmax(Dense($logits_{in}$, REIntent))
- Calculate loss using Equation. 5.15
- Update parameter θ

Output: neural network model parameter

Frames dataset:

Frames dataset[2] contains hotel and travel-booking dialogues collected in Wizard-of-Oz Scheme. It consists of 1369 multi-turn dialogues with an average of 15 turns per dialogue, for a total of 19986 turns. Each dialogue turn in the dataset is annotated with intent, slot types, and slot values. There are 20 intents and 28 slot types. Similar to the preprocessing steps adopted in the KVRET dataset,

[2]https://datasets.maluuba.com/Frames

表 5.1: Example dialogues from KVRET and Frames dataset. Example taken from our paper [3]

Dataset	Speaker	Utterance	Slots	Intent
KVRET	driver	What is the highest temperature for this week	weather_attribute, date	weather
	assistant	What location do you want to know?		
	driver	Menlo Park, please	location	weather
	assistant	On Friday in Menlo Park it will be a high of 100f.		
	driver	Get me directions to the nearest shopping mall	poi_type, distance	navigate
Frames	user	Hi im from Leon and looking to get away	or_city	inform
	wizard	Do you have a special destination in mind		request
	user	Maybe milan	dst_city	inform
	wizard	Any specific dates in mind		request
	user	august 18th until september 2nd	str_date, end_date	inform
	wizard	I have a trip available departing on Aug 19 and returning on the 23rd	str_date, end_date	offer

表 5.2: Basic Statistics of Datasets. First Published in our paper [3].

	KVRET	**Frames**
Training dialogues	2425	1109
Validation dialogues	302	125
Test dialogues	304	135
Total turns	12732	19986
Avg. turns per dialogue	5.25	15
intent type	3	20
Slot types	15	28

we obtained word-level tags by matching slot value with the concerned slot type in the sentences. Furthermore, the dataset contains a total of 11 participants dialogues, we selected two participants (ids 'U21E41CQP' and 'U231PNNA3') dialogues for testing and the other nine users dialogues are randomly divided into training (90%) and validation (10%) sets.

5.3.2 Creation of Regular Expressions

We used python `re` module for writing regular expressions. Our REs are written by a domain expert. It took less than 20 hours to write all intent REs and slot REs for both datasets. We randomly selected 10 training dialogues to create REs, and the writing process of REs is completed when REs cover most of the cases with reasonable accuracy. After that, we used those REs throughout the experiments. Generally, more complex REs containing multiple REs groups and alternation lead to better precision but slightly lower coverage. Furthermore, it is much faster to apply many small REs containing few groups and alternation instead of one large RE containing a lot of groups and alternation[172]. Therefore, we constructed REs with few groups, and alternation details are given below.

For the intent detection task, we created 15 distinct intent REs containing on average 3 groups and 5 alternations for the KVRET dataset. For the Frames dataset, 25 intent REs containing 2.5 groups on average with 3 alternations are created. To write 40 intent REs, it took around 7 hours. For the slot filling task, we created 30 slot REs with an average 2 groups and 6 alternations for the KVRET dataset and 35 slot REs with an average 2 groups and 4 alternations for the Frames dataset. It took us about 12.5 hours to write 65 slot REs.

5.3.3 Training Settings

The training setup of the proposed model consists of two kinds of settings namely: full training data setting and limited training data setting. In a full training data setting, we consider all training data to learn contextual features. Whereas, in limited training data setting only 20% of training data for each intent and slots are taken. Furthermore, for both settings, the proposed model is trained with Adam optimizer [165] using the default coefficients $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and fixed learning rate of 0.001. Besides, each update is computed through the batch size of 64 training samples. In addition, the number of epochs per batch is set to 150. Also, the model employed the pre-trained Bert-Base model[3] with 12 Transformer encoder layers, 768 hidden states, and 12 self-attention heads to obtain 768 dimension contextual word embeddings. Additionally, we use the memory size of 5 to store the knowledge from the previous five turns. Moreover, to avoid over-fitting in the model dropout is used as a regularizer. Furthermore, the rate of dropout is set to 0.5 for all dropout layers. Table 5.3 summarizes the details of hyper-parameters related to CNN and LSTM models used in this research study.

[3] https://storage.googleapis.com/bert_models/2020_02_20/uncased_L-12_H-768_A-12.zip

表 5.3: Hyper-parameters of experiments. Hyper-parameters setup, first published in our paper [3].

Layer	Hyper-parameter	Size
BERT Embedding	Dimension	768
CNN	number of filters	768
	filter size	3
LSTM	hidden layer	100
DCNN	number of filters	100
	filter size	3
	dilation rate	1,2,4,8
Dropout	dropout rate	0.5
	memory size	5
	batch size	64
	number of epochs	150
	learning rate	0.001

To evaluate the results of the proposed model, We have utilized the most commonly used evaluation metrics such as precision, recall, and F1 score as defined in section 4.3.2.

5.3.4 Baseline Methods

To assess the performance of the proposed model, we compared the results of the model against the following baseline methods.

1. Attention Encoder-Decoder: The model [40] encodes current utterance using bi-directional RNN-LSTM and generates the output using another uni-directional RNN-LSTM with attention. The number of units in the LSTM cell is set to 100 while the dropout rate of 0.2 is applied to avoid over-fitting. In this model, we do not provide any contextual information. We used the online available code[4].

2. Previous Turn Context: This architecture encodes previous turn and current utterance via bidirectional RNN-LSTM, which is given as an input to another RNN-LSTM for predicting the user intent and slots. The dimensions for the distributed word representations and the size of a hidden layer of LSTM are set to 100 each.

3. Memory Network (MemNet): The model architecture [85] encodes previous dialogue history and current utterances with attention and memory mechanisms. The dialogue context vector is fed as an additional input to the RNN-GRU tagger for predicting the user intent and slots. We trained this model with a memory size of 5 while the size of the hidden layer in LSTM is set to 100. We used the online available code[5] provided by the authors.

[4]https://github.com/HadoopIt/rnn-nlu

[5]https://github.com/yvchen/ContextualSLU

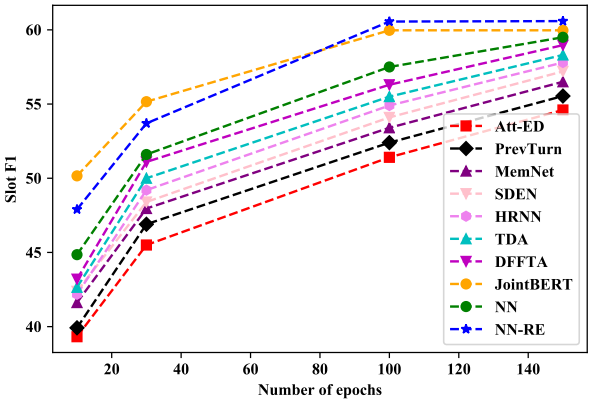


图 5.4: Learning from limited training data results on Frames dataset for slot filling task. Plot adopted from our paper [3].

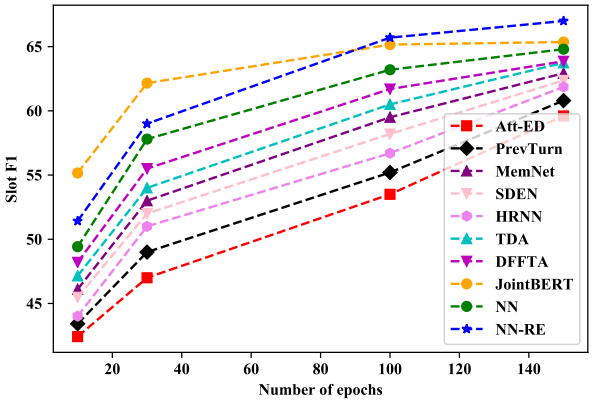


图 5.5: Learning from limited training data results on KVRET dataset for slot filling task. Plot taken from our paper [3].

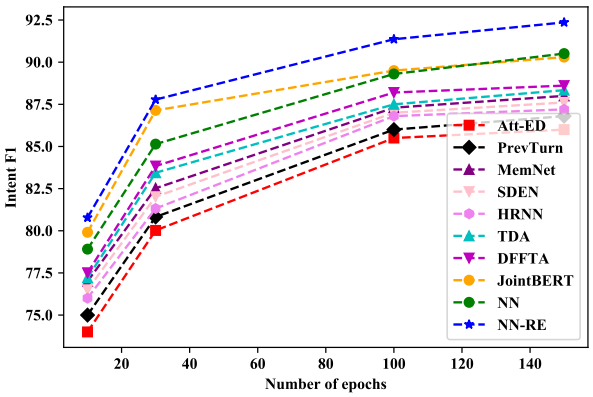


图 5.6: Learning from limited training data results on KVRET dataset for intent determination task. Plot referred from our paper [3].

4. Sequential Dialogue Encoder Network (SDEN): This model [86] encodes the previous history into memory vectors using bidirectional RNN-GRU. The number of units in BiGRU is set to 128 (64 each direction), while the 128-dimensional context vector is produced by a feed-forward layer from the concatenated vector of memory vector and current utterance. These context vectors are combined using another bidirectional RNN-GRU to represent dialogue context encoding used by tagger architecture for intent detection and sequence tagging. We trained the SDEN model with a memory size of 5. We have utilized MemNet code[6] after making the necessary modifications.

5. Hierarchical Recurrent Neural networks (HRRN): The model architecture [169], encodes system dialogue acts instead of the system utterance for producing dialogue context. It employs hierarchical RNN for encoding dialogue context obtained by summarizing the content of the dialogue act and the user utterance vector. This representation is then used by bidirectional RNN-LSTM for slot tagging. The size of the hidden layer in LSTM is set to 100.

[6]https://github.com/yvchen/ContextualSLU

6. Time-Decay Attention (TDA): The Time Decay Attention model [87], encodes dialogue history into the memory using bidirectional RNN-LSTM. The model uses a time-decay attention mechanism to pay more attention to recent utterances of dialogue by decaying attention weights over time. This representation is used by RNN-LSTM as additional input for sequence tagging. We trained the model with a history length of 5. The size of the hidden layer in LSTM is set to 128. We utilized the online available code[7] provided by the authors.

7. Decay-Function-Free Time-Aware Attention (DFFTA): The model [173] takes the current utterance as input and fed into bidirectional RNN-LSTM to obtain the current utterance summary. To summarize the dialogue context, the model uses the time difference between history utterance and current utterance. The model automatically learns the time-decay function of the history by introducing a trainable distance vector to pay more attention to important history utterances based on distance with current utterance. The dialogue context summary is fed as an additional input into RNN-LSTM for sequence tagging. We trained the model with a history length of 5. The size of the hidden layer in LSTM is set to 128. We used the code[8] provided by the authors.

8. BERT for Joint Intent Classification and Slot Filling (Joint BERT): The model [167] fine-tunes the pre-trained BERT model to jointly perform intent classification and slot filling tasks. The model utilizes the hidden states of the first `[CLS]` token for predicting user intent. For slot filling, the model feeds the final hidden states of all other tokens to the softmax layer to classify the slot labels. To make results comparable with other models, the length of history is set to be 5. The dialogue history utterances are sum together to form a dialogue context vector. The dialogue context vector is concatenated with the last hidden states of the current utterance. The model is fine-tuned with Adam optimizer using a learning rate of $5e-5$ and a batch size of 128 as suggested by authors.

5.4 Results and Discussion

Our experiments aim to investigate, whether REs can improve the learning ability of neural networks when we have limited training data. To answer this question, we have performed experiments on a limited training data setting (Section 5.4.1), which considers only 20% of the training data. We also investigated, whether REs are still helpful when we have enough training data. For this question, we have conducted experiments on a full training data setting (Section 5.4.2), which considers complete training data. Furthermore, this section compares the results of the proposed model against the results generated by state-of-the-art baseline models on Frames and key-value retrieval datasets.

[7] https://github.com/MiuLab/Time-Decay-SLU

[8] https://github.com/jgkimi/Decay-Function-Free-Time-Aware

表 5.4: Learning from limited training data results for the proposed Neural Network without Regular expression (NN) and Neural Network with Regular expression (NN-RE) models against baseline methods. Table adopted from our paper [3] with permission from Elsevier.

Dataset	Model	Context	Intent			Slot		
			P	**R**	**F1**	**P**	**Re**	**F1**
KVRET	Att-ED [40]	None	86.74	85.41	86.07	66.51	55.91	60.73
	PrevTurn	PrevTurn	87.06	86.56	86.80	60.84	62.33	61.58
	MemNet [85]	history	89.62	86.71	88.10	60.39	67.43	63.13
	SDEN [86]	history	88.44	87.39	87.91	67.13	68.67	62.27
	HRNN [169]	dialogue acts	87.26	86.73	86.99	61.08	62.56	61.86
	TDA [87]	history	89.63	86.82	88.15	61.41	67.75	63.45
	DFFTA [173]	history	89.67	87.16	88.34	60.87	68.05	63.74
	JointBERT [167]	history	90.14	89.47	90.13	68.82	63.58	65.81
	NN	history	90.72	89.94	90.51	68.19	62.74	65.16
	NN-RE	history	92.83	91.44	**92.16**	70.61	64.37	**67.23**
Frames	Att-ED [40]	None	74.36	69.86	72.04	58.31	49.23	53.34
	PrevTurn	PrevTurn	75.80	72.77	74.25	54.31	58.06	55.65
	MemNet [85]	history	76.29	73.72	74.54	60.12	54.23	57.00
	SDEN [86]	history	77.87	74.72	76.27	56.87	59.32	58.05
	HRNN [169]	dialogue acts	77.80	75.29	76.53	57.12	59.65	58.36
	TDA [87]	history	78.29	76.63	77.10	60.15	56.86	58.41
	DFFTA [173]	history	78.71	77.14	77.89	60.37	57.15	58.67
	JointBERT [167]	history	79.51	78.91	79.37	60.91	58.72	59.64
	NN	history	80.25	79.05	79.82	60.73	58.37	59.28
	NN-RE	history	82.26	79.74	**81.17**	62.85	57.64	**60.13**

5.4.1 Learning from Limited Training Data

Limited training data setting results are presented in Table 5.4. From the table, it can be observed that contextual models work better than the model with no context i.e Att-ED, or model with just previous turn context i.e. PrevTurn. Furthermore, results reveal that dialogue context is essential for multi-turn intent detection and slot filling tasks, giving an improvement of 4% and 5% F1 respectively on the KVRET dataset, and 7% and 6% improvement on the Frames dataset.

Furthermore, it can be observed in Fig. 5.4 and Table 5.4 that DFFTA and TDA models achieved superior results on both KVRET and Frames dataset. Because, these models can pay more attention to important history utterance by automatically decaying weights over time and manually decay weights over time, respectively. While on the Frames dataset HRNN produces superior results in comparison to SDEN and MemNet, as it utilizes dialogue context vector efficiently by initializing the hidden states of RNN tagger from it, while the dialogue act vector is fed as an additional input to RNN. In contrast, SDEN and MemNet use dialogue context to initialize the hidden state of RNN or as an additional input to RNN tagger. Furthermore, jointBERT achieved superior results than MemNet, SDEN, DFFTA, and TDA models as it leverages contextual word information using a pre-trained BERT model as opposed to static word embedding used by former models. However, NN-RE outperforms jointBERT in both precision and F1 score metrics, as the model lack domain information about slot types and particular patterns it may contain. On the contrary, the NN-RE model provides additional information about the slot type and its value, which boosts the performance of the slot filling task.

The results in Fig. 5.5 reveals that HRNN produces worse results than SDEN and MemNet on KVRET datasets, as dialogue acts information is limited in the KVRET dataset, and the dialogue context vector is too weak. Besides, Fig. 5.6 shows that the MemNet model achieved superior results on the intent detection task as it utilizes a single RNN tagger for jointly generating intent and slot information. Furthermore, the DFFTA model produced the superior on the intent detection task as it applies the sentence-level attention mechanism by leveraging the trainable distance vector to exploit the temporal information of the dialogue context. However, our proposed multi-task NN model based on RNN and dilated CNN significantly improved the performance of the intent detection task by almost 1% in terms of F1 score without degrading slot filling performance. The reason behind the improvement is that dilated CNN enables our model to capture the most important ngram with a wider context instead of a sequential RNN model. Additionally, if we look at the results of NN, it shows that the NN model learns task-specific features in an effective manner which, boosts the overall performance of the model.

Furthermore, the proposed NN-RE model provides a substantial boost on the F1 score of both intent detection and slot filling tasks over both KVRET and Frames datasets. The reason behind NN-RE model superior performance is that slot REs, and intent REs encode domain information about slot types and intent type in the utterance, respectively. This additional information helps the neural network to generalize better on smaller training data. We also observe that REs have less impact on results performance when there are large training data, it is not surprising as knowledge encoded in

REs can be captured by exploiting a large amount of training data.

表 5.5: Learning from full training data results for multi-turn intent detection and slot filling. Table is referred from our paper [3]. Reprinted with permission from Elsevier.

Dataset	Model	Context	Intent			Slot		
			P	**R**	**F1**	**P**	**R**	**F1**
KVRET	Att-ED [40]	None	92.86	91.76	92.31	70.25	76.63	73.30
	PrevTurn	PrevTurn	94.58	93.62	94.10	70.60	78.03	74.13
	MemNet [85]	history	98.01	98.01	98.01	71.53	78.27	74.75
	SDEN [86]	history	97.83	98.18	97.99	71.31	78.09	74.54
	HRNN [169]	dialogue acts	97.69	98.01	97.84	70.99	78.04	74.35
	TDA [87]	history	98.04	98.10	98.09	72.04	77.93	74.87
	DFFTA [173]	history	98.18	98.18	98.16	72.19	78.21	75.14
	JointBERT [167]	history	98.20	98.24	98.22	73.24	79.06	76.32
	NN	history	98.22	98.45	98.31	73.18	78.75	76.06
	NN-RE	history	98.50	98.46	**98.48**	74.32	79.18	**76.91**
Frames	Att-ED [40]	None	92.95	72.14	81.23	66.94	67.33	67.13
	PrevTurn	PrevTurn	88.35	87.14	87.74	67.51	71.49	69.44
	MemNet [85]	history	91.88	92.43	92.15	69.45	72.31	70.85
	SDEN [86]	history	92.98	91.61	92.29	70.70	71.98	71.33
	HRNN [169]	dialogue acts	93.25	91.79	92.51	71.82	71.23	71.42
	TDA [87]	history	92.45	92.36	92.56	70.64	72.34	71.58
	DFFTA [173]	history	92.52	92.48	92.67	70.72	72.47	71.71
	JointBERT [167]	history	93.96	92.10	93.31	71.16	73.42	72.51
	NN	history	93.60	92.27	93.06	70.38	73.59	71.94
	NN-RE	history	95.32	92.95	**94.17**	74.23	73.15	**73.53**

5.4.2 Learning from Full Training Data

The Full training data setting results are presented in Table 5.5. The results reveal that attention Encoder-Decoder performs poorly on both datasets against all baseline competitors, as it lacks in exploiting contextual information. Similarly, the Previous Turn context model generated the second poorest results, because the model merely considers previous turn context, ignoring the full dialogue information and therefore fail to determine the user intent and utterance slots. On the other hand, MemNet, SDEN, HRNN, TDA, DFFTA, NN models have exploited the full dialogue context; therefore, they perform better than the former models.

Results on two datasets reveal that the MemNet model produced superior results on the KVRET dataset as it utilizes dialogue context more effectively by concatenating it with LSTM input during

each time step. In contrast, the SDEN model achieved superior results on the Frames dataset because initial states of the LSTM tagger of SDEN are generated from dialogue context, which allowed the tagger to better handle noise in the dialogue context. Comparing the results on two datasets indicate that SDEN worked better than MemNet when dialogue length grew.

Moreover, the HRNN model produces superior results on the Frames dataset and inferior results on the KVRET dataset; as it uses system dialogue acts to generate dialogue context which is partially available in the KVRET dataset. Furthermore, the DFFTA and TDA model achieved superior results on both datasets as compared to other contextual models, as these models exploit the temporal information of dialogue history. The results indicate that DFFTA performs better than TDA as it automatically learns flexible and optimal time-decay function by using distance embeddings instead of utilizing manual time decay used by the TDA model. Besides, the JointBERT model produces superior results than DFFTA and TDA models because it exploits the pre-trained language model trained on large-scale English corpora. However, it is evident from the results that the proposed NN-RE model significantly outperformed all baseline competitors, as it leverages contextual word embedding from BERT model which is utilized in task-specific CNN and RNN networks to learn local and temporal features while leveraging the shared dialogue context information for intent detection and slot filling tasks. Furthermore, our model exploits domain knowledge encoded through regular expressions, hence further improving the F1 score on both tasks.

5.4.3 Effect of hyper-parameters

Table 5.3 shows the hyper-parameters setting used in this research. The most important hyper-parameters of the proposed model are learning rate, batch size, dropout, hidden layers, and dilation factor. We analyzed the performance of our model concerning these parameters. We use the random search [168] method to search hyper-parameters randomly from a specified subset of hyper-parameters. The learning rate for optimizing the model was chosen log-uniformly from 0.00005 to 1. In Fig. 5.7, we can observe that the model performs poorly on a learning rate of 1, which indicates that the model is unable to converge on a high learning rate. In contrast, using a too-small learning rate of 0.00005, the model takes more time to converge. On the KVRET dataset, the performance of the model increases steadily, especially on smaller learning rates of 0.001 and 0.005, and obtains F1 scores greater than 67 and 92 on the slot filling and intent detection, respectively. To choose an adequate learning rate, we apply a fine search scheme and find that our model generates the most significant results when configured on a learning rate of 0.001. Furthermore, we select batch sizes ranging from 2 to 1000. Fig. 5.8 reveals that in limited training data set, the model obtained the best performance with smaller batch sizes such as 32, and 64. Whereas, the learning efficiency of our model decreased for the larger batch sizes i.e., 128, 256, 512. When batch size is set to 64, our model achieves the best F1 score on both datasets. That is close to 60 and 81 for slot filling and intent detection on the Frames dataset, while the model achieved an F1 score 67 and 92 for slot filling and intent detection respectively on the KVRET dataset. Therefore, we chose batch size 64 for the limited

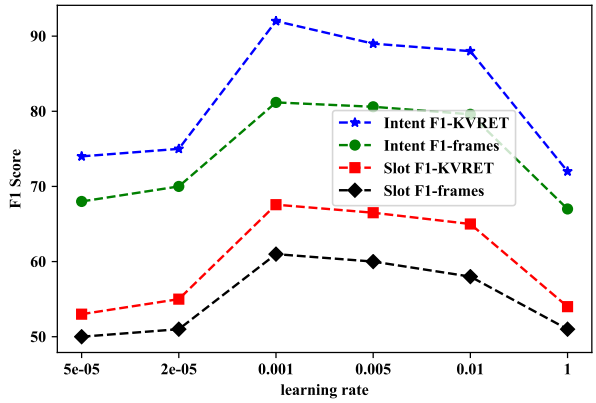


图 5.7: Effect of learning rate. Plot taken from our paper [3] with permission from Elsevier.

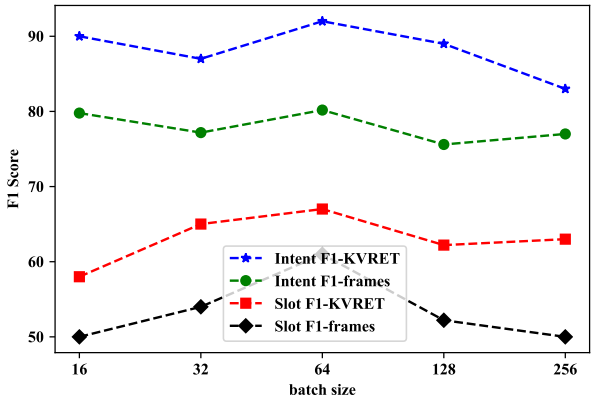


图 5.8: Effect of batch size. Plot adopted from our paper [3].

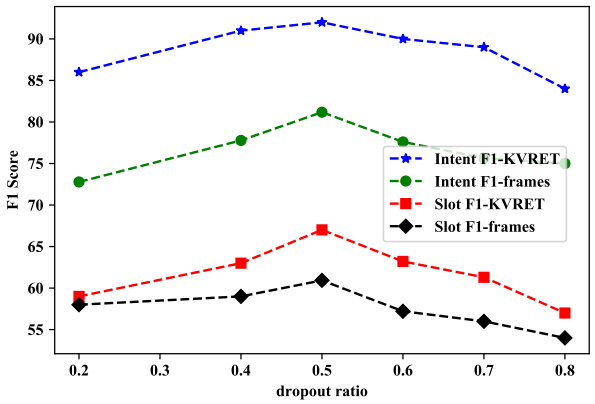


图 5.9: Effect of dropout ratio. Plot referred from our paper [3]. Reprinted with permission from Elsevier.

training data and full training data setting. Furthermore, we analyze the effect of the dropout layer depicted in Fig. 5.9. We observed a significant improvement for both intent detection and slot filling tasks when dropout is between 0.4 and 0.6. When dropout is greater than 0.6 model pays focus to the regular expression output and overlooks neural network output; therefore, it does not generalize well. For dilated convolution, the most important factor is the kernel size k and dilated factor d; we empirically found that $k = 3$ and $d = 4$ provide a sufficiently large receptive field for intent detection task. We also observed that the full training data was far less sensitive to hyper-parameters choice than the limited training data.

5.5 Summary

In this chapter, we have proposed a model that combines the expressive power of REs with the generalization ability of the neural network for multi-turn intent detection and slot filling tasks. The proposed multi-task model obtained contextual word representations of user utterances from the pre-trained BERT model. These representations were used by memory networks to encode dialogue context which is shared by the tasks. Furthermore, dilated CNN and RNN networks have been employed

on top of contextual word representations and dialogue context vector to learn task-specific features for intent detection and slot filling tasks. These neural networks have been combined with REs to encode domain knowledge about intent and slot values. Experiments on two real-world datasets demonstrate that the combination improves the performance of the model in both the full training data and limited training data setting. We showed that encoding domain knowledge into neural network training significantly improved the performance of neural networks. The encouraging results indicate that the proposed model can be used for improving other applications of text classification and sequence labeling.

第六章 Conclusion and Future Work

In this thesis, we have presented various methods for combining neural networks with knowledge. First, we pointed to recent development of spoken dialogue systems, especially the advancement of the NLU component for better understanding user requests. We discussed the challenges in existing spoken dialogue systems based on neural networks and rule based systems. To effectively address these challenges, we proposed solutions for combining neural knowledge with knowledge and evaluated them on real world datasets for a single turn and multi-turn dialogue system. In this last chapter, we provide conclusions with a brief summary of some achievements and present some possible future works.

6.1 Contribution 1: Natural language understanding framework for argumentative dialogue systems

The first contribution of the dissertation introduces the NLU model for argumentative dialogue systems in the domain of information seeking and opinion building described in chapter 3. We presented how we can detect user arguments within his or her utterance and distinguish between multiple intents including user preferences towards the respective arguments by multi task model. The proposed framework consists of two sub-models, namely intent classifier and argument similarity. Intent classifier model stack BiLSTM with attention mechanism on top of pre-trained BERT model and fine-tune the model for recognizing the user intent, whereas argument similarity model employs BERT+BiLSTM for identifying system arguments the user refers to in his or her natural language utterances. Our proposed model was applied and tested in an actual argumentative dialogue system on data collected in an extensive user study. We showed the proposed NLU model obtained state-of-the-art performance against baselines models on different datasets. We also demonstrated the robustness of the proposed approach against new topics, and the robustness of the approach against different language proficiency and cultural diversity.

Additionally, we demonstrated the superiority of our intent classifier model over baseline techniques in both few-shot and complete data setups. Furthermore, results show the better accuracy of the proposed model against baselines models for argument similarity tasks as well as the accuracy of 87.7% in complete pipeline testing. Moreover, no significant difference between utterances from UK users and Chinese users was detected. We showed that our model has to be trained only once for each system in order to learn the required system specific intents but does not require pretraining for new topics or user groups which ensures high flexibility of the respective system.

6.2 Contribution 2: The WFST-BERT Model for Joint Intent Detection and Slot Filling

Another main goal of this research was to reduce the need for massive labeled data. Therefore, In chapter 4, we proposed a novel WFST-BERT model that combines a BERT architecture with WFST for single-turn conversations. The WFST-BERT employs REs rules to encode domain knowledge and converts them into the trainable weighted finite-state transducer. Furthermore, the model utilizes the language representation power of BERT to generate contextual representations and improve the generalization capability. BERT representation is combined with WFST and trained simultaneously on supervised data using a gradient descent algorithm. Throughout the evaluation, we evaluate the performance of the WFST-BERT with baselines models on the ATIS and SNIPS dataset. Additionally, the performance of the WFST-BERT is assessed in limited data and full data setups. Our experiment results on intent detection and slot filling tasks show that the proposed model outperforms baselines neural approaches in both limited data and full data setups. Moreover, the WFST-BERT requires fewer training examples to learn user intents and associated slots.

6.3 Contribution 3: Multi-turn Intent Detection and Slot Filling with neural networks and regular expressions

The final contribution of this dissertation was to develop the multi-turn model with limited labeled data. So in Chapter 5, we presented a multi-task model for multi-turn intent determination and slot filling tasks that combines the expressive capacity of REs with the generalization capabilities of the neural network. The proposed multi-task model used the pre-trained BERT model to obtain contextual word representations of user utterances. Memory networks used these representations to encode the context of the shared conversation between tasks. Furthermore, dilated CNN and RNN networks have been employed on top of contextual word representations and dialogue context vector to learn task specific features for intent determination and slot filling tasks. These neural networks have been combined with REs to encode domain knowledge about intent and slot values. Experiments on two real world datasets demonstrate that the combination improves the performance of the model in both the full training data and limited training data setting. We showed that encoding domain knowledge into neural network training significantly improved the performance of neural networks. The encouraging results indicate that the proposed model can be used for improving other applications of text classification and sequence labelling.

6.4 Future Work

Our future work on the NLU model for argumentative dialogue systems will be focused on three separate aspects. First, we aim to integrate the herein presented NLU model into the argumentative dialogue system BEA and investigate in an extensive user study in real time its performance and

robustness. Second, we want to explore different confirmation strategies in the dialogue management in order to further improve the recognition rate of the complete system. Third, we will extend the capacities of the NLU to allow the user to introduce new arguments which will be learned by the system during the discussion.

Concerning the contribution of WFST-BERT, We intend to develop the WFST-BERT model to include other capabilities such as belief tracking and response generating. A regular expression could be used to track the state of the chat and create an appropriate answer. Additionally, we intend to generalize our framework to produce regular expressions from training data automatically. Another area of future research relevant is the interpretability of weighted transducers-BERT models by converting trained models back to regular expressions.

In a multi-turn approach, different memory-augmented networks or hierarchical neural structures can be utilized to store multi-turn knowledge. In multi-domain, we expect to further improve few-shot learning or zero-shot performance by writing regular expressions for different domains.

参考文献

[1] Abro W A, Aicher A, Rach N, et al. Natural language understanding for argumentative dialogue systems in the opinion building domain[J]. Knowledge-Based Systems, 2022: 108318.

[2] Abro W A, Qi G, Aamir M, et al. Joint intent detection and slot filling using weighted finite state transducer and BERT[J]. Applied Intelligence, 2022: 1-15.

[3] Abro W A, Qi G, Ali Z, et al. Multi-turn intent determination and slot filling with neural networks and regular expressions[J]. Knowledge-Based Systems, 2020, 208: 106428.

[4] Abro W A, Qi G, Gao H, et al. Multi-turn intent determination for goal-oriented dialogue systems[C]. in: 2019 International Joint Conference on Neural Networks (IJCNN). 2019: 1-8.

[5] Shang L, Lu Z, Li H. Neural Responding Machine for Short-Text Conversation[C]. 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, 2015: 1577-1586.

[6] Sordoni A, Galley M, Auli M, et al. A Neural Network Approach to Context-Sensitive Generation of Conversational Responses[C]. in: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Denver, Colorado: Association for Computational Linguistics, 2015: 196-205.

[7] Li J, Monroe W, Ritter A, et al. Deep Reinforcement Learning for Dialogue Generation[C]. in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, Texas: Association for Computational Linguistics, 2016: 1192-1202.

[8] Serban I V, Sordoni A, Bengio Y, et al. Building End-to-End Dialogue Systems Using Generative Hierarchical Neural Network Models[C]. in: AAAI'16: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. Phoenix, Arizona: AAAI Press, 2016: 3776-3783.

[9] Serban I V, Sordoni A, Lowe R, et al. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues[C]. in: AAAI'17: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. San Francisco, California, USA: AAAI Press, 2017: 3295-3301.

[10] Li J, Monroe W, Shi T, et al. Adversarial Learning for Neural Dialogue Generation[C]. in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen, Denmark: Association for Computational Linguistics, 2017: 2157-2169.

[11] Raux A, Langner B, Bohus D, et al. Let's Go Public! Taking a spoken dialog system to the real world[C]. in: Ninth European conference on speech communication and technology. 2005.

[12] Williams J D, Young S. Partially observable Markov decision processes for spoken dialog systems[J]. Computer Speech & Language, 2007, 21(2): 393-422.

[13] Eric M, Krishnan L, Charette F, et al. Key-Value Retrieval Networks for Task-Oriented Dialogue[C]. in: Jokinen K, Stede M, DeVault D, et al. Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017. Association for Computational Linguistics, 2017: 37-49.

[14] Madotto A, Wu C S, Fung P. Mem2Seq: Effectively Incorporating Knowledge Bases into End-to-End Task-Oriented Dialog Systems[C]. in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, 2018: 1468-1478.

[15] Qin L, Xu X, Che W, et al. Dynamic Fusion Network for Multi-Domain End-to-end Task-Oriented Dialog[C]. in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics, 2020: 6344-6354.

[16] Weizenbaum J. ELIZA—a computer program for the study of natural language communication between man and machine[J]. Communications of the ACM, 1966, 9(1): 36-45.

[17] Parkinson R C, Colby K M, Faught W S. Conversational language comprehension using integrated pattern-matching and parsing[J]. Artificial Intelligence, 1977, 9(2): 111-134.

[18] Jafarpour S, Burges C J, Ritter A. Filter, rank, and transfer the knowledge: Learning to chat[J]. Advances in Ranking, 2010, 10(2329-9290): 17.

[19] Yan R, Song Y, Wu H. Learning to respond with deep neural networks for retrieval-based human-computer conversation system[C]. in: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. 2016: 55-64.

[20] Yan Z, Duan N, Bao J, et al. Docchat: An information retrieval approach for chatbot engines using unstructured documents[C]. in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2016: 516-525.

[21] Chinnakotla M K, Agrawal P. Lessons from building a large-scale commercial IR-based chatbot for an emerging market[C]. in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. 2018: 1361-1362.

[22] Chiang D. Hierarchical Phrase-Based Translation[J]. Computational Linguistics, 2007, 33(2): 201-228.

[23] Ritter A, Cherry C, Dolan W B. Data-Driven Response Generation in Social Media[C]. in: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing. Edinburgh, Scotland, UK.: Association for Computational Linguistics, 2011: 583-593.

[24] Chang A X, Manning C D. TokensRegex: Defining cascaded regular expressions over tokens[J]. Tech. Rep. CSTR 2014-02, 2014.

[25] Zhang S, He L, Vucetic S, et al. Regular Expression Guided Entity Mention Mining from Noisy Web Data[C]. in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium: Association for Computational Linguistics, 2018: 1991-2000.

[26] Li Y, Krishnamurthy R, Raghavan S, et al. Regular Expression Learning for Information Extraction[C]. in: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing. Honolulu, Hawaii: Association for Computational Linguistics, 2008: 21-30.

[27] Tur G, Deng L, Hakkani-Tür D, et al. Towards deeper understanding: Deep convex networks for semantic utterance classification[C]. in: International Conference on Acoustics, Speech, and Signal Processing (ICASSP). 2012: 5045-5048.

[28] Ravuri S V, Stolcke A. Recurrent neural network and LSTM models for lexical utterance classification.[C]. in: INTERSPEECH. ISCA, 2015: 135-139.

[29] Ravuri S, Stolcke A. A comparative study of recurrent neural network models for lexical domain classification[C]. in: International Conference on Acoustics, Speech, and Signal Processing. 2016: 6075-6079.

[30] Haffner P, Tur G, Wright J H. Optimizing SVMs for complex call classification[C]. in: International Conference on Acoustics, Speech, and Signal Processing: vol. 1. 2003: I-I.

[31] Kim Y. Convolutional Neural Networks for Sentence Classification[C]. in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, 2014: 1746-1751.

[32] Zhang X, Zhao J, LeCun Y. Character-level convolutional networks for text classification[C]. in: Advances in neural information processing systems. 2015: 649-657.

[33] Chen Y N, Hakkani-Tür D, He X. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models[C]. in: International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2016: 6045-6049.

[34] Casanueva I, Temčinas T, Gerz D, et al. Efficient Intent Detection with Dual Sentence Encoders[C]. in: Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI. Online: Association for Computational Linguistics, 2020: 38-45.

[35] Henderson M, Casanueva I, Mrkšić N, et al. ConveRT: Efficient and Accurate Conversational Representations from Transformers[C]. in: Findings of the Association for Computational Linguistics: EMNLP 2020. Online: Association for Computational Linguistics, 2020: 2161-2174.

[36] Yao K, Zweig G, Hwang M Y, et al. Recurrent neural networks for language understanding.[C]. in: INTERSPEECH. 2013: 2524-2528.

[37] Mesnil G, Dauphin Y, Yao K, et al. Using recurrent neural networks for slot filling in spoken language understanding[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2015, 23(3): 530-539.

[38] Raymond C, Riccardi G. Generative and discriminative algorithms for spoken language understanding[C]. in: INTERSPEECH. 2007.

[39] Nguyen N, Guo Y. Comparisons of sequence labeling algorithms and extensions[C]. in: Proceedings of the 24th international conference on Machine learning. 2007: 681-688.

[40] Liu B, Lane I. Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling[C]. in: INTERSPEECH 2016. 2016: 685-689.

[41] Goo C W, Gao G, Hsu Y K, et al. Slot-Gated Modeling for Joint Slot Filling and Intent Prediction[C]. in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). New Orleans, Louisiana: Association for Computational Linguistics, 2018: 753-757.

[42] Wang Y, Shen Y, Jin H. A Bi-Model Based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling[C]. in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). New Orleans, Louisiana: Association for Computational Linguistics, 2018: 309-314.

[43] Devlin J, Chang M W, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[C]. 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, 2019: 4171-4186.

[44] Yang Z, Dai Z, Yang Y, et al. Xlnet: Generalized autoregressive pretraining for language understanding[C]. in: Advances in neural information processing systems. 2019: 5753-5763.

[45] Reimers N, Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks[C]. in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019: 3982-3992.

[46] Wang B, Kuo C J. SBERT-WK: A Sentence Embedding Method by Dissecting BERT-Based Word Models[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2020, 28: 2146-2157.

[47] He H, Gimpel K, Lin J. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks[C]. in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics, 2015: 1576-1586.

[48] Zheng T, Gao Y, Wang F, et al. Detection of medical text semantic similarity based on convolutional neural network[J]. BMC medical informatics and decision making, 2019, 19(1): 1-11.

[49] Kiros R, Zhu Y, Salakhutdinov R R, et al. Skip-Thought Vectors[C]. in: Cortes C, Lawrence N, Lee D, et al. Advances in Neural Information Processing Systems: vol. 28. Curran Associates, Inc., 2015: 3294-3302.

[50] Conneau A, Kiela D, Schwenk H, et al. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data[C]. in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen, Denmark: Association for Computational Linguistics, 2017: 670-680.

[51] Cer D, Yang Y, Kong S y, et al. Universal Sentence Encoder for English[C]. in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Brussels, Belgium: Association for Computational Linguistics, 2018: 169-174.

[52] Elman J L. Finding structure in time[J]. Cognitive science, 1990, 14(2): 179-211.

[53] Bengio Y, Simard P, Frasconi P. Learning Long-Term Dependencies with Gradient Descent is Difficult[J]. IEEE Transaction on Neural Network, 1994, 5(2): 157-166.

[54] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[55] Chung J, Gulcehre C, Cho K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[C]. in: NIPS 2014 Workshop on Deep Learning. 2014.

[56] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[57] Krizhevsky A, Sutskever I, Hinton G E. ImageNet Classification with Deep Convolutional Neural Networks[C]. in: Pereira F, Burges C J C, Bottou L, et al. Advances in Neural Information Processing Systems: vol. 25. Curran Associates, Inc., 2012.

[58] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[J]. ArXiv preprint arXiv:1512.03385, 2015.

[59] Sandler M, Howard A, Zhu M, et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks[C]. in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

[60] Tan M, Le Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks[C]. in: Proceedings of the 36th International Conference on Machine Learning. 2019: 6105-6114.

[61] Bai S, Kolter J Z, Koltun V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.[J]. CoRR, 2018, abs/1803.01271.

[62] Kalchbrenner N, Grefenstette E, Blunsom P. A Convolutional Neural Network for Modelling Sentences[C]. in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2014: 655-665.

[63] Zhang X, Zhao J, LeCun Y. Character-level Convolutional Networks for Text Classification[C]. in: Cortes C, Lawrence N, Lee D, et al. Advances in Neural Information Processing Systems: vol. 28. Curran Associates, Inc., 2015.

[64] Strubell E, Verga P, Belanger D, et al. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions[C]. in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen, Denmark: Association for Computational Linguistics, 2017: 2670-2680.

[65] Gupta A, Hewitt J, Kirchhoff K. Simple, Fast, Accurate Intent Classification and Slot Labeling for Goal-Oriented Dialogue Systems[C]. in: Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue. Stockholm, Sweden: Association for Computational Linguistics, 2019: 46-55.

[66] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]. in: Advances in neural information processing systems. 2017: 5998-6008.

[67] Wu Y, Schuster M, Chen Z, et al. Google's neural machine translation system: Bridging the gap between human and machine translation[J]. ArXiv preprint arXiv:1609.08144, 2016.

[68] Sarikaya R, Hinton G E, Ramabhadran B. Deep belief nets for natural language call-routing[C]. in: International Conference on Acoustics, Speech, and Signal Processing (ICASSP). 2011: 5680-5683.

[69] Zhang Z, Luo L. Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter[J]. Semantic Web, 2018, 10: 925-945.

[70] Ravuri S, Stolcke A. Recurrent neural network and LSTM models for lexical utterance classification[C]. in: Sixteenth Annual Conference of the International Speech Communication Association. 2015.

[71] Deng L, Yu D. Deep convex net: A scalable architecture for speech pattern classification[C]. in: INTERSPEECH. 2011: 2285-2288.

[72] Lin T E, Xu H. A post-processing method for detecting unknown intent of dialogue system via pre-trained deep neural network classifier[J]. Knowledge-Based Systems, 2019, 186: 104979.

[73] Howard N, Cambria E. Intention awareness: improving upon situation awareness in human-centric environments[J]. Human-centric Computing and Information Sciences, 2013, 3(1): 1-17.

[74] Liu R, Zhang X, Webb J, et al. Context-specific intention awareness through web query in robotic caregiving[C]. in: 2015 IEEE International Conference on Robotics and Automation (ICRA). 2015: 1962-1967.

[75] Xia C, Zhang C, Yan X, et al. Zero-shot User Intent Detection via Capsule Neural Networks[C]. in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium: Association for Computational Linguistics, 2018: 3090-3099.

[76] Zhang C, Li Y, Du N, et al. Joint Slot Filling and Intent Detection via Capsule Neural Networks[C]. in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, 2019: 5259-5267.

[77] Obuchowski A, Lew M. Transformer-Capsule Model for Intent Detection[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2020, 34(10): 13885-13886.

[78] Bunk T, Varshneya D, Vlasov V, et al. DIET: Lightweight Language Understanding for Dialogue Systems[J]. CoRR, 2020, abs/2004.09936.

[79] Minaee S, Kalchbrenner N, Cambria E, et al. Deep Learning–Based Text Classification: A Comprehensive Review[J]. ACM Computing Surveys, 2021, 54(3).

[80] Akhtar M S, Ekbal A, Cambria E. How Intense Are You? Predicting Intensities of Emotions and Sentiments using Stacked Ensemble[J]. IEEE Computational Intelligence Magazine, 2020, 15(1): 64-75.

[81] Basiri M E, Nemati S, Abdar M, et al. ABCDM: An Attention-based Bidirectional CNN-RNN Deep Model for sentiment analysis[J]. Future Generation Computer Systems, 2021, 115: 279-294.

[82] Cambria E, Li Y, Xing F Z, et al. SenticNet 6: Ensemble Application of Symbolic and Subsymbolic AI for Sentiment Analysis[M]. in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. New York, NY, USA: Association for Computing Machinery, 2020: 105-114.

[83] Hakkani-Tür D, Tür G, Celikyilmaz A, et al. Multi-Domain Joint Semantic Frame Parsing Using Bi-Directional RNN-LSTM[C]. in: INTERSPEECH. 2016: 715-719.

[84] Firdaus M, Kumar A, Ekbal A, et al. A Multi-Task Hierarchical Approach for Intent Detection and Slot Filling[J]. Knowledge-Based Systems, 2019, 183: 104846.

[85] Chen Y N, Hakkani-Tür D, Tür G, et al. End-to-End Memory Networks with Knowledge Carryover for Multi-Turn Spoken Language Understanding.[C]. in: INTERSPEECH. 2016: 3245-3249.

[86] Bapna A, Tür G, Hakkani-Tür D, et al. Sequential Dialogue Context Modeling for Spoken Language Understanding[C]. in: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue. Saarbrücken, Germany: Association for Computational Linguistics, 2017: 103-114.

[87] Su S Y, Yuan P C, Chen Y N. How Time Matters: Learning Time-Decay Attention for Contextual Spoken Language Understanding in Dialogues[C]. in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). New Orleans, Louisiana: Association for Computational Linguistics, 2018: 2133-2142.

[88] Kim Y B, Lee S, Sarikaya R. Speaker-sensitive dual memory networks for multi-turn slot tagging[J]. 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 2017: 541-546.

[89] Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations[C]. in: Proc. of NAACL. 2018.

[90] Howard J, Ruder S. Universal Language Model Fine-tuning for Text Classification[C]. in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, 2018: 328-339.

[91] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training (2018)[J]. URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf, 2018.

[92] Dong L, Yang N, Wang W, et al. Unified language model pre-training for natural language understanding and generation[C]. in: Advances in Neural Information Processing Systems. 2019: 13063-13075.

[93] Liu X, He P, Chen W, et al. Multi-Task Deep Neural Networks for Natural Language Understanding[C]. in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, 2019: 4487-4496.

[94] Sun Y, Wang S, Li Y, et al. Ernie: Enhanced representation through knowledge integration[J]. ArXiv preprint arXiv:1904.09223, 2019.

[95] Wang A, Singh A, Michael J, et al. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding[C]. in: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Brussels, Belgium: Association for Computational Linguistics, 2018: 353-355.

[96] Wang A, Pruksachatkun Y, Nangia N, et al. Superglue: A stickier benchmark for general-purpose language understanding systems[C]. in: Advances in Neural Information Processing Systems. 2019: 3266-3280.

[97] Minaee S, Kalchbrenner N, Cambria E, et al. Deep learning based text classification: A comprehensive review[J]. ArXiv preprint arXiv:2004.03705, 2020.

[98] Peters M E, Ruder S, Smith N A. To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks[C]. in: Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019. Association for Computational Linguistics, 2019: 7-14.

[99] Mikolov T, Sutskever I, Chen K, et al. Distributed Representations of Words and Phrases and their Compositionality[C]. in: Burges C J C, Bottou L, Welling M, et al. Advances in Neural Information Processing Systems: vol. 26. Curran Associates, Inc., 2013: 3111-3119.

[100] Hill F, Cho K, Korhonen A. Learning Distributed Representations of Sentences from Unlabelled Data[C]. in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. San Diego, California: Association for Computational Linguistics, 2016: 1367-1377.

[101] Sun C, Qiu X, Xu Y, et al. How to fine-tune bert for text classification?[C]. in: China National Conference on Chinese Computational Linguistics. 2019: 194-206.

[102] Luo B, Feng Y, Wang Z, et al. Marrying Up Regular Expressions with Neural Networks: A Case Study for Spoken Language Understanding[C]. in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, 2018: 2083-2093.

[103] Thompson K. Programming Techniques: Regular Expression Search Algorithm[J]. Commun. ACM, 1968, 11(6): 419-422.

[104] Viterbi A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm[J]. IEEE transactions on Information Theory, 1967, 13(2): 260-269.

[105] Schwartz R, Thomson S, Smith N A. Bridging CNNs, RNNs, and Weighted Finite-State Machines[C]. in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, 2018: 295-305.

[106] Jiang C, Zhao Y, Chu S, et al. Cold-Start and Interpretability: Turning Regular Expressions into Trainable Recurrent Neural Networks[C]. in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Online: Association for Computational Linguistics, 2020: 3193-3207.

[107] Hu Z, Ma X, Liu Z, et al. Harnessing Deep Neural Networks with Logic Rules[C]. in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics, 2016: 2410-2420.

[108] Alashkar T, Jiang S, Wang S, et al. Examples-Rules Guided Deep Neural Network for Makeup Recommendation[C]. in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. San Francisco, California, USA, 2017: 941-947.

[109] Awasthi A, Ghosh S, Goyal R, et al. Learning from Rules Generalizing Labeled Exemplars[C]. in: International Conference on Learning Representations. 2020.

[110] Xu J, Zhang Z, Friedman T, et al. A Semantic Loss Function for Deep Learning with Symbolic Knowledge[C]. in: Proceedings of the 35th International Conference on Machine Learning: vol. 80. PMLR, 2018: 5502-5511.

[111] Li T, Srikumar V. Augmenting Neural Networks with First-order Logic[C]. in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, 2019: 292-302.

[112] Waqas M, Khan Z, Anjum S, et al. Lung-Wise Tuberculosis Analysis and Automatic CT Report Generation with Hybrid Feature and Ensemble Learning.[C]. in: CLEF (Working Notes). 2020.

[113] Guo S, Wang Q, Wang L, et al. Knowledge Graph Embedding with Iterative Guidance from Soft Rules[C]. in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence. 2018: 4816-4823.

[114] Zhou H, Young T, Huang M, et al. Commonsense Knowledge Aware Conversation Generation with Graph Attention[C]. in: Proceedings of the 27th International Joint Conference on Artificial Intelligence. Stockholm, Sweden: AAAI Press, 2018: 4623-4629.

[115] Guan J, Wang Y, Huang M. Story ending generation with incremental encoding and commonsense knowledge[C]. in: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence: vol. 33. 2019: 6473-6480.

[116] Young T, Pandelea V, Poria S, et al. Dialogue systems with audio context[J]. Neurocomputing, 2020, 388: 102-109.

[117] Wen T H, Miao Y, Blunsom P, et al. Latent Intention Dialogue Models[C]. in: ICML'17: Proceedings of the 34th International Conference on Machine Learning - Volume 70. Sydney, NSW, Australia: JMLR.org, 2017: 3732-3741.

[118] Xu H, Peng H, Xie H, et al. End-to-End latent-variable task-oriented dialogue system with exact log-likelihood optimization[J]. World Wide Web, 2020, 23(3): 1989-2002.

[119] Locascio N, Narasimhan K, DeLeon E, et al. Neural Generation of Regular Expressions from Natural Language with Minimal Domain Knowledge[C]. in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, Texas: Association for Computational Linguistics, 2016: 1918-1923.

[120] Lawrence J, Reed C. Argument mining: A survey[J]. Computational Linguistics, 2020, 45(4): 765-818.

[121] Rach N, Weber K, Pragst L, et al. EVA: a multimodal argumentative dialogue system[C]. in: Proceedings of the 20th ACM International Conference on Multimodal Interaction. 2018: 551-552.

[122] Sakai K, Higashinaka R, Yoshikawa Y, et al. Hierarchical Argumentation Structure for Persuasive Argumentative Dialogue Generation[J]. IEICE TRANSACTIONS on Information and Systems, 2020, 103(2): 424-434.

[123] Hunter A, Chalaguine L, Czernuszenko T, et al. Towards computational persuasion via natural language argumentation dialogues[C]. in: Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz). 2019: 18-33.

[124] Shigehalli P R. Natural language understanding in argumentative dialogue systems[J]., 2020.

[125] Liu N F, Gardner M, Belinkov Y, et al. Linguistic Knowledge and Transferability of Contextual Representations[C]. 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, 2019: 1073-1094.

[126] Cer D, Diab M, Agirre E, et al. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation[C]. in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). Vancouver, Canada: Association for Computational Linguistics, 2017: 1-14.

[127] Aicher A, Rach N, Minker W, et al. Opinion Building based on the Argumentative Dialogue System BEA[C]. in: Proceedings of the 10th International Workshop on Spoken Dialog Systems Technology (IWSDS 2019). Siracusa (Sicily, Italy), 2019.

[128] Yuan T, Moore D, Grierson A. A Human-Computer Dialogue System for Educational Debate: A Computational Dialectics Approach[J]. Int. J. Artif. Intell. Ed., 2008, 18(1): 3-26.

[129] Hunter A. Towards a framework for computational persuasion with applications in behaviour change[J]. Argument & Computation, 2018, 9(1): 15-40.

[130] Ma Y, Nguyen K L, Xing F Z, et al. A survey on empathetic dialogue systems[J]. Information Fusion, 2020, 64: 50-70.

[131] Rakshit G, Bowden K K, Reed L, et al. Debbie, the debate bot of the future[G]. in: Advanced Social Interaction with Agents. Springer, 2019: 45-52.

[132] Le D T, Nguyen C T, Nguyen K A. Dave the debater: a retrieval-based and generative argumentative dialogue agent[C]. in: Proceedings of the 5th Workshop on Argument Mining. 2018: 121-130.

[133] Higashinaka R, Sakai K, Sugiyama H, et al. Argumentative dialogue system based on argumentation structures[C]. in: Proceedings of the 21st Workshop on the Semantics and Pragmatics of Dialogue. 2017: 154-155.

[134] Rosenfeld A, Kraus S. Strategical argumentative agent for human persuasion[C]. in: Proceedings of the Twenty-second European Conference on Artificial Intelligence. 2016: 320-328.

[135] Stab C, Gurevych I. Annotating Argument Components and Relations in Persuasive Essays[C]. in: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. Dublin, Ireland: Dublin City University, 2014: 1501-1510.

[136] Amgoud L, Ben-Naim J. Weighted Bipolar Argumentation Graphs: Axioms and Semantics[C]. in: IJCAI'18: Proceedings of the 27th International Joint Conference on Artificial Intelligence. Stockholm, Sweden: AAAI Press, 2018: 5194-5198.

[137] Lin Z, Feng M, dos Santos C N, et al. A Structured Self-Attentive Sentence Embedding[C]. in: 5th International Conference on Learning Representations, ICLR. 2017.

[138] Speer R, Chin J, Havasi C. Conceptnet 5.5: An open multilingual graph of general knowledge[C]. in: Thirty-First AAAI Conference on Artificial Intelligence. 2017: 4444-4451.

[139] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. ArXiv preprint arXiv:1301.3781, 2013.

[140] Pennington J, Socher R, Manning C. GloVe: Global Vectors for Word Representation[C]. in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, 2014: 1532-1543.

[141] Speer R, Havasi C. Representing General Relational Knowledge in ConceptNet 5[C]. in: Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12). Istanbul, Turkey: European Language Resources Association (ELRA), 2012: 3679-3686.

[142] Rach N, Langhammer S, Minker W, et al. Utilizing argument mining techniques for argumentative dialogue systems[C]. in: 9th International Workshop on Spoken Dialogue System Technology. 2019: 131-142.

[143] Aharoni E, Polnarov A, Lavee T, et al. A Benchmark Dataset for Automatic Detection of Claims and Evidence in the Context of Controversial Topics[C]. in: Proceedings of the First Workshop on Argumentation Mining. Baltimore, Maryland: Association for Computational Linguistics, 2014: 64-68.

[144] Coope S, Farghly T, Gerz D, et al. Span-ConveRT: Few-shot Span Extraction for Dialog with Pretrained Conversational Representations[C]. in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics, 2020: 107-121.

[145] Wolf T, Debut L, Sanh V, et al. Transformers: State-of-the-Art Natural Language Processing[C]. in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Online: Association for Computational Linguistics, 2020: 38-45.

[146] Reimers N, Beyer P, Gurevych I. Task-Oriented Intrinsic Evaluation of Semantic Textual Similarity[C]. in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. Osaka, Japan: The COLING 2016 Organizing Committee, 2016: 87-96.

[147] Wang R, Su H, Wang C, et al. To tune or not to tune? how about the best of both worlds?[J]. ArXiv preprint arXiv:1907.05338, 2019.

[148] Wu L Y, Fisch A, Chopra S, et al. Starspace: Embed all the things![C]. in: Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[149] Sanh V, Debut L, Chaumond J, et al. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter[J]. ArXiv preprint arXiv:1910.01108, 2019.

[150] Liu Y, Ott M, Goyal N, et al. Roberta: A robustly optimized bert pretraining approach[J]. ArXiv preprint arXiv:1907.11692, 2019.

[151] Lan Z, Chen M, Goodman S, et al. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations[J]. ArXiv, 2020, abs/1909.11942.

[152] Chi Z, Dong L, Zheng B, et al. Improving Pretrained Cross-Lingual Language Models via Self-Labeled Word Alignment[C]. in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Online: Association for Computational Linguistics, 2021: 3418-3430.

[153] Tur G, De Mori R. Spoken language understanding: Systems for extracting semantic information from speech[M]. John Wiley & Sons, 2011.

[154] E H, Niu P, Chen Z, et al. A Novel Bi-directional Interrelated Model for Joint Intent Detection and Slot Filling[C]. in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, 2019: 5467-5471.

[155] Wen T H, Vandyke D, Mrkšić N, et al. A Network-based End-to-End Trainable Task-oriented Dialogue System[C]. in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. Valencia, Spain: Association for Computational Linguistics, 2017: 438-449.

[156] Liu Y, Ott M, Goyal N, et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach[J]. ArXiv, 2019, abs/1907.11692.

[157] Arase Y, Tsujii J. Transfer Fine-Tuning: A BERT Case Study[C]. in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019: 5393-5404.

[158] Li X L, Rush A. Posterior Control of Blackbox Generation[C]. in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics, 2020: 2731-2743.

[159] Rabin M O, Scott D. Finite automata and their decision problems[J]. IBM journal of research and development, 1959, 3(2): 114-125.

[160] Hopcroft J. An n log n algorithm for minimizing states in a finite automaton[G]. in: Theory of machines and computations. Elsevier, 1971: 189-196.

[161] Hemphill C T, Godfrey J J, Doddington G R. The ATIS spoken language systems pilot corpus[C]. in: Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990. 1990: 24-27.

[162] Coucke A, Saade A, Ball A, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces[J]. ArXiv preprint arXiv:1805.10190, 2018.

[163] Mesnil G, Dauphin Y, Yao K, et al. Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2015, 23(3): 530-539.

[164] Zhu Y, Kiros R, Zemel R S, et al. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books.[C]. in: ICCV. IEEE Computer Society, 2015: 19-27.

[165] Kingma D P, Ba J. Adam: A Method for Stochastic Optimization[J]. CoRR, 2014, abs/1412.6980.

[166] Qin L, Che W, Li Y, et al. A Stack-Propagation Framework with Token-Level Intent Detection for Spoken Language Understanding[C]. in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019: 2078-2087.

[167] Chen Q, Zhuo Z, Wang W. BERT for Joint Intent Classification and Slot Filling[J]. CoRR, 2019, abs/1902.10909.

[168] Bergstra J, Bengio Y. Random search for hyper-parameter optimization[J]. Journal of machine learning research, 2012, 13(Feb): 281-305.

[169] Gupta R, Rastogi A, Hakkani D Z. An Efficient Approach to Encoding Context for Spoken Language Understanding[C]. in: INTERSPEECH. 2018: 3469-3473.

[170] Jacovi A, Sar Shalom O, Goldberg Y. Understanding Convolutional Neural Networks for Text Classification[C]. in: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Brussels, Belgium: Association for Computational Linguistics, 2018: 56-65.

[171] El Asri L, Schulz H, Sharma S, et al. Frames: a corpus for adding memory to goal-oriented dialogue systems[C]. in: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue. Saarbrücken, Germany: Association for Computational Linguistics, 2017: 207-219.

[172] Friedl J E. Mastering regular expressions[M]. ” O’Reilly Media, Inc.”, 2006.

[173] Kim J, Lee J H. Decay-Function-Free Time-Aware Attention to Context and Speaker Indicator for Spoken Language Understanding[C]. 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, 2019: 3718-3726.

Curriculum Vitae

Waheed Ahmed

Waheed Ahmed pursued his Doctoral degree from the School of Computer Science and Engineering, Southeast University, China and from the Institute of Communications Engineering of Ulm University, Germany. He has completed his Masters from FAST-National University of Computer and Emerging Sciences, Karachi, Pakistan. His research interest includes Language Understanding of Dialogue Systems, Natural Language Processing, Machine learning, and Computer Vision. During his PhD he has worked on various methods for combining neural networks with symbolic rules knowledge to reduce the need for labeled training data, and for improving generalization and scalability to new domains.

Education

Southeast University *Nanjing, China*
PhD Computer Science *2016 - 2022*
- Advisor: Prof. Guilin Qi

Ulm University *Ulm, Germany*
PhD Computer Science *2019 - 2022*
- Advisor: Prof. Wolfgang Minker

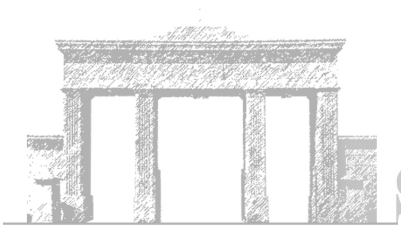
National University of Computer & Emerging Sciences *karachi, Pakistan*
MS Computer Science *2013 - 2016*
- Advisor: Dr. Syed Tahir Qasim

Mehran University of Engineering and Technology *Jamshoro, Pakistan*
BE Computer System *2004–2008*
- Thesis advisor: Engr. Syed Naveed Ahmed Jaffari

Professional Experience

2022-Present	**Assistant Professor.**, National University of Computer & Emerging Sciences, karachi, Pakistan
2011-2016	**SENIOR SOFTWARE ENGINEER.**, Xolva, Information technology services, Karachi **Research and Develop High quality, low cost software solution.**,
2010-2011	**JUNIOR SOFTWARE DEVELOPER.**, Apace Technologies, Karachi **Application Developer.**,

心求至善



SOUTHEAST UNIVERSITY