

AMERICAN UNIVERSITY OF BEIRUT

MACHINE LEARNING MODELS AND
RESOURCES FOR TASK-ORIENTED
CHATBOTS IN ARABIC

by

NOUR GHASSAN EL DROUBI

A thesis

submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Electrical and Computer Engineering
of the Maroun Semaan Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
August 2021

AMERICAN UNIVERSITY OF BEIRUT

MACHINE LEARNING MODELS AND
RESOURCES FOR TASK-ORIENTED
CHATBOTS IN ARABIC

by
NOUR GHASSAN EL DROUBI

Approved by:


Dr. Hazem Hajj, Associate Professor
Electrical and Computer Engineering

Advisor



Dr. Mazen Saghir, Associate Professor
Electrical and Computer Engineering

Member of Committee



Dr. Wassim El Hajj, Associate Professor
Computer Science

Member of Committee



Date of thesis defense: August 16, 2021

Acknowledgements

The work done in this thesis would not have been possible without the continuous support of my advisor, Dr. Hazem Hajj. His support, motivation, and guidance throughout my research helped me tremendously in completing this thesis. I would like to also thank my thesis committee members, Dr. Mazen Saghir and Dr. Wassim El Hajj for their support. Also, thanks to my family and friends who supported and motivated me throughout this thesis work. Finally, I would like to thank the American University of Beirut for all the opportunities offered and knowledge and experiences I acquired during my undergraduate and graduate studies.

An Abstract of the Thesis of

Nour Ghassan El Droubi for Master of Engineering
Major: Electrical and Computer Engineering

Title: Machine Learning Models and Resources for Task-Oriented Chatbots in Arabic

Recent developments enabled chatbots to be an essential part of people's daily lives from asking general questions about the weather to booking movie tickets. Chatbots can be classified into open-domain bots or task-oriented bots. Open domain chatbots can have engaging conversations in any domain. On the other hand, task-oriented chatbots, which are the focus of this thesis, aim at handling specific tasks such as booking movie tickets. While task-oriented chatbots have seen significant advances in English, task-oriented chatbots in Arabic remain limited in their capabilities mainly due to the scarcity of the available datasets and resources for training task-oriented dialogue systems in Arabic. To overcome these challenges, we have explored two state-of-the-art strategies for task-oriented bots: End-to-end models and pipeline models that consist of Natural Language Understanding (NLU) followed by the Dialogue Manager (DM) and Natural Language Generation (NLG). For end-to-end, we proposed the use of AraGPT2 and created a large multi-domain human-to-human conversational dataset in Arabic by translating a large-scale English dataset. Our end-to-end model achieved state-of-the-art results for Arabic and proved to be comparable in performance to what has been achieved by state-of-the-art English end-to-end models. For pipeline models, we addressed the NLU challenge by developing a multi-task model that can simultaneously perform intent classification and slot filling using AraBERT. To train the NLU model, we created a large dataset labeled for intents and slots by translating another large English dataset for training task-oriented bots. The developed NLU model was able to achieve comparable results with respect to the state-of-the-art results of pipeline models in English.

Contents

Acknowledgements	v
Abstract	vi
1 Introduction	1
2 Literature Review	4
2.1 English Task-Oriented Dialogue Systems	4
2.1.1 Pipeline Approach	5
2.1.2 End-to-End Approach	10
2.2 Arabic Task-Oriented Dialogue Systems	11
3 Methodology - End-to-End Approach	13
3.1 End-to-End Approach	13
3.1.1 AraGPT2	13
3.1.2 AraGPT2 for End-to-End Task-Oriented Chatbots	13
3.2 Dataset Created	16
3.2.1 MultiWOZ Dataset	16
3.2.2 Translation of the MultiWOZ Dataset	18
4 Evaluation and Results - End-to-End Approach	26
4.1 Training Details	26
4.2 Error Measures	26
4.3 Results	27
4.3.1 Response Generation	27
4.3.2 Policy Optimization	29
4.3.3 End-to-End System	30
4.3.4 Comparison of Results	33
5 Methodology - NLU Component of the Pipeline Approach	35
5.1 Background on the Pipeline Approach Architecture	35
5.1.1 Natural Language Understanding (NLU)	36
5.1.2 Dialog Manager (DM)	36
5.1.3 Natural Language Generation (NLG)	38

5.2	Natural Language Understanding (NLU) Training	39
5.2.1	Approaches	39
5.2.2	Dataset Created	42
5.2.3	Evaluation and Results	46
6	Future Work	56
7	Conclusion	58
	Bibliography	59

List of Figures

3.1	Example of GPT2 Training Sequence	14
3.2	AraGPT2 End-to-End Task Oriented Dialogue System Workflow .	15
3.3	Example of Response Delexicalization	16
3.4	Example Database Entry	17
3.5	Example Dialogue from MultiWOZ dataset	19
3.6	Example Translated MutliWOZ Database Entry to Arabic	20
3.7	Example of MutliWOZ Goals Translation	20
3.8	Example of Translated User Utterance	21
3.9	Example of Alignment Results of the Translated User Utterance .	22
3.10	Example of creating the Delexicalized User Utterance in Arabic .	22
3.11	Example of Replacing the placeholders in the Arabic Delexicalized User Utterance	23
3.12	Example of Dialogue Act Translation	23
3.13	Example of System Act Translation	24
3.14	Example of System Response Translation	24
3.15	Example Translated Dialogue from MultiWOZ dataset	25
4.1	Response Generation Results - Example 1	28
4.2	Response Generation Results - Example 2	29
4.3	Policy Optimization Results - Example 1	31
4.4	Policy Optimization Results - Example 2	32
4.5	Database Results of Example 2	32
4.6	End-to-End System Results - Example 1	33
4.7	End-to-End System Results - Example 2	34
4.8	Comparison of Our Results with State-of-the-Art	34
5.1	Architecture of Task-Oriented Chatbots	35
5.2	Example NLU Output	36
5.3	Examples of Dialog Acts	37
5.4	Example Output of the Dialog State Tracker	38
5.5	Example Output of the NLG	39
5.6	AraBERT for Intent Classification	40
5.7	AraBERT for Slot Filling	41
5.8	AraBERT for Joint Intent Classification and Slot Filling	42

5.9	ATIS Data Example	43
5.10	ATIS Sentence Translation Example	43
5.11	ATIS Word Alignment Example	43
5.12	ATIS Slot Alignment Example	44
5.13	ATIS Translation Problem - 1	44
5.14	ATIS Translation Problem - 2	45
5.15	ATIS Translation Problem - 3	45
5.16	AraBERT for Intent Classification - Train Results	47
5.17	AraBERT for Intent Classification - Test Results	47
5.18	AraBERT for Slot Filling - Train Results	48
5.19	AraBERT for Slot Filling - Test Results	48
5.20	Joint AraBERT – No CRF - Train Results	49
5.21	Joint AraBERT – No CRF - Test Results	49
5.22	Joint AraBERT – With CRF - Train Results	50
5.23	Joint AraBERT – With CRF - Test Results	50
5.24	NLU Results Comparison	51
5.25	Intent Classification Results	52
5.26	Slot Filling Results - 1	53
5.27	Slot Filling Results - 2	53
5.28	Slot Filling Results - 3	54
5.29	NLU Results Compared with SOTA	54

List of Tables

3.1	Domain Translation Mappings	20
4.1	Settings of Different Evaluation Scenarios	27
4.2	Response Generation Results	28
4.3	Policy Optimization Results	30
4.4	End-to-End System Results	31
5.1	NLU Training Parameters	46

Chapter 1

Introduction

Chatbots have emerged to be an essential part of daily communication and interaction with technology nowadays. Chatbots are computer programs that receive inputs from the user, process them, generate the appropriate responses and then send them back to the user [1]. Chatbots can be classified into two main categories which are open domain and closed domain. Open domain chatbots are capable of handling conversations in any domain such as social media conversations. However, closed domain or task-oriented chatbots are restricted to one or multiple domains and can only handle conversations in those specific domains [2]. Additionally, task-oriented chatbots have an end-goal to achieve depending on the user requirements and their aim is to meet this goal successfully. This is in comparison with open domain chatbots which their main goal is to respond reasonably and enhance user engagement [3].

The traditional approach used to implement task-oriented chatbots is the pipeline approach. Using this approach, task-oriented chatbots are divided into three main components. The first component is the Natural Language Understanding component which receives the user utterance and extracts the required information from it resulting in the belief state, then the Dialogue Manager component receives the belief state and communicates with a database to decide on the next action to be taken which is the system act. Finally, the Natural Language Generation component receives the system act and generates the corresponding system response in natural language. Each component in the pipeline approach is trained separately and then they are combined into one system [3]. Several work has been done on each of those components separately and on the system as a whole. On the other hand, a recent approach was introduced which is the end-to-end approach that treats the task-oriented dialogue system as one block. This approach was introduced due to the limitation of the interdependence between the components in the pipeline approach and the propagation of errors [4].

The area of creating chatbots has been attracting huge focus in the past period. Many chatbots were developed in the English language using both the

pipeline and the end-to-end approach. However, in the Arabic language significantly less work has been done in this area. This is mainly due to several challenges. The first challenge is that the Arabic language is a morphologically rich language and it has different forms which are the Modern Standard Arabic (MSA), the official written and read language, and the dialectal Arabic, the spoken form of the language [5]. In addition to that, resources available for the Arabic language are very scarce compared to the English language.

To address these challenges, an end-to-end approach is proposed for implementing the task-oriented dialogue system which is based on fine-tuning the AraGPT2 [6] pretrained language generation model. This approach was selected as the AraGPT2 model was pretrained on huge amount of data and this will help improve the performance of the model. In addition to that, a multi-domain human-to-human conversational dataset was created by translating the well-known MultiWOZ dataset [7] to Arabic. The MultiWOZ dataset is considered one of the largest datasets in English and it is now the first dataset available in Arabic to train task-oriented dialogue systems. The developed Arabic end-to-end task-oriented dialogue system was able to achieve comparable results with the state-of-the-art results achieved in the English language.

In addition to the end-to-end approach, the Natural Language Understanding (NLU) component which is part of the pipeline approach was also developed. The proposed method for the implementation is the use of AraBERT [8] for the joint implementation of both the intent classification and slot filling tasks of the NLU component. Additionally, a dataset for training the NLU component was created by translating the well-known ATIS [9] dataset. As there are no available datasets for training the NLU component in Arabic, the created dataset is the first available dataset.

As a summary, the contributions of this work are the following:

- The first task-oriented dialogue system in Arabic developed using the end-to-end approach with fine-tuned AraGPT2
- A multi-task BERT-based language model to simultaneously predict slots and intent for task-oriented chatbots in Arabic which is the objective of the NLU component that is part of the pipeline architecture of chatbot models
- A large Arabic dataset of 10438 dialogues each comprised of multiple utterances (4-10), labelled with user utterance, belief state, system act, and system response covering multiple domains which are hotel, restaurant, attraction, hospital, train, taxi, and police. This dataset is useful for training any task-oriented dialogue system in Arabic.
- A large Arabic dataset of 5871 user utterances comprised of users requesting flight reservations and information, labelled with intents and labels. This dataset is useful for training the NLU component of the task-oriented dialogue system.

The rest of this paper is structured as follows: Chapter 2 provides a literature review on previous work done on task-oriented dialogues in both the English and Arabic languages. Chapter 3 explains the methodology used for implementing the end-to-end approach and translating the dataset used. Chapter 4 provides the results achieved using the proposed end-to-end approach. Chapter 5 explains the pipeline approach and the work done on the NLU component and the translation of the dataset used. Chapter 6 provides the future work and finally Chapter 7 concludes the paper.

Chapter 2

Literature Review

Extensive work has been done in the development of task-oriented chatbots in English. Recent advances in the field of natural language processing have been implemented in the development of chatbots in English. However, the work in this area is still very scarce for the Arabic language. We will go through the advancements in the English and Arabic task-oriented chatbots in the following sections.

2.1 English Task-Oriented Dialogue Systems

The strong interest for the development and improvement of English chatbots has been increasing in the last few years. The first chatbot invented was ELIZA [10] in 1966 which used keyword recognition in the text to ask questions to the user. From that time, several chatbots were developed in the open and closed domains. In this section, we will focus on the most recent advances in the development of task-oriented chatbots in English.

Two main approaches have been used in the development of task-oriented dialogue systems. The first approach is the Pipeline approach which divides the system into several components which are the Natural Language Understanding (NLU), the Dialog Manager consisting of the dialogue state tracker and the dialog policy, and the Natural Language Generation (NLG). Each component in the pipeline approach is trained separately and then they are combined together in a pipeline manner [3]. On the other hand, the second approach is the end-to-end approach. In this approach, the task-oriented dialogue system is trained in an end-to-end manner where all the system is treated as one component. Previous work done on these two approaches will be mentioned in the following sections.

2.1.1 Pipeline Approach

Natural Language Understanding

The Natural Language Understanding (NLU) component has two main tasks which are the intent detection and the slot-value extraction [3]. The intent detection task is formulated as an intent classification problem and the slot-value extraction task is formulated as a slot filling problem and several approaches were used to tackle these problems.

For the problem of intent classification, the work in [11] proposed the use of Convolutional Neural Networks (CNNs). In this work, the proposed approach was based on the use of CNNs on characters only. This proves that the knowledge of the words or the semantic structure of the language is not required. Also the work in [12] proposed the use of CCNs. In this work, the query vector representations were used as features for the query classification. A vector representation was used to be able to group queries which are similar semantically. In [13], the use of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) were proposed for the task of intent classification. It was shown that RNNs performed better when the utterances were short, and LSTMs performed better when the utterances were long. In [14], an adversarial multi-task learning framework was introduced for text classification which trains the model on multiple tasks by dividing the task-specific and the shared parameters more accurately. This helps in creating a shared knowledge for all domains which can be transferred to any new domain.

For the problem of slot filling, the work in [15] used Recurrent Neural Networks (RNNs) to train the Language Understanding model. The RNN model was trained using the words in the sentence as input and the semantic labels as output. The work in [16] tackled the challenge of not taking previous context into consideration when predicting the slots. The approach suggested was the use of Convolutional Neural Networks (CNNs) which takes previous context into consideration and adds attention to the current word and its surrounding words. Also, the bi-directional sequential CNN architecture was introduced which takes into consideration the previous and future words for the slot prediction. In [17], the use of deep Long Short-Term Memory (LSTM) was introduced for the task of slot filling. The deep LSTM is made up of several layers of LSTMs and a regression model is added to the LSTM model to better model the relationships between the semantic labels. Also, the unnormalized scores were used before the Softmax layer to avoid the bias in slot labelling. Another approach introduced in [18] is the use of a generative neural network model. In this work, the challenges of having sentence-level annotations instead of word-level annotations and the presence of out-of-vocabulary words in the slot values were tackled. These were tackled by using both a sequence-to-sequence model and a pointer network in the generative neural network.

In addition to the separate approaches for the intent classification and slot filling, more recent approaches have proposed to model these two tasks jointly. In [19], the use of bi-directional RNN with LSTM cells was proposed where one RNN model jointly models the three tasks which are domain detection, intent classification, and slot filling. The input to the model is the user utterances and the output is a semantic frame which include the predicted domain, intent, and slots. In [20], the use of a slot-gated modelling approach was proposed which uses a slot gate to learn the relationships between the intent and the slots. This approach tackles the challenge of modelling the relationships between intents and slots as the slots are highly dependent on the intent. With the advancement of pretrained language models, most recent approaches have been using them to jointly model intent classification and slot filling. In [21], the use of BERT [22] was proposed to jointly model the intent classification and slot filling. This approach also tackled the challenge of scarcity of labelled data needed to train the NLU component. It was shown that the proposed joint BERT was able to generalize with less training data and was able to outperform previous work specifically the attention-based RNN and the slot-gated models.

Dialog Manager

Dialog State Tracking The dialog state tracking (DST) which is part of the dialog manager component in the task-oriented dialog systems has the goal of understanding the user’s utterances and to update the belief state accordingly [23]. The belief state, which is the probability distribution over all the dialog states, is then used by the Dialog Policy Learning component to decide on the next action to take [24]. Consequently, since the results of the DST component are then used to decide on the next action to take and the response generated, having accurate results from the DST component is important for a good performance for the whole system [25].

Initially, DST was implemented using traditional rule-based statistical approaches [26, 27, 28]. However, these approaches depend on manually created semantic dictionaries which are used to link slots and values with generic tags to have a generalized output [29]. This approach consequently faces the issue of lexical and morphological variations in the user’s utterances [25]. Then after the popularity of the use of deep learning, it was introduced in the implementation of DST. In [30], Deep Neural Networks were used in belief tracking. The use of Neural Networks enabled the inference of interactions found between features in the conversation. This work used the method of sliding windows to learn the tied weights of the neural network to output a sequence of probability distributions over several possible arbitrary values. The training was done on a specific domain, but the trained models are domain independent, and the learning can be transferred to new domains. The work in [23] tackled the problem of training the DST on one specific domain which was a limitation in previous work by propos-

ing a multi-domain belief tracking method. Recurrent Neural Networks (RNNs) were used for DST and were trained using data available from multiple domains in order to train a general belief tracking model by learning the most general and recurrent features. Then the model was tuned for every domain by learning the specific domain features. This model was able to outperform the DST which was trained on one specific domain only.

In [31], the Neural Belief Tracking (NBT) method was introduced and used in the development of the DST. This work tackled the limitation of scaling in DST since usually large amounts of annotated data are required for training the language understanding models and also manually created lexicons are required to handle linguistic variations in the conversation. The proposed NBT model uses pretrained word vectors to capture linguistic variations. The model then iterates over all the possible slot-value pairs to choose the one which was intended by the user. Following the limitation of having to manually retune the NBT model in [31] for every new domain, the work in [32] addressed this issue. In this work, several mechanisms for automatic updating were used by learning from the semantic decoding and context modelling of the NBT model. Another limitation in previous work addressed in [29] is the failure to determine the rare slot-value pairs which leads to the inability to determine the turn-level goal. This work proposed the Global-Locally Self-Attentive Dialogue State Tracker (GLAD) which uses global modules to learn shared parameters between the estimators of the different slots of the dialog state and the local modules for the slot-specific features. This proposed method was able to predict rare slot-value pairs by just using few training examples.

The work in [33] also tackled several limitations present in previous work. The limitations are the inability to dynamically change the domain ontology such as the slot and values for the DST, the models for every slot being different, and the difficulty in manually creating the semantic dictionaries for large scale domains. The model proposed is a universal state tracker StateNet which compares the distance between the representation of the dialog history and the vectors in the possible set to make a decision and this possible set can be dynamically updated. Also, the parameters of the model are shared across the slots which enables the transfer of knowledge and the reduction in the number of parameters. Also the work in [25] addressed the issue of scalability in domain wise training and in the addition of new slot-values that are not part of the ontology. This work proposed the slot-utterance matching belief tracking (SUBMT) approach which is a scalable and universal belief tracker that is independent from the domain and the slots. The model uses a non-parametric way to predict the slot-values so that the structure is not dependent on the domains and the slots. The model architecture also uses a shared knowledge across the domains and slots. SUBMT uses a slot-word attention mechanism using BERT to learn the relationships between dialog utterances and slots.

Additionally, the work in [34] tackles the challenges in tracking conversations

with long interaction and excessive information. The Slot Attention and Slot Information Sharing (SAS) model is proposed which applies slot attention to learn the features which are slot-specific from the original conversation and then uses Slot Information Sharing to integrate the slot values to be able to determine slots which are related. In [35] also the limitation of finding correlations among slots is considered where the Slot self-attentive dialogue state tracking (STAR) model is proposed. This model tackles the limitation of only considering the slot names to find the slot correlations. Consequently, STAR uses both the slot names and their values to find the slot correlations by using a slot-token attention module to find slot-specific features and then uses a stacked slot self-attention module to learn the correlations between the slots. This work was able to achieve state-of-the-art results on both the MultiWOZ 2.0 and MultiWOZ 2.1 compared to other DST models.

Policy Learning Based on the dialog state resulting from the DST, the goal of the policy learning is to decide on the next system action [4]. To optimize the policy learning, approaches used are either supervised learning approaches or reinforcement learning [4]. Several supervised learning approaches have been used which consider the agent utterances as labels and the model is trained to predict the next utterance [36]. However, the problem faced by this approach is that the model is trained to optimize the prediction of the next utterance and not the whole dialog and this can eventually lead to accumulating errors especially in multi-turn dialogues [37].

Other than the supervised learning approach, the policy learning problem is usually formulated as a Markov Decision Process (MDP) since the dialog acts are outputted sequentially and it is usually solved using Reinforcement Learning [3]. However, training the environment requires a user for interaction and two approaches are considered for this interaction which are training with real users or with user simulators. Several approaches considered real users for the training of the Reinforcement Learning models using Deep Q-Network (DQN) and Policy Gradient methods [3, 36, 38]. In [39], Deep Reinforcement Learning was used instead of the supervised learning and the traditional reinforcement learning approaches used in previous work. In this work Deep Reinforcement Learning was used with a high-dimensional state space in the board game of Settlers of Catan and results showed the improvement of the performance compared with other approaches such as the rule-based and supervised-based approaches. In [38], the work tackled the challenges faced in previous work implementing the DQN agents using the e-greedy heuristic which failed when the rewards are sparse and the action spaces are being large. As a solution for that, the Bayesian exploration strategy is proposed to help the agent in the selection of the action when the agent is uncertain by allowing it to explore the state-action regions.

The other approach which has been used as an alternative to real users is

the use of user simulators which act like the real users to train the environment [3]. In [40], the challenges of requiring a task-specific corpus and the extensive domain knowledge required to annotate human-to-human or human-to-machine conversations are tackled. The approach proposed is the use of a user simulator which is trained on example conversations. The use of user simulators to train the Reinforcement Learning agents act a starting point for the environment and then agents continue being trained by interacting with real users. However, using user simulators instead of real users is still not the optimal way to imitate the real human behaviours and this leads to having good performance when training but poor performance when used in the real human conversations [3]. In [41], an approach of integrating the Deep Dyna-Q (DDQ) with a switcher between using real or simulated users for the Q-Learning is proposed. This approach was proposed after the effectiveness of using DDQ in the training of Reinforcement Learning models but required the addition of planning to set the ratio of real users and simulators. The switcher used is implemented using an LSTM model and is trained with the dialog policy. Additionally, an active sampling strategy is used to generate the simulated experiences which are in the state-action space and are not explored by the agent. In [42], an approach was proposed which provides the agent with extra positive feedback. Also, user modelling was used to train the agent from the simulated interaction experience. Additionally, a meta-learning approach was proposed which allows the agent to learn from both the simulated users and the hindsight experience. The proposed approach in this work was able to achieve state-of-the-art results compared to other dialog policy learning approaches.

Natural Language Generation

The main task of the Natural Language Generation (NLG) component is to generate the system response based on the system act generated by the Dialog Policy component. The goal of the NLG component is to generate a natural language response which satisfies the requirements of the dialog act and hence this task is usually formulated as a conditioned language generation task [3]. Several approaches were proposed to train the NLG component. In [43], a forward RNN generator, a CNN reranker, and backward RNN reranker were trained to generate responses which are conditioned according to the dialog act. This approach is based on the training without semantic alignments or predefined grammar trees. In [44], the use of a Long Short-Term Memory (LSTM) recurrent network was proposed. The proposed LSTM architecture is semantically controlled and is trained using unaligned data. In [45], the challenge of training the NLG component which can be used in multiple domains was tackled. The proposed approach was the use of an RNN language generation model and first to train it using out-of-domain data and then fine tune the model using in-domain data.

In [46], a Context-Aware Long Short-term Memory (CA-LSTM) architecture

was proposed. The inputs to the model are the question, semantic slot values, and dialogue act and the output is the system response. An attention mechanism is also used to attend to the key information in the question. Also, the dialogue act type embedding is encoded to allow the generation of different responses based on the dialogue act. In [47], the sequence-to-sequence generation approach was proposed. This approach was used with the beam search and the n-best list reranker in order to stop any irrelevant information in the output. In [48], the challenge of limited data available to train the NLG component was tackled. To tackle this challenge, the model SC-GPT was proposed. This approach is based on first pre-training GPT using large datasets of plain text and then continuously pre-training on dialogue-act labelled utterances. Then finally the model would be fine-tuned on a specific domain using limited domain-specific data. This approach was shown to outperform previous methods based on automatic metrics and human evaluation.

2.1.2 End-to-End Approach

Following the pipeline approach explained earlier for the implementation of task-oriented dialogue systems, several limitations were observed from this approach. The main limitation observed is the interdependence between the components in the process [4]. Since each component is trained separately and then all components are combined, when one of the components is updated or retrained on new data, all the other components need to be updated as well. As a result of this limitation, the end-to-end approach was introduced to implement task-oriented dialogue systems. The end-to-end approach is based on developing the task-oriented dialogue system as one component that interacts with the knowledge base.

Several methods were suggested in the end-to-end approach. In [49], a single seq2seq model is proposed which is trained on both the task completion and the response generation problems. This model is based on a two stage CopyNet which has a smaller number of parameters compared with previous work. Several recent approaches have suggested the use of the pretrained language model GPT-2 [50] to train the end-to-end model. In [51], a single neural network model is trained on the tasks of state tracking, dialogue policy, and response generation. The approach proposed is to pretrain the GPT-2 model to generate system responses using large multi-domain data and then the model will be fine-tuned on a specific domain. This approach outperformed previous work especially in the case of having domains with a small amount of labelled data. In [52], GPT2 was fine-tuned to perform the functionalities of all the components present in the pipeline approach. The challenge addressed in this paper is the issue of interpretability present in previous work. Previously, only the system response would be generated by the end-to-end system. In this paper, in addition to the system response, the dialogue states and system acts are also generated which help in

understanding the model outputs and results. In [53], also the GPT2 model was fine-tuned to perform the different functionalities of the task-oriented dialogue system. The approach suggested in this work is the use of all the intermediate outputs of the model such as the belief state, database results, and system act instead of just having the user utterance and the system response. Based on this, the GPT2 model would be fine-tuned on the entire dialogue session. The results of this approach showed the importance of the added intermediate information on the performance of the model as the results achieved outperformed previous state-of-the-art models not having this added information.

2.2 Arabic Task-Oriented Dialogue Systems

Much less work has been done in the development of task-oriented chatbots in Arabic. In [54], a question-answering chatbot was developed by retraining the ALICE chatbot [55] using the Qur'an. The input of the user is a set of Arabic words, and the output is the set of Ayahs from the Qur'an which contain the set of words mentioned by the user. The approach used in this work is AIML-based consisting of three steps which are creating the frequency list of the Qur'an first and then creating the template files and finally applying, restructuring, and generating the AIML files. In [56], also a question-answering chatbot was proposed by retraining the ALICE chatbot [55] using the Qur'an as the training data. The input of the user is a set of words in English and the output is the set of Ayahs in both the English and in Arabic languages which contain the input words. The framework used was by first creating the frequency list, then creating patterns, and finally rearranging the patterns and generating the AIML files. In [57], a question-answering chatbot in the medical field was developed. The chatbot was trained using a set of 412 Arabic questions and answers that cover five domains. The methodology used in this work is matching the keywords in the user's input to the ones in the corpora. The first word and the most significant word approach were used where the first word in the question is the classifier, and the most significant word is the one with the smallest frequency in the question. Also, in [58], the same approach was used but with retraining ALICE using a dataset of the frequently asked questions in the School of Computing at the University of Leeds.

In [59], a web-based Arabic Conversational Tutoring System named Abdullah is proposed. This chatbot teaches children about Islam including the Qur'an and the Hadith. The chatbot deals with different topics and asks and answers questions. It also includes responses which are figures and sounds instead of only text. The methodology used to develop the chatbot is based on pattern matching. This consists of creating a knowledge base with all topics to be discussed, the Conversational Agent to generate the responses and the Tutorial Knowledge Base to manage the information between the learners. In [60], a question-answering

chatbot was developed to help the students in the Applied Science University in Jordan. The chatbot was developed based on a scripting engine and a scripting language which is rule-based. The chatbot can handle different topics which are divided into context. Each context has its own rules, which are made up of patterns, and responses, which are based on the user's input. The chatbot depends on the matching techniques which link the input of the user to the scripted patterns. In [61], a mobile-based chatbot was developed for the Android platform which is based on ArabChat [60]. In [62], an Enhanced ArabChat was developed which is an enhancement of ArabChat [60]. This enhancement tackled two issues that existed previously; the first issue was not being able to differentiate between questions and non-questions and the second issue is not being able to target several topics requiring several rules simultaneously. These issues were fixed by performing changes in the scripting language and the knowledge base used. In [63], an educational chatbot for tutoring children with ASD in Arabic was developed. The methodology used is based on both the Arabic Pattern Matching and the Arabic Short Text Similarity in order to match the question to the responses from the database. This chatbot can respond in different ways including visuals and sound effects based on the child's specific needs. In [64], an Arabic dialogue system was proposed that is composed of the parser and the dialogue manager where the parser was implemented using the Government and Binding theory.

Deep learning was also used in some recent work. In [65], the NLU component of the Arabic task-oriented dialogue system was implemented for the task of home automation. For the task of intent classification, CNNs and LSTMs were used and for the task of slot filling the bidirectional LSTM was used. In [66], a dialogue act recognition model was created for Levantine Arabic. The training was done on a created dataset for the domains of restaurant ordering and airline ticketing. Several classification models were implemented, and the best results were achieved using the Support Vector Machine (SVM) model. In [67], a dialogue act classification model was also implemented using the SVM classifier. The model was trained using a dataset collected from Egyptian call centers. In [68], a hybrid rule-based and data-driven approach was proposed for the implementation of Arabic task-oriented dialogue systems for the flight booking domain. The pipeline approach was used for the implementation of the dialogue system where the hybrid approach was used for training the NLU component.

The main challenge facing the development of task-oriented chatbots in Arabic is the scarcity of available datasets for training. However, what encourages the work in this direction is the development of language models in Arabic such as AraBERT [8] and AraGPT2 [6] that were trained on huge datasets and that achieved results comparable to English. These language models can greatly help in the development of Arabic task-oriented chatbots.

Chapter 3

Methodology - End-to-End Approach

3.1 End-to-End Approach

To train an end-to-end task-oriented dialogue system, the use of the pretrained language generation model AraGPT2 was proposed. We propose the development of an Arabic Task Oriented chatbot using Cascade of Generative Pre-trained End-to-End Language Model. In the following sections, an overview will be first given on the AraGPT2 model and then the proposed end-to-end approach will be explained.

3.1.1 AraGPT2

AraGPT2 [6] is a pretrained Arabic language generation model which has the same architecture as GPT2 [50]. AraGPT2 is a Transformer-based model which was trained on 77GB of Arabic data collected from several data sources. AraGPT2 is available in different sizes and the available models are the Base, Medium, Large, and Mega. The AraGPT2 used in this work is the Base model which has a context size of 1024, an embedding size of 768, 12 heads, 12 layers, and was trained using the LAMB optimizer. Training AraGPT2 followed the same process as that used for training GPT2.

3.1.2 AraGPT2 for End-to-End Task-Oriented Chatbots

Fine-Tuning AraGPT2

The suggested approach for training an end-to-end task oriented chatbot is to use AraGPT2. AraGPT2 is fine-tuned on a dialogue session level and hence each input includes all the turns in the session. This approach is an interpretable approach as it includes all the components which exist in the pipeline approach

which are the user utterance, the belief state, the database result, the system act, and the system response. In this way, the results can be easily analyzed and evaluated. In order to fine-tune AraGPT2 on the dialogue session data, the training data should include the user utterance (U), the belief state (B), the database result (D), the system act (A), and the system response (R) and this different information should be separated by special tokens in order to distinguish between them. The user utterance is surrounded by the tokens $\langle \text{sos}_u \rangle$ and $\langle \text{eos}_u \rangle$, the belief state by $\langle \text{sos}_b \rangle$ and $\langle \text{eos}_b \rangle$, the database result by $\langle \text{sos}_d \rangle$ and $\langle \text{eos}_d \rangle$, the system act by $\langle \text{sos}_a \rangle$ and $\langle \text{eos}_a \rangle$, and the system response by $\langle \text{sos}_r \rangle$ and $\langle \text{eos}_r \rangle$. These tokens are then added to the special tokens of the GPT2 tokenizer. An example of the format of the input data is shown in Figure 3.1 where the different components are colored in different colors.

Figure 3.1: Example of GPT2 Training Sequence

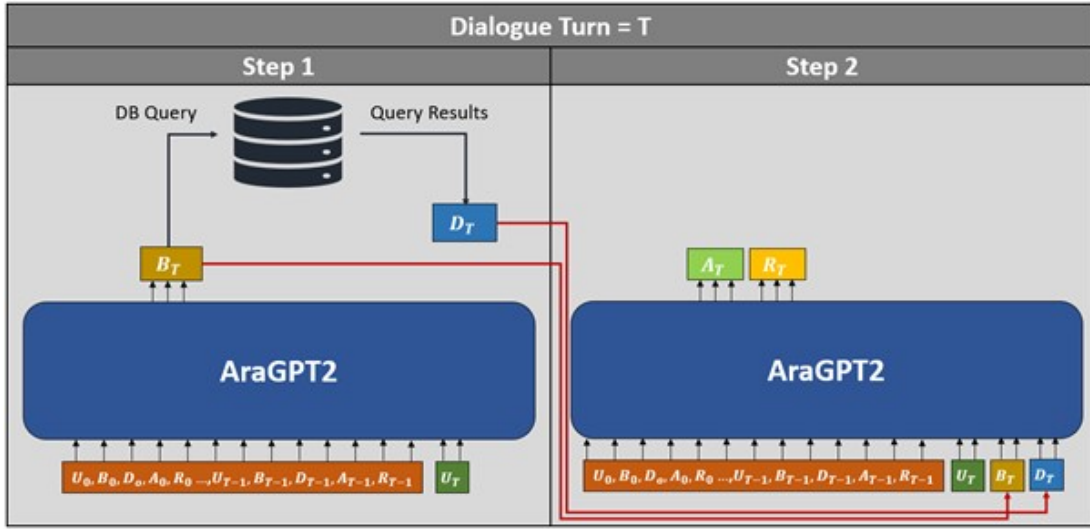
Fine-tuning AraGPT2 is done based on the Language Modeling objective of maximizing the probability of next word prediction:

$$L = \sum_i \log P(w_i | w_1, \dots, w_{i-1}) \quad (3.1)$$

After fine-tuning AraGPT2 using the training data of all the sessions, the next step is the generation of these components. The generation process resembles the process followed in the pipeline approach. The first step is to generate the belief state (B) based on the user utterance (U). Then the generated belief state is used to query the database and the results of the query generated are the database results (D). Then based on the user utterance, the generated belief state, and the database result, the system act and the system response are generated. The context input of every turn also includes all the previous turns in the dialogue session. Hence, in general, the overall workflow is as shown in Figure 3.2. For every dialogue turn T, the history of all the previous turns $(U_0, B_0, D_0, A_0, R_0, \dots, U_{T-1}, B_{T-1}, D_{T-1}, A_{T-1}, R_{T-1})$ and the user

utterance (U_T) are used to generate the belief state (B_T). Then the generated belief state is used to query the database and based on the query results, the database result (D_T) is generated. Then based on the history of all previous turns ($U_0, B_0, D_0, A_0, R_0, \dots, U_{T-1}, B_{T-1}, D_{T-1}, A_{T-1}, R_{T-1}$), the user utterance (U_T), the generated belief state (B_T), and database result (D_T), the system act (A_T) and the system response (R_T) are generated.

Figure 3.2: AraGPT2 End-to-End Task Oriented Dialogue System Workflow



Data Delexicalization

To train AraGPT2, it is essential to use delexicalized system responses in order to train the model on value-independent parameters [53]. The delexicalization process consists of replacing specific slot values by placeholders. For example, in Figure 3.3 the hotel name "هوليداي إن كامبريدج" was replaced by the placeholder [وحدة - الاسم]. This allows for generalization while training the model. The placeholders are then replaced by specific values based on the database results. A domain-adaptive delexicalization method was used as the one suggested by [53] which uses generalized placeholders instead of domain-specific ones. In addition to that, generalized slots are used instead of domain-specific ones in the

belief states and system acts which also helps the model learn value-independent parameters.

Figure 3.3: Example of Response Delexicalization

Original Response	يبدو وكأنه هوليداي إن كامبريدج سوف تعمل بشكل جيد بالنسبة لك هل تريد مني أن أحجز ذلك؟
Delexicalized Response	يبدو وكأنه [وحدة_الاسم] سوف تعمل بشكل جيد بالنسبة لك هل تريد مني أن أحجز ذلك؟

3.2 Dataset Created

As there are no resources available in the Arabic Language for training end-to-end task-oriented dialogue systems, resources had to be created. For this purpose, the MultiWOZ 2.0 dataset was translated to Arabic. In the following sections, first an overview will be given on the MultiWOZ dataset and then the approach used to translate the MultiWOZ dataset will be explained.

3.2.1 MultiWOZ Dataset

The Multi-Domain Wizard-of-Oz (MultiWOZ) dataset [7] is a large-scale multi-turn human-to-human conversational dataset which spans over multiple domains. The dataset consists of 10438 dialogues which span over seven domains which are hotel, restaurant, attraction, hospital, train, taxi, and police. The data split used for the train, validation, and test sets is 8438, 1000, and 1000 respectively which is the random split suggested in [7]. All the dialogues in the validation and test sets were chosen to be fully successful dialogues in order to have a fair evaluation.

Dialogues in the MultiWOZ dataset vary in length and complexity as each dialogue can cover between 1 to 5 domains. Hence each dialogue can be either single-domain or multi-domain. There are 3406 dialogues which are single-domain and 7032 dialogues which are multi-domain with the number of domains ranging from 2 to 5 domains. The complexity and multi-domain nature of the MultiWOZ dataset allows it to represent natural conversations where a user can ask questions about several domains and request bookings in several domains. In addition to the multi-domain aspect, the number of turns in the dialogue vary where more than half of dialogues have more than 10 turns.

The MultiWOZ 2.0 dataset is used in the format suggested by [53] which applies the domain-adaptive delexicalization explained in the previous section. In addition to the dialogue data, the MultiWOZ dataset also includes database

files for every domain in the seven included domains in the dialogues. We will be explaining the format of the database and dialogue data in the following sections.

Database

Each domain database contains several entries for possible available options in the specific domain. In each entry, all the information related to it are available. However, the included information varies between the domains as the information required for hotels is different than the one required for hospitals, for example. Figure 3.4 shows an example of one hotel available in the hotel database.

Figure 3.4: Example Database Entry

Entry Slot	Entry Value
name	alpha - milton guest house
postcode	cb41xa
internet	no
id	4
price	Double: 80, Single: 45
address	63 milton road
area	north
location	[52.2173388888889, 0.127638888888889]
parking	no
stars	3
pricerange	moderate
type	guest house
phone	1223311625
takesbookings	yes

Dialogue Data

Each dialogue in the dataset consists of a goal and the turn logs. The goal includes mainly the requestable and informable slots of the dialogue and any additional information regarding the booking status in case the booking failed. On the other hand, the dialogue turn logs include the user utterance, dialogue act, system act, and response for every turn in the dialogue. Figure 3.5 shows three turns from an example entry in the dataset showing how the dialogue act, system act, and response are updated after every new user utterance and hence with the addition of new constraints. The example also shows the hotel that was chosen from the database which satisfies the constraints of the user in terms of the area, stars, internet, and type. The format of each turn in the logs is as follows:

User Utterance The user utterance.

User Utterance Delexicalized The user utterance after applying delexicalization and replacing the specific slot values by placeholders. The placeholders are added inside brackets.

Dialogue Act The dialogue act in the form of *[domain1] slot value slot value. . . [domain2] slot value slot value. . .*

System Act The system act in the form of *[domain1] [system_act] slot slot. . .* where the system act can be either request, inform, recommend, select, . . .

System Response The system response based on the system act, and it is also delexicalized with specific slot values replaced by placeholders in brackets.

3.2.2 Translation of the MultiWOZ Dataset

In order to translate the MultiWOZ dataset, the Microsoft Translator API was used. However, a specific approach had to be used for translation since several formats exist in the dataset.

Database

To translate the database, all the slots and values should be translated. Regarding the slots, the translation should always be the same since they should be fixed. For that, a fixed translation was added to the slots available. Regarding the values, all values should be translated individually. However, some values should not be translated and that is in case the values were numbers or a list of number or a phone number. For example, Figure 3.6 shows an example of the translated hotel entry shown previously in Figure 3.4 in English.

Dialogue Data

To translate the dialogue data, different approaches had to be implemented for the goals and the logs.

Domains The names of the seven available domains are translated to Arabic in a fixed way and replaced in all instances according to the mappings in Table 3.1.

Figure 3.5: Example Dialogue from MultiWOZ dataset

Dialogue ID: sng01669			
Goals		Hotels Database	
Domain	Hotel		
Informable Slots	area		east
	internet		yes
	stars		2
	type		hotel
Requestable Slots	parking		
	address		
	phone		
Name	express by holiday inn cambridge		
Phone	01223866800		
Postcode	cb13lh		
Pricerange	expensive		
Stars	2		
Type	hotel		
Area	east		
Internet	yes		
Parking	yes		
Dialogue Turn Logs			
Turn 1	User Utterance	I am looking for a place to stay. the hotel should include free wifi and should be in the type of hotel	
	User Utterance Delexicalized	I am looking for a place to stay. the [value_type] should include free wifi and should be in the type of hotel	
	Dialogue Act	[hotel] internet yes type hotel	
	System Act	[hotel] [inform] choice type [request] area price	
	System Response	there are [value_choice] [value_type] that fit that description. do you have a preference as to area or price range?	
Turn 2	User Utterance	I would like something in the east with 2 stars.	
	User Utterance Delexicalized	I would like something in the [value_area] with [value_stars] stars.	
	Dialogue Act	[hotel] internet yes type hotel area east stars 2	
	System Act	[hotel] [inform] name [offerbook]	
	System Response	it looks like [value_name] will work well for you. would you like me to book that?	
Turn 3	User Utterance	does it have free parking? can you provide me with the contact details of this hotel?	
	User Utterance Delexicalized	does it have free parking? can you provide me with the contact details of this [value_type]?	
	Dialogue Act	[hotel] internet yes type hotel area east stars 2	
	System Act	[hotel] [inform] phone parking	
	System Response	yes, it offers free parking. the phone number is [value_phone].	

Goals To translate the goals, the translation of the slots should be fixed and hence a mapping of all available slots from English to Arabic was created to translate each slot. Regarding the values, each value was then translated using the Microsoft Translator API. Hence the translation was done word by word as shown in Figure 3.7.

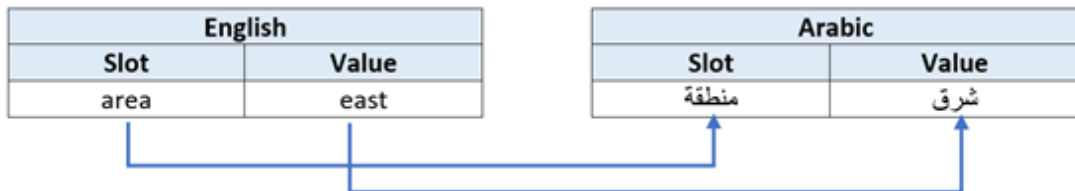
Figure 3.6: Example Translated MutliWOZ Database Entry to Arabic

Entry Value	Entry Slot
ألفا - ميلتون بيت الضيافة	الاسم
cb41xa	الرمز البريدي
لا	انترنت
4	رقم
Double: 80, Single: 45	سعر
63 ميلتون الطريق	عنوان
شمال	منطقة
[52.2173388888889, 0.127638888888889]	موقع
لا	موقف
3	نجوم
المعتدل	نطاق السعر
بيت الضيافة	نوع
1223311625	هاتف
نعم	يأخذ الحجوزات

Domain English	Domain Arabic
Restaurant	مطعم
Hotel	فندق
Attraction	جذب
Hospital	مستشفى
Train	قطار
Taxi	تاكسي
Police	شرطة

Table 3.1: Domain Translation Mappings

Figure 3.7: Example of MutliWOZ Goals Translation



Logs In the dialogue logs, the user utterance, user utterance delexicalized, dialogue act, system act, and system response should be translated and each one of these requires a different translation method.

User utterance The user utterance is translated as a whole sentence and the alignments resulting from the translation are saved to be used in the translation of the delexicalized user utterance. An example of a translated user utterance is shown in Figure 3.8.

Figure 3.8: Example of Translated User Utterance

Original	I would like something in the east with 2 stars
Translated	أود شيئاً في الشرق مع 2 نجوم

Delexicalized user utterance To translate the delexicalized user utterance, the alignments resulting when the user utterance was translated should be used. In the delexicalized user utterance, some of the words in the sentence are replaced by specific placeholders. In order to replace the Arabic words by placeholders, these Arabic words should be linked with their English words. Then if the word in English was replaced by a placeholder, its linked Arabic word should be replaced by the same placeholder. To do that, the word alignment option available in the Microsoft Translator API was used. This word alignment option returns for every translated sentence, the link for every translated word with the original word. An example of the alignment returned with the translated sentence is shown in Figure 3.9. The alignment results include the link between every word or entity in English with its translated word or entity in Arabic in terms of index position. For example, the alignment 30:33-12:16 shows that the word “east” which is in the index positions 30 to 33 in the English sentence is the translation of the word الشرق which is in the index positions 12 to 16 in the Arabic sentence. After getting the alignment results, the delexicalized user utterance in English is used to replace the words in Arabic by placeholders. So, in case a word in English is replaced by a placeholder, this same placeholder is linked to this word’s translation in Arabic. An example of this replacement is shown in Figure 3.10. In this example, as was shown in the alignment that the

word “east” is mapped to the word الشرق, the placeholder of the word “east” which is [value_area] replaced the word الشرق in Arabic. All words which are not replaced by placeholders in English were kept the same in Arabic. After the Arabic delexicalized user utterance was created, the next step is to replace the placeholders that are in English with their Arabic translation. This translation, however, should be fixed since these are fixed slots used in the dataset. Hence, a dictionary was used to map the English placeholders to Arabic placeholders. After replacing the placeholders, the results will be as shown in Figure 3.11. The same format used in English for the placeholders was used for Arabic which is adding the placeholder inside brackets.

Figure 3.9: Example of Alignment Results of the Translated User Utterance

Alignment		0:0-0:2 13:21-4:7 23:24-9:10 30:33-12:16 35:38-18:19 40:40-21:21 42:46-23:26				
0:0-0:2	13:21-4:7	23:24-9:10	30:33-12:16	35:38-18:19	40:40-21:21	42:46-23:26
أود - ا	شيئا - something	في - in	الشرق - east	مع - with	2 - 2	نجوم - stars

Figure 3.10: Example of creating the Delexicalized User Utterance in Arabic

			نجوم	2	مع	الشرق	في	شيئا	أود
			نجوم	[value_stars]	مع	[value_area]	في	شيئا	أود
I	would	like	something	in	the	east	with	2	stars
I	would	like	something	in	the	[value_area]	with	[value_stars]	stars

Dialogue Act The dialogue act is in the form *[domain] slot value... [domain] slot value....* In order to translate this, the translation should be done word by word. Then every word should be checked and in case the word was identified to be one of the defined domains or slots, the translation should be done using a dictionary mapping the English domains and slots to Arabic. On the other hand, if the word was not identified to be a domain or slot, the translation using the Microsoft Translator API should be done. An example of translating the

Figure 3.11: Example of Replacing the placeholders in the Arabic Delexicalized User Utterance

English Placeholders	نجوم	[value_stars]	مع	[value_area]	في	شينا	أود
Arabic Placeholders	نجوم	[وحدة_نجوم]	مع	[وحدة_منطقة]	في	شينا	أود

dialogue act is shown in Figure 3.12. As shown in the example, the translation is done word by word and depending on the word identification, the word is either mapped to its specific translation in case of being a domain or slot or directly translated in case it was a value. For example, the word “area” is a defined slot and hence using the predefined map, the word was translated to **منطقة**. However, the word “east” is a value, and it was translated to **الشرق** using the Microsoft Translator API.

Figure 3.12: Example of Dialogue Act Translation

English Dialogue Act	[hotel]	internet	yes	type	hotel	area	east	stars	2
Word Identification	<i>domain</i>	<i>slot</i>	<i>value</i>	<i>slot</i>	<i>value</i>	<i>slot</i>	<i>value</i>	<i>slot</i>	<i>value</i>
Word by Word Translation	[فندق]	انترنت	نعم	نوع	فندق	منطقة	شرق	نجوم	2
Arabic Dialogue Act	[فندق] [انترنت نعم نوع فندق منطقة شرق نجوم 2]								

System Act The system act is in the form $[domain1] [system_act] slot slot \dots$. Hence, the translation of the system act should be done word by word and since all words are predefined slots or domains or system acts, translation should be fixed and hence using a predefined mapping from English to Arabic. An example of translating the system act is shown in Figure 3.13 where each word in the English system act was mapped with a fixed translation in Arabic.

For example, the system act “inform” was mapped to the Arabic system act إبلاغ. Also, the same format convention was used for adding the domain name and system act in brackets.

Figure 3.13: Example of System Act Translation

English System Act	[hotel] [inform] name [offerbook]
Arabic System Act	[فندق] [إبلاغ] الاسم [حجز عرض]

System Response The system response is dellexicalized and hence to translate it to Arabic, the sentence was partitioned based on the placeholders and different partitions were translated separately. An example of translating the system act is shown in Figure 3.14. As shown, the system response was partitioned based on the placeholders into three parts and each part is translated separately. However, the translation of the placeholders was done using predefined mappings to make sure translations are always the same as these placeholders are fixed. Finally, all the partitions are combined into one sentence as shown in the last row in Figure 3.14.

Figure 3.14: Example of System Response Translation

English System Response	it looks like [value_name] will work well for you. would you like me to book that?		
Partitioned English System Response	it looks like	[value_name]	will work well for you. would you like me to book that?
Partitioned Arabic System Response	يبدو وكأنه	[وحدة_الاسم]	سوف تعمل بشكل جيد بالنسبة لك هل تريد مني أن أحجز ذلك؟
Arabic System Response	يبدو وكأنه [وحدة_الاسم] سوف تعمل بشكل جيد بالنسبة لك هل تريد مني أن أحجز ذلك؟		

Figure 3.15 shows a completely translated dialogue from the MultiWOZ dataset which is the translation of the previous example shown in Figure 3.5. It can be seen that the translation is accurate and the same format and content were preserved after translation.

Figure 3.15: Example Translated Dialogue from MultiWOZ dataset

Dialogue ID: sng01669		
Goals		Hotels Database
Domain	فندق	
Informable Slots	انترنت	نعم
	منطقة	شرق
	نجوم	2
	نوع	فندق
Requestable Slots	وقوف السيارات	
	عنوان	
	الهاتف	
الاسم	عبران طريق عطلة نزل كامبريدج	
هاتف	01223866800	
الرمز البريدي	cb13lh	
نطاق السعر	مكلف	
نجوم	2	
نوع	فندق	
منطقة	شرق	
انترنت	نعم	
موقف	نعم	
Dialogue Turn Logs		
Turn 1	User Utterance	أنا أبحث عن مكان للإقامة يجب أن يشمل الفندق خدمة الاتصال اللاسلكي بالإنترنت عبر تقنية واي فاي مجانية وينبغي أن يكون في نوع الفندق
	User Utterance Delexicalized	أنا أبحث عن مكان للإقامة يجب أن يشمل [وحده_نوع] خدمة الاتصال اللاسلكي بالإنترنت عبر تقنية واي فاي مجانية وينبغي أن يكون في نوع [وحده_نوع]
	Dialogue Act	[فندق] انترنت نعم نوع فندق
	System Act	[فندق] [إبلاغ] خيار نوع [طلب] منطقة سعر
	System Response	هناك [وحده_خيار] [وحده_نوع] التي تناسب هذا الوصف هل لديك تفصيل للمنطقة أو النطاق السعري؟
Turn 2	User Utterance	أود شيئاً في الشرق مع 2 نجوم
	User Utterance Delexicalized	أود شيئاً في [وحده_منطقة] مع [وحده_نجوم] نجوم
	Dialogue Act	[فندق] انترنت نعم نوع فندق منطقة شرق نجوم 2
	System Act	[فندق] [إبلاغ] الاسم [حجز عرض]
	System Response	يبدو وكأنه [وحده_الاسم] سوف تعمل بشكل جيد بالنسبة لك هل تريد مني أن أحجز ذلك؟
Turn 3	User Utterance	هل لديها مواقف مجانية للسيارات؟ هل يمكنك تزويدي بتفاصيل الاتصال بهذا الفندق؟
	User Utterance Delexicalized	هل لديها مواقف مجانية للسيارات؟ هل يمكنك تزويدي بتفاصيل الاتصال بهذا الفندق؟
	Dialogue Act	[فندق] انترنت نعم نوع فندق منطقة شرق نجوم 2
	System Act	[فندق] [إبلاغ] هاتف موقف
	System Response	نعم ويوفر مواقف مجانية للسيارات رقم الهاتف [وحده_هاتف]

Chapter 4

Evaluation and Results - End-to-End Approach

4.1 Training Details

The implementation of our model was done using the “aragpt2-base” model with 135M parameters. The model has 12 layers, 12 heads, and a maximum context length of 1024. AdamW optimizer was used which is a stochastic optimization method. The learning rate used is 1e-04, the seed used is 11, and the number of epochs is 20. Data Pre-processing was done using the AraBERTPreprocessor for aragpt2-base and tokenization was done using the GPT2Tokenizer. The data split used was 8434 dialogues for training, 999 dialogues for validation, and 1000 dialogues for testing.

4.2 Error Measures

Several error measures were used to automatically evaluate the results of the trained model. Evaluation is based on different criteria related to the response generation quality, the policy optimization, and the performance of the complete chatbot in an end-to-end way. In order to evaluate these different criteria, the error measures used are Inform, Success, and BLEU. The Inform error measure evaluates if the system has provided a correct entity. The Success error measure evaluates if the system was able to answer all the requested information and if all the answered information matches the user’s goal. The BLEU error measure evaluates how natural and fluent are the generated responses. A combined score is also used which was suggested in [7]. This combined score combines the three mentioned error measures as follows:

$$CombinedScore = (Inform + Success) \times 0.5 + BLEU \quad (4.1)$$

4.3 Results

To evaluate the performance of the fine-tuned AraGPT2, the model was tested in several scenarios. The difference between the testing scenarios is the use of either the ground truth or the generated results for the different components. In the real scenario, all the components used will be generated. However, to assess the quality of the performance of the chatbot in a specific aspect such as response generation, using the ground truth belief state and system act would result in a better evaluation of only the response generation aspect. Three scenarios were considered to evaluate the quality of the response generation, policy optimization, and the end-to-end system. The different settings chosen for these scenarios are shown in Table 4.1. The response generation aspect is evaluated by using the ground truth values for the belief state, database result, and system act and the generated system response is used in the context. For the evaluation of the policy optimization, the ground truth values are used for the belief state and database result and the generated values are used for the system act and the system response. To evaluate the performance of the end-to-end system, all the components used in the context are the generated ones. In the following sections, the evaluation results of these three scenarios will be shown. The results shown include the delexicalized responses which enables the evaluation of the system in a generalized condition. However, these delexicalized responses in a natural workflow will get filled before being sent to the user according to the database results.

Scenario	Belief State	Database Result	System Act	System Response
Response Generation	Ground Truth	Ground Truth	Ground Truth	Generated
Policy Optimization	Ground Truth	Ground Truth	Generated	Generated
End-to-End System	Generated	Generated	Generated	Generated

Table 4.1: Settings of Different Evaluation Scenarios

4.3.1 Response Generation

As mentioned previously, to evaluate the response generation quality of the model, the ground truth values for the belief state, database result, and system act will be used. However, the generated system response will be used in the dialogue turn context. The model was tested on both the validation and test datasets and the results are shown in Table 4.2. The model was able to achieve an inform score of 79.2%, a success of 76.7%, a BLEU score of 23.6%, and a combined score

of 101.55%. Figures 4.1 and 4.2 show two examples of dialogues from the test dataset. The example in Figure 4.1 shows a dialogue where the chatbot is helping the user look for a hotel and a specific restaurant. The quality of the responses achieved is very similar to those of the ground truth ones where all the required information is available in the generated response. In Figure 4.2, the example shows a case where the booking couldn't be done based on the user constraints, but the generated responses were of similar quality to the ground truth ones which helped the user change the constraints and then successfully book a room in the hotel.

Dataset	Inform	Success	BLEU	Combined Score
Validation	77.38	76.38	23.15	100.03
Test	79.20	76.70	23.60	101.55

Table 4.2: Response Generation Results

Figure 4.1: Response Generation Results - Example 1

أنا أبحث عن فندق	User
هناك ما مجموعه [وحدة_خيار] فندق [وحدة_الاسم] هو خيار جيد مع [وحدة_نجوم] تصنيف النجوم هل تريد رقم هاتف؟	GT Response
هناك [وحدة_خيار] فندق مع [وحدة_نجوم] تصنيف النجوم [وحدة_الاسم] و [وحدة_الاسم] هل تريد المزيد من المعلومات عن 1 من هذه؟	Generated Response
أنا فعلا بحاجة إلى المكان الذي لديه خدمة الاتصال اللاسلكي بالإنترنت عبر تقنية واي فاي مجانا ويقع في الجزء الجنوبي من المدينة ويجب أن يكون فندقا وليس بيت ضيافة	User
[وحدة_الاسم] هل [وحدة_خيار] الفندق الذي يطابق المعايير الخاصة بك هل ترغب في حجز إقامة؟	GT Response
هناك [وحدة_خيار] الفندق التي تلبى المعايير الخاصة بك [وحدة_الاسم] و [وحدة_الاسم] هل تريد مني أن أحجز لك 1 من هذه؟	Generated Response
وأود أن مجرد الحاجة إلى تصنيف النجوم ورقم الهاتف وشكرا لكم	User
مطلقا! [وحدة_الاسم] هو [وحدة_نجوم] نجم الفندق ورقم هاتفهم هو [وحدة_هاتف] هل هناك أي شيء آخر يمكنني مساعدتك به؟	GT Response
[وحدة_الاسم] قد [وحدة_نجوم] نجوم رقم هاتفهم هو [وحدة_هاتف] هل هناك أي شيء آخر يمكنني مساعدتك به؟	Generated Response
نعم أحتاج إلى العثور على مطعم يسمى مطبخ البيئزا البقر وبار	User
يقع هذا المطعم في [وحدة_منطقة] على [وحدة_عنوان] ورقم هاتفهم [وحدة_هاتف] هل هناك أي شيء آخر يمكنني مساعدتك به؟	GT Response
يقع في [وحدة_منطقة] في [وحدة_عنوان] رقم الهاتف [وحدة_هاتف] هل هناك أي شيء آخر يمكنني مساعدتك به؟	Generated Response
أعتقد أنك تساعدني في كل شيء أحتاجه شكرا لك	User
أنا سعيد أنني يمكن أن تساعدك يكون يوما عظيما!	GT Response
شكرا لاستخدام خدمتنا اليوم!	Generated Response

Figure 4.2: Response Generation Results - Example 2

أنا في حاجة إلى مطعم هندي مكلف لتناول العشاء	User
هناك [وحدة_خيار] مثل هذا المطعم هل تريد أن تكون في جزء محدد من المدينة؟	GT Response
لدي [وحدة_خيار] خيارات لك هل هناك منطقة معينة من المدينة كنت مهتما؟	Generated Response
لا يهمني ما هو جزء من المدينة في وأود أن رقم الهاتف من فضلك	User
بالتأكيد أنا أسف لدي [وحدة_الاسم] رقم هاتفهم هو [وحدة_هاتف] هل تريد مني أن أحجز طاولة لك؟	GT Response
[وحدة_الاسم] رقم الهاتف هو [وحدة_هاتف] هل تريد مني أن أحجز لك طاولة؟	Generated Response
ليس في هذا الوقت أنا أبحث عن فندق 3 نجوم وإن كان لا يوجد بيت ضيافة	User
لقد وجدت [وحدة_الاسم] [وحدة_نوع] بل هو [وحدة_نجوم] فندق ستار هل تريد مني أن أحجزها لك؟	GT Response
[وحدة_الاسم] هو [وحدة_نجوم] نجم [وحدة_نوع] هل تريد مني أن أحجز لك؟	Generated Response
ليس الآن هو فندق غونفل في نفس النطاق السعري كحديقة الكاري؟	User
نعم بل هو [وحدة_سعر] [وحدة_نوع]	GT Response
بل هو [وحدة_سعر] [وحدة_نوع]	Generated Response
يمكنك حجز غرفة لشخصين لمدة ليلتين ابتداء من يوم الأحد؟	User
لم ينجح الحجز هل ترغب في محاولة ليوم آخر أو إقامة أقصر؟	GT Response
أنا أسف ولكن لا يوجد توافر لذلك الوقت هل ترغب في تجربة يوم مختلف أو أقصر إقامة؟	Generated Response
هل يمكنك تجربتها لليلة واحدة؟ إذا كان ذلك يعمل يرجى حجزه والحصول على الرقم المرجعي	User
كنت كل مجموعة ل [وحدة_إقامة] ليلة ([وحدة_يوم]) الرقم المرجعي هو [وحدة_المرجع] هل ترغب في تجربة موقع آخر ل [وحدة_يوم] الليل؟	GT Response
كنت قادرا على حجز حزبكم من [وحدة_اشخاص] من أجل [وحدة_إقامة] ليلة تبدأ [وحدة_يوم] رقم المرجع الخاص بك هو [وحدة_المرجع]	Generated Response
وداعا وهذا هو كل شيء لهذا اليوم	User
استمتع بإقامتك	GT Response
شكرا لاستخدام خدمتنا يكون يوم عظيم!	Generated Response

4.3.2 Policy Optimization

To evaluate the model performance on policy optimization, the ground truth values for the belief state and database result were used. However, the system act and the system responses used are the generated ones by the model. In this way, the model's ability on generating the system act based on the knowledge of the belief state and the database result will be evaluated and also consequently the generated response based on the system act. The model was tested on both the validation and test datasets and the results are shown in Table 4.3. The

model was able to achieve an inform score of 76.8%, a success of 72.4%, a BLEU score of 12.78%, and a combined score of 87.38%. An example of a successful conversation is shown in Figure 4.3. The user’s goal in this example is to book a taxi and the system was able to successfully request the destination from the user when the origin and time were given. After the destination was given from the user, the chatbot was able to successfully book a taxi and inform the user about the taxi type and phone number. An example of an unsuccessful conversation is shown in Figure 4.4. In this dialogue, the goal of the user is to book a table at a restaurant. In the second turn of the dialogue, the user requests to book a table in the recommended restaurant, the ground truth system act requests the day, time, and number of people for the booking. However, our model doesn’t request this information required for the booking and doesn’t book a table for the user even though the user requested a booking. In order to show how the database results which are queried with the results, the database results of the dialogue in Figure 4.4 are shown in Figure 4.5. In the results, there are 4 chosen restaurants which satisfy the requirements of the user regarding the price and the cuisine type. If the user does not choose a specific restaurant, a random choice will be suggested to the user.

Dataset	Inform	Success	BLEU	Combined Score
Validation	72.77	70.17	12.80	84.27
Test	76.80	72.40	12.78	87.38

Table 4.3: Policy Optimization Results

4.3.3 End-to-End System

To evaluate the model performance as an end-to-end system, all the components of the system to be used are the generated ones and not the ground truth. Hence, the model will generate the belief state, and then query the database and get the database results, then based on the results the system will generate the system act and the system response. This workflow resembles the real workflow of a task-oriented dialogue system. The model was tested on both the validation and test datasets and the results are shown in Table 4.4. The model was able to achieve an inform score of 79.6%, a success of 71.9%, a BLEU score of 12.26%, and a combined score of 88.01%. An example of a successful dialogue is shown in Figure 4.6 which is the same example shown previously in Figure 4.3 where the ground truth belief state was used. It can be seen that the results are the same because the belief state was accurately generated. Consequently, the system act and system response were also accurately generated based on the generated belief state and database results and hence the conversation was successful, and the booking was done. An example of an unsuccessful dialogue is shown in Figure

Figure 4.3: Policy Optimization Results - Example 1

أحتاج سيارة أجرة لاصطحابي في فندق أشلي للمغادرة بعد الساعة 10 45	User
[تاكسي] [طلب] الوجهة	GT System Act
[تاكسي] [طلب] الوجهة	Generated System Act
إلى أين تريد أن تذهب؟	GT Response
أين وجهتك؟	Generated Response
أنا بحاجة للوصول إلى مطعم 2 اثنين	User
[تاكسي] [إبلاغ] سيارة هاتف [عام] [طلبالمزيد]	GT System Act
[تاكسي] [إبلاغ] سيارة هاتف	Generated System Act
اكتمل الحجز! هل هناك أي شيء آخر يمكنني مساعدتك به اليوم؟ نوع السيارة المحجوزة [وحدة_سيارة] رقم الاتصال [وحدة_هاتف]	GT Response
اكتمل الحجز! نوع السيارة المحجوزة [وحدة_سيارة] رقم الاتصال [وحدة_هاتف]	Generated Response
شكرا! هذا كل شيء	User
[عام] [طلبالمزيد]	GT System Act
[عام] [وداعا]	Generated System Act
ويمكنني أيضا مساعدتك مع مناطق الجذب والقطارات إذا كنت مهتما	GT Response
شكرا لاستخدام خدماتنا	Generated Response

4.7 where in the first turn the model was able to correctly generate the belief state and accordingly generate the system act and system response which includes a request of the area and price range from the user. However, even after the user mentions that there are no preferences on the area and price range, the system keeps on asking about them in the next two turns. In the fourth turn, since the true user utterance is asking to book a hotel, the true system response asks the user regarding the number of days which is required to be able to book. However, the model made the booking without asking this question to the user which is by taking a random choice from the database results but without considering the number of days to be booked which will result in a wrong booking.

Dataset	Inform	Success	BLEU	Combined Score
Validation	76.28	70.57	12.06	85.48
Test	79.60	71.90	12.26	88.01

Table 4.4: End-to-End System Results

Figure 4.4: Policy Optimization Results - Example 2

وأود أن تجد مطعم التي تخدم الطعام الصيني الرخيص!	User
[مطعم] [إبلاغ] سعر منطقة الاسم [حجز عرض]	GT System Act
[مطعم] [إبلاغ] طعام سعر خيار [طلب] منطقة	Generated System Act
[وحدة_الاسم] يقع في [وحدة_منطقة] وهو [وحدة_سعر]! هل تريد مني أن أحجزها لك؟	GT Response
هناك [وحدة_خيار] [وحدة_سعر] [وحدة_طعام] مطعم هل هناك منطقة معينة من المدينة التي ترغب في تناول الطعام فيها؟	Generated Response
نعم تفضلوا وحجزوها ورجاء أرسلوا لي العنوان	User
[مطعم] [إبلاغ] عنوان [طلب] يوم وقت اشخاص	GT System Act
[مطعم] [إبلاغ] عنوان الاسم طعام [عام] [طلبالمزيد]	Generated System Act
العنوان هو [وحدة_عنوان] ما هو اليوم والوقت الذي ترغب في حجزه؟ كم عدد الناس؟	GT Response
[وحدة_الاسم] يقدم [وحدة_طعام] طعام وهي تقع في [وحدة_عنوان] هل هناك أي شيء آخر يمكنني مساعدتك به؟	Generated Response
أنا لست بحاجة لحجزه ما هو الرمز البريدي والمنطقة من فضلك؟	User
[مطعم] [إبلاغ] الرمز البريدي	GT System Act
[مطعم] [إبلاغ] الاسم منطقة الرمز البريدي	Generated System Act
الرمز البريدي هو [وحدة_الرمز البريدي]	GT Response
[وحدة_الاسم] يقع في [وحدة_منطقة] والرمز البريدي الخاص بهم [وحدة_الرمز البريدي]	Generated Response
شكرا جزيلاً	User
[عام] [أهلا بك] [وداعا]	GT System Act
[عام] [أهلا بك] [وداعا]	Generated System Act
مرحبا بك! يكون يوم عظيم! وداعا	GT Response
مرحبا بك وداعا	Generated Response

Figure 4.5: Database Results of Example 2

Database Results				
Information	Restaurant 1	Restaurant 2	Restaurant 3	Restaurant 4
طعام	الصينية	الصينية	الصينية	الصينية
نطاق السعر	رخيص	رخيص	رخيص	رخيص
الاسم	تشارلي تشان	منزل الأرز	النجم المحفوظ	البيت الذهبي

Overall, the results generated by the end-to-end system are accurate and in most cases the belief state, system act, and system response are accurately generated. The current evaluation is on the true user utterances and not utterances

Figure 4.6: End-to-End System Results - Example 1

Generated	Ground Truth	
أحتاج سيارة أجرة لاصطحابي في فندق أشلي للمغادرة بعد الساعة 10 45		User
[تاكسي] مغادرة 10 45 مغادرة أشلي فندق	[تاكسي] مغادرة 10 45 مغادرة أشلي فندق	Belief State
[تاكسي] [طلب] الوجهة	[تاكسي] [طلب] الوجهة	System Act
أين وجهتك؟	إلى أين تريد أن تذهب؟	System Response
أنا بحاجة للوصول إلى مطعم 2 اثنين		User
[تاكسي] مغادرة 10 45 مغادرة أشلي فندق الوجهة مطعم 2 اثنان	[تاكسي] مغادرة 10 45 مغادرة أشلي فندق الوجهة مطعم 2 اثنان	Belief State
[تاكسي] [إبلاغ] سيارة هاتف	[تاكسي] [إبلاغ] سيارة هاتف [عام] [طلبالمزيد]	System Act
اكتمل الحجز! نوع السيارة المحجوزة [وحدة_سيارة] رقم الاتصال [وحدة_هاتف]	اكتمل الحجز! هل هناك أي شيء آخر يمكنني مساعدتك به اليوم؟ نوع السيارة المحجوزة [وحدة_سيارة] رقم الاتصال [وحدة_هاتف]	System Response
شكرا! هذا كل شيء		User
[تاكسي] مغادرة 10 45 مغادرة أشلي فندق الوجهة مطعم 2 اثنان	[تاكسي] مغادرة 10 45 مغادرة أشلي فندق الوجهة مطعم 2 اثنان	Belief State
[عام] [وداعا]	[عام] [طلبالمزيد]	System Act
شكرا لاستخدام خدماتنا	ويمكنني أيضا مساعدتك مع مناطق الجذب والقطارات إذا كنت مهتما	System Response

generated in response to the system response and hence this disables the evaluation on the true flow of the conversation. However, even with the ground truth system responses, the results are accurate.

4.3.4 Comparison of Results

The results of our model trained using AraGPT2 and the translated MultiWOZ dataset are compared to the results achieved by UBAR [53] which was trained using the same approach using GPT2 and the English MultiWOZ dataset and shown in Figure 4.8. The results achieved using our model are comparable to the English results. The results achieved using the English dataset are better than the ones achieved using our translated dataset. However, considering that the dataset was translated and hence contains some errors and inaccuracies, the results are very close to the English ones. This shows that the quality of the dataset is good, and the trained model achieved comparable results to state-of-the-art in English.

Figure 4.7: End-to-End System Results - Example 2

Generated	Ground Truth	
	مرحبا أنا أبحث عن مكان للإقامة التي لديها خدمة الاتصال اللاسلكي بالإنترنت عبر تقنية واي فاي مجاناً ومواقف مجانية للسيارات	User
[فندق] موقف نعم انترنت نعم	[فندق] موقف نعم انترنت نعم	Belief State
[فندق] [إبلاغ] خيار [طلب] منطقة سعر	[فندق] [طلب] منطقة سعر	System Act
هناك [وحدة_خيار] مكان التي تلي تلك المعايير هل لديك نطاق سعري أو منطقة ترغب في البقاء فيها؟	هل لديك تفضيل للمنطقة أو النطاق السعري؟	System Response
	ليس لدي تفضيل ولكن أريد فندقاً وليس بيت الضيافة	User
[فندق] موقف نعم انترنت نعم نوع فندق	[فندق] موقف نعم انترنت نعم نوع فندق	Belief State
[فندق] [إبلاغ] خيار نوع [طلب] سعر	[فندق] [نصح] الاسم [إبلاغ] خيار [حجز عرض]	System Act
هناك [وحدة_خيار] [وحدة_نوع] التي تلي تلك المعايير هل لديك تفضيل على النطاق السعري؟	هناك [وحدة_خيار] مثل هذا الفندق أوصي [وحدة_الاسم] هل تريد مني أن أحجز لك غرفة؟	System Response
	واثق هذا سيكون رائعاً	User
[فندق] موقف نعم انترنت نعم نوع فندق	[فندق] موقف نعم انترنت نعم نوع فندق	Belief State
[فندق] [إبلاغ] خيار [طلب] سعر منطقة	[فندق] [طلب] إقامة اشخاص	System Act
هناك [وحدة_خيار] [وحدة_نوع] في المدينة هل لديك تفضيل على المنطقة أو النطاق السعري؟	كم عدد الأشخاص الذين سيكونون في حفلاتك وكم من الوقت ترغب في البقاء؟	System Response
	هناك 3 منا وسوف نصل يوم السبت	User
[فندق] موقف نعم انترنت نعم نوع فندق اشخاص 3 يوم السبت إقامة 3	[فندق] موقف نعم انترنت نعم نوع فندق	Belief State
[فندق] [حجز العرض] المرجع الاسم [عام] [طلب المزيد]	[فندق] [طلب] إقامة	System Act
حجزت لك غرفة في [وحدة_الاسم] رقم المرجع الخاص بك هو [وحدة_المرجع] هل هناك أي شيء آخر يمكنني مساعدتك به؟	وكم يوماً؟	System Response

Figure 4.8: Comparison of Our Results with State-of-the-Art

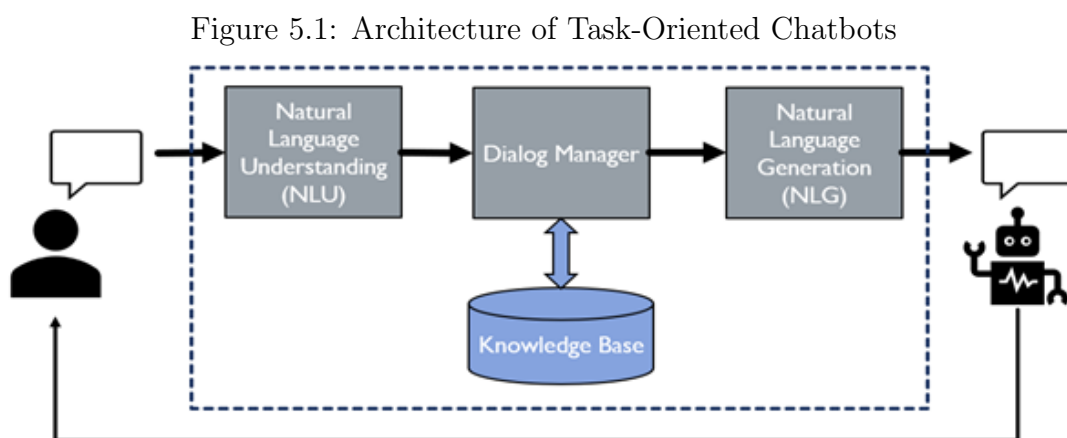
Model	Scenario	Inform	Success	BLEU	Combined Score
Our Model	Response Generation	79.20	76.70	23.60	101.55
UBAR		96.90	92.20	28.60	123.20
Our Model	Policy Optimization	76.80	72.40	12.78	87.38
UBAR		94.00	83.60	17.20	106.00
Our Model	End-to-End System	79.60	71.90	12.26	88.01
UBAR		95.40	80.70	17.00	105.10

Chapter 5

Methodology - NLU Component of the Pipeline Approach

5.1 Background on the Pipeline Approach Architecture

A typical architecture of task-oriented chatbots is as shown in Figure 5.1. The architecture is made up of three main components which are the Natural Language Understanding (NLU), the Dialog Manager which contains the Dialog state tracker and the Dialog policy learning, and the Natural Language Generation (NLG). These three components will be explained in the following sections.



5.1.1 Natural Language Understanding (NLU)

The Natural Language Understanding (NLU) component has two main objectives. The first objective is Intent Classification and the second objective is Slot Filling [69]. The intent classification task is to determine the goal of the user from the utterance. For example, in the flight booking domain, the intent of the user might be to Book a Flight or to Find a Flight. The slot filling task is to extract the required slots from the user utterance which the user wants to send to the chatbot and this is linked with the intent of the utterance. For example in Figure 5.2, for a user utterance **أريد حجز طائرة من بيروت الى عمان**, the intent of the user is to book a flight and the slots are extracted for each word in the utterance that the user wants the chatbot to understand and that is related to booking the flight. In this example, the user wants the chatbot to know that the origin of the flight is **بيروت** and the destination of the flight is **عمان** and these two words should be annotated in the user utterance.

Figure 5.2: Example NLU Output

User Utterance	عمان	الى	بيروت	من	طائرة	حجز	أريد
Slots	Destination		Origin				
Intent	Book a Flight						

In datasets where multiple domains exist, an additional task for the NLU is required which is the domain classification. This is necessary to classify to which domain the utterance is referring before determining the intent of the user and extracting the slots [69].

5.1.2 Dialog Manager (DM)

The Dialog Manager component is made of two main components which are the Dialog state tracker and the Dialog policy learning. The input to Dialog State tracker is the output of the NLU component and it is in the form of a dialog act [70]. The next section will be explaining the form of the dialog act and the two components of the Dialog Manager.

Dialog Act

The dialog act represents the interpretation of the user's utterance. Dialog acts have different forms but the most popular one is the CUED standard dialog act [71]. The CUED form of the dialog act is represented in the form of dialog act type followed by a list of act items:

$$ActType(a = x, b = y, \dots) \quad (5.1)$$

Where the ActType represents the type of the dialog act such as inform, request, confirm, and select. The act items $a=x$, $b=y$ represent the attribute-value pairs of the corresponding dialog act. For example, the dialog act can be INFORM(destination=عمان) which means that the type of the dialog act is to inform and the attribute to inform is that the destination country is عمان.

The dialog acts depend on the dialog system and the task of the chatbot [69]. For example, the dialog acts of a chatbot for flight booking are different than the ones for restaurant booking. Figure 5.3 shows examples of different dialog acts as mentioned in [71]. The system and user columns indicate whether the dialog act is applicable to the chatbot only, the user only, or to both. The dialog act items a , $b,..$ are however task-dependent in which they can be destination country and origin country in case of flight booking or cuisine type and price range in case of restaurant booking.

Figure 5.3: Examples of Dialog Acts

Dialog Act	System	User	Description
HELLO()	✓	✓	Start conversation
HELLO($a = x$, $b = y,..$)	✗	✓	Start conversation and give information $a = x$, $b = y,..$
INFORM($a = x$, $b = y,..$)	✓	✓	Give information $a = x$, $b = y,..$
REQUEST(a , $b = x,..$)	✓	✓	Request value for a given $b = x ...$

Dialog State Tracker

The main goal of the dialog state tracker is to determine the state of the user at each turn of the conversation [4]. This is done by determining the current state of the frame using the current slots from the user and also all the past constraints given. Using an example conversation shown in Figure 5.4 , the output of the dialog state tracker after every turn of the conversation will be as indicated in the figure.

The task of predicting the dialog act type is usually performed as a supervised classification task where the dialog act type is predicted using the current user utterance and the previous dialog acts. Also, one of the tasks of the dialog state tracker is to determine whether any of the values of the previously assigned slots have been changed in the current utterance and to update it if necessary [69]. Another important part of the Dialog state tracker is to perform generalization

Figure 5.4: Example Output of the Dialog State Tracker

Dialog State	Utterance	
INFORM(destination="عمّان")	أريد حجز طائرة الى عمّان	User
	حسناً. ما هو مكان الإقلاع؟	Chatbot
INFORM (destination="عمّان ", origin="بيروت ")	بيروت	User
	متى تاريخ السفر؟	Chatbot
INFORM (destination="عمّان ", origin="بيروت ", date="20 كانون الثاني")	20 كانون الثاني	User

to the user constraints. For example, the words **رخيص** and **سعر منخفض** both refer to the same meaning of cheap and hence they should be linked with the same output slot. This step is important to ensure a generalized Dialogue State Tracker component.

Dialog Policy Learning

The main job of the dialog policy learning is to determine the next action that the chatbot should do based on the current dialog state generated by the dialog state tracker [4]. So, the aim of the dialog policy is to determine the next action A_i based on all the previous chatbot (A) and user (U) actions:

$$\hat{A}_i = \underset{(A_i \in A)}{\operatorname{argmax}} P(A_i | A_1, U_1, \dots, A_{i-1}, U_{i-1}) \quad (5.2)$$

One of the advanced approaches used in dialog policy learning is reinforcement learning where the reinforcement learning systems gets rewarded at the end of the conversation if the dialog acts taken were correct and the goal of the user was achieved. However, a negative reward is given to the system in case the goal was not achieved correctly.

5.1.3 Natural Language Generation (NLG)

The main objective of the Natural Language Generation (NLG) component is to generate a natural system response based on the dialog act outputted from the dialog policy learning component [69]. An example output of the NLG component is shown in Figure 5.5 where the output of the dialogue act is to inform the user of the origin and destination of the booked flight. The objective of the NLG component is to generate a system response from the dialogue act as shown in the second row in the figure. However, training an NLG component using this data is hard as it is difficult to generate responses for all combinations of slots

in each of the different dialogue acts [69]. Consequently, it is recommended to use delexicalized system responses to train the NLG component in order to train the NLG component using more generalized data. Delexicalization is the replacement of specific words in the sentence by generic placeholders. For example, the word **بيروت** is replaced by the generic placeholder **الوجهة** in the system response. In that way, the data becomes more generalized, and the placeholder can be replaced by the actual values after the NLG component have generated the generalized system response with the placeholder.

Figure 5.5: Example Output of the NLG

Dialogue Act	INFORM (الوجهة = "عمّان", بيروت = الانطلاق)
System Response	وجدت رحلة من بيروت الى عمّان
Delexicalized System Response	وجدت رحلة من الانطلاق الى الوجهة

5.2 Natural Language Understanding (NLU) Training

5.2.1 Approaches

The first component in the pipeline approach of the task-oriented dialog system is the Natural Language Understanding (NLU) component and for that component we propose the Multi-task Chat Understanding Towards a Pipeline Approach. As mentioned earlier, the main tasks of the NLU component are the Intent Classification and the Slot Filling. In order to train the NLU component, two approaches are suggested. The first approach is training a separate model for each task and hence training one model for Intent Classification and another model for Slot Filling. The second approach is to jointly train one model on both tasks at the same time. In both approaches the pretrained language model AraBERT [8] is used. In the following sections, first the AraBERT pretrained language model will be introduced and then the two approaches will be explained.

AraBERT

The AraBERT [8] pretrained language model is based on the BERT model [22] with both having the same base configuration which is 12 encoder blocks, 768 hidden dimensions, 12 attention heads, 512 maximum sequence length. The BERT architecture is made up of multi-layer attention-based Bidirectional Transformer

Encoders. The AraBERT pretraining setup was based on the same setup used by BERT which consists of two main steps which are the Masked Language Modeling (MLM) and the Next Sentence Prediction (NSP). The MLM task helps in capturing the language properties by randomly masking some tokens to avoid a token observing itself. The NSP task is important to capture information in the settings of having sentence pairs.

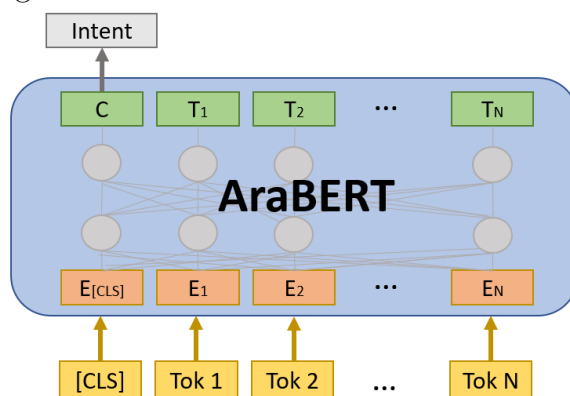
The input representation to AraBERT is the concatenation of word embeddings, positional embeddings, and segment embeddings. A special token is inserted as the first token for the input which is [CLS] and a special token is added as the final token which is [SEP]. The AraBERT pretrained language model have been evaluated on several NLU tasks such as classification and Named Entity Recognition tasks and the model was able to achieve state-of-the-art results compared to other approaches and language models [8].

AraBERT for Intent Classification

The first suggested approach is to train a separate model for each of the NLU tasks which are the Intent Classification and Slot Filling. For that, the pretrained language model AraBERT was fine-tuned on each of these tasks separately.

For the intent classification task, AraBERT was fine-tuned using user utterances as input while adding the special token [CLS] as the first token and the special token [SEP] as the last token. Then the output class label which is the intent would be the final hidden state of the first token in the input sentence which is the [CLS] token as shown in Figure 5.6.

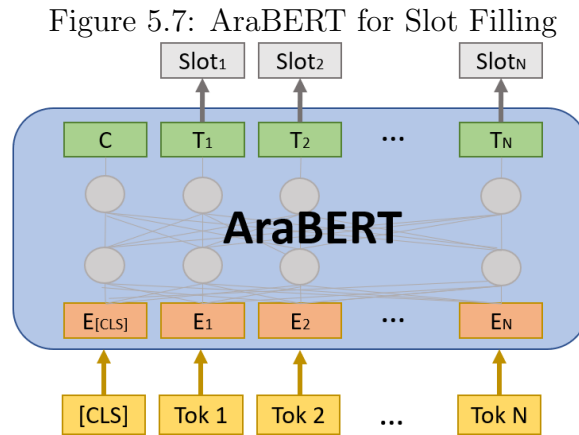
Figure 5.6: AraBERT for Intent Classification



AraBERT for Slot Filling

For the slot filling task, the same process of adding the first and final special tokens was applied. The input sentence has the same format as the one used for

intent classification. The difference in this task is the output where the output in this task would be the output of each token after the output vector passes through a classification layer that predicts the required slot labels as shown in Figure 5.7.



Joint AraBERT for Intent Classification and Slot Filling

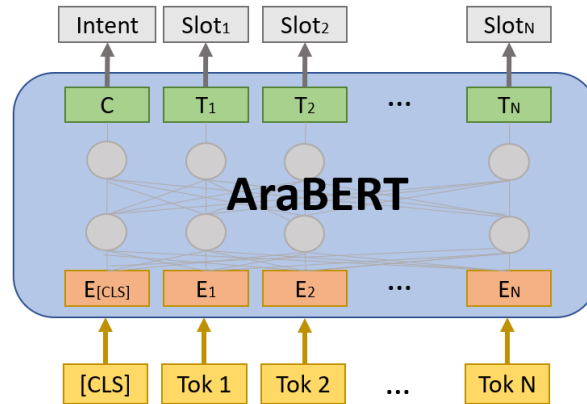
The second suggested approach is to train one model for both intent classification and slot filling. For that, AraBERT was fine-tuned on these two tasks jointly as shown in Figure 5.8. The input to the model is the user utterance with the special token [CLS] added as the first token and the special token [SEP] added as the final token. The predicted intent would then be the final hidden state of the first special token [CLS]. For the slot filling task, the final hidden states of all the tokens except the first token are fed into a softmax layer. The outputs would then be the predicted slot labels for every token. In order to train the model on both tasks of intent classification and slot filling, the total loss function to be minimized while fine-tuning is chosen to be:

$$TotalLoss = IntentLoss + Coefficient \times SlotLoss \quad (5.3)$$

Using this loss function, both the intent and slot loss are considered where the weight of the slot loss can be altered using the coefficient in the equation. A coefficient of 1 gives both the intent and slot loss equal weights.

In addition to the mentioned joint AraBERT architecture, another approach used is the addition of a Conditional Random Fields (CRF) layer on top of the joint AraBERT model. CRFs are structured prediction models and adding them on top of the model can help in the prediction of slots from the predictions of the surrounding words.

Figure 5.8: AraBERT for Joint Intent Classification and Slot Filling



5.2.2 Dataset Created

As there are no resources available in the Arabic Language for training the NLU component, resources had to be created. For this purpose, the ATIS dataset, which is a well-known dataset for NLU, was translated to Arabic. In the following sections, first an overview will be given on the ATIS dataset and then the approach used to translate the ATIS dataset will be explained.

ATIS Dataset

The Air Travel Information System (ATIS) dataset [9] is a well-known dataset used for training NLU models. The dataset includes user utterances of users requesting flight reservations and information. There are a total of 5871 utterances in the dataset where the same data division as [20] was used which divides the data to 4478 utterances for train, 500 utterances for evaluation, and 893 utterances for test.

Figure 5.9 shows an example sample in the ATIS dataset. Each entry in the dataset includes the user utterance, the intent of the utterance, and a slot for every word in the utterance. There are a total 120 slot labels and 21 intent types. The slots in the dataset are annotated based on the IOB (Inside, Outside, Beginning) Tags. The IOB tagging is similar to the part-of-speech tagging where it is used for tagging tokens in a chunking task. In this tagging format, the tag (B-) represents the beginning of the chunk, the tag (I-) represents the inside of the chunk, and the tag (O) represents no chunk.

Translation of the ATIS Dataset

In order to translate the ATIS dataset, the Microsoft Translator API was used. However, a specific approach had to be used for translation. As was mentioned in the previous section, each sample in the dataset includes three values which

Figure 5.9: ATIS Data Example

Sentence	show	me	evening	flights	to	baltimore
Slots	O	O	B-period_of_day	O	O	B-toloc.city_name
Intent	atis_flight					

are the actual utterance, an annotated slot for every word in the utterance, and the intent for the whole utterance. A specific approach had to be implemented in order to link every word in English with its translated word in Arabic and then to assign that word in Arabic the same slot of the English word. To do that, the word alignment option available in the Microsoft Translator API was used. This word alignment option returns for every translated sentence, the link for every translated word with the original word.

Figure 5.10 shows an example of an utterance in the dataset which is translated to Arabic. In Figure 5.11, the alignment returned with the translated sentence is shown. In the alignment, each chunk represents an alignment. For example, 0:3-0:1 indicates that the letters in index 0 to 3 in English are linked with the letters in index 0 to 1 in Arabic. This indicates that the word “What” in English is the translation of ”ما” in Arabic. In this way, all the words in English are mapped to their translations in Arabic as shown in the last row in Figure 5.11.

Figure 5.10: ATIS Sentence Translation Example

Original	what is the meaning of meal code
Translation	ما هو معنى رمز وجبة

Figure 5.11: ATIS Word Alignment Example

Alignment	0:3-0:1 5:6-3:4 12:18-6:9 23:26-15:18 28:31-11:13				
0:3-0:1	5:6-3:4	12:18-6:9	23:26-15:18	28:31-11:13	
what-ما	is-هو	meaning-معنى	meal-وجبة	code-رمز	

After the words have been aligned, the slots have to be mapped. In that way, the assigned slot of the word in English will be assigned to its translated word

in Arabic. Figure 5.12 shows the slot alignment process done to assign the slot for the Arabic words. This slot alignment is done based on the alignment found before as shown previously in Figure 5.11.

Figure 5.12: ATIS Slot Alignment Example

		وجبة	رمز	معنى	هو	ما
		I-meal_code	B-meal_code	O	O	O
what	is	the	meaning	of	meal	code
O	O	O	O	O	B-meal_code	I-meal_code

Using this approach, all the utterances in the ATIS dataset were translated. However, there were several errors faced when using the word alignment option. The first problem is that some words in Arabic had no alignment with any English word in the returned word alignments. In this case, the slot of that word was assigned to be “O” and then these results were checked again. Another problem faced is when one word in English is translated to two words in Arabic. In that case, the two Arabic words will be assigned the same slot. This is shown in Figure 5.13 where the word “early” is translated to the two words ”وقت مبكر” in Arabic and they are both given the slots “B-arrive_time.period_mod”. The problem here is that these two words are considered one chunk and the second word in the chunk should start with “I”.

Figure 5.13: ATIS Translation Problem - 1

الصباح	من	مبكر	وقت	في	تصل
B-arrive_time.period_of_day	O	B-arrive_time.period_mod	B-arrive_time.period_mod	O	O
arrive	early	in	the	morning	
O	B-arrive_time.period_mod	O	O	B-arrive_time.period_of_day	

Another problem faced is having chunks which do not start with “B-”. This also occurs when two words are aligned with one word in English and that word

is the second word in a chunk and hence starts with “I-”. This is shown in Figure 5.14 where both words in Arabic are aligned with the word “Petersburg” and hence both are given the slot “I-fromloc.city_name”.

Figure 5.14: ATIS Translation Problem - 2

بطرسبرغ	سانت	من	الجوية	الرحلات	لي	تبين
I- fromloc.city_ name	I- fromloc.city_ _name	O	O	O	O	O
show	me	the	flights	from	st.	petersburg
O	O	O	O	O	B- fromloc.city_na me	I- fromloc.city_na me

There are several other issues in alignments that can result in wrong slot assignment. For this, the slots were checked for these issues to make sure all chunks start with a “B-” and all other words in the same chunk start with an “I-”.

Apart from these problems that could be solved automatically in the code, there were other problems in the alignment that required a manual QC of the results to be done. One of these problems is having wrong alignment results which means wrong words are mapped to each other. This can be explained using Figure 5.15 which shows that the word “الثامن” was mapped to the word “February” which is a wrong mapping and hence the assigned slot is wrong.

Figure 5.15: ATIS Translation Problem - 3

فبراير	والعشرين	الثامن	في
B- arrive_date.month_na me	B- arrive_date.day_numb er	B- arrive_date.month_na me	O
on	february	twenty	eighth
O	B- arrive_date.month_na me	B- arrive_date.day_numb er	I- arrive_date.day_numb er

Apart from the mentioned problems in alignment results, there were problems in the translation of the sentences where some words were translated in a wrong way. Consequently, a manual QC was done on all the translations and all the annotated slots to make sure that the data is translated correctly, and the slots are

assigned correctly. Following this, a clean Arabic ATIS dataset of 5871 utterances is available which can be used to train the NLU component.

5.2.3 Evaluation and Results

Training Details

For training, the AraBERT Base model [8] was used which has 798 hidden states, 12 heads, and 12 layers. The model was pretrained using 200M sentences collected from the OSCAR corpus, the Arabic Wikipedia dump, the 1.5B words Arabic Corpus, the OSIAN Corpus, and the Assafir news articles. For training, 8 epochs were used, and Adam was used for optimization. The batch size used for training and evaluation is 16. Hyperparameter tuning was done on the learning rate, the seed, and the number of warmup steps. The optimal parameters found for each of the different approaches are shown in Table 5.1.

Parameter	Intent Classification	Slot Filling	Joint Intent Classification and Slot Filling
Learning Rate	2e-05	5e-05	5e-05
Seed	42	42	42
Warmup Steps	0	41	0

Table 5.1: NLU Training Parameters

Error Measures Used

The error measures used to evaluate the trained model are the F1, precision, recall, and accuracy for both the intents and the slots predicted. An additional error measure used to evaluate the Joint model is the semantic frame accuracy. The semantic frame accuracy error measure is used to evaluate the performance of the joint model by evaluating the accuracy of having both the intent and the slots predicted correctly in one sentence. The formulas used for each error measure are the following:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad (5.4)$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (5.5)$$

$$Accuracy = \frac{TruePositive + TrueNegative}{AllSamples} \quad (5.6)$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (5.7)$$

$$SemanticFrameAccuracy = Accuracy(correctIntent + correctallslots) \quad (5.8)$$

Results – AraBERT for Intent Classification

The training results on each epoch for fine-tuning AraBERT for Intent classification are shown in Figure 5.16. The least errors were reached at epoch 8 where a training loss of 0.0264, a validation loss of 0.197928, an F1 of 83.26%, a precision of 83.31%, a recall of 86.1%, and an accuracy of 96.8% were achieved. Results on the test dataset are shown in Figure 5.17. The test results achieved are an F1 score of 88.947%, a precision of 92.33%, a recall of 89.795%, and an accuracy of 97.2%.

Figure 5.16: AraBERT for Intent Classification - Train Results

Epoch	Training Loss	Validation Loss	F1	Precision	Recall	Accuracy
1	No log	0.310757	0.462724	0.475161	0.464243	0.932000
2	No log	0.200623	0.720932	0.729824	0.719652	0.962000
3	No log	0.190137	0.851363	0.860151	0.860760	0.966000
4	0.279100	0.194558	0.842653	0.834295	0.876503	0.968000
5	0.279100	0.192020	0.832550	0.833115	0.860947	0.968000
6	0.279100	0.195810	0.828559	0.832929	0.854280	0.966000
7	0.279100	0.198383	0.832550	0.833115	0.860947	0.968000
8	0.026400	0.197928	0.832550	0.833115	0.860947	0.968000

Figure 5.17: AraBERT for Intent Classification - Test Results

Error Measure	Score (%)
F1	88.947
Precision	92.330
Recall	89.795
Accuracy	97.200

Results – AraBERT for Slot Filling

The training results on each epoch for fine-tuning AraBERT for slot filling are shown in Figure 5.18. The least errors were reached at epoch 8 where a training

loss of 0.0345, a validation loss of 0.110035, an F1 of 97.63%, a precision of 94.91%, a recall of 95.02%, and an accuracy of 94.96% were achieved. Results on the test dataset are shown in Figure 5.19. The test results achieved are an F1 score of 97.040%, a precision of 93.471%, a recall of 93.768%, and an accuracy of 93.619%.

Figure 5.18: AraBERT for Slot Filling - Train Results

Epoch	Training Loss	Validation Loss	F1	Precision	Recall	Accuracy
1	No log	0.215457	0.947930	0.884930	0.887522	0.886224
2	No log	0.130123	0.968541	0.933918	0.935559	0.934738
3	No log	0.106840	0.973965	0.938882	0.944933	0.941898
4	0.341000	0.100229	0.975954	0.943572	0.950205	0.946877
5	0.341000	0.106430	0.975773	0.947862	0.947862	0.947862
6	0.341000	0.107096	0.975592	0.946262	0.949033	0.947646
7	0.341000	0.109431	0.976135	0.949678	0.950791	0.950234
8	0.034500	0.110035	0.976315	0.949093	0.950205	0.949649

Figure 5.19: AraBERT for Slot Filling - Test Results

Error Measure	Score (%)
F1	97.040
Precision	93.471
Recall	93.768
Accuracy	93.619

Results – Joint AraBERT for Intent Classification and Slot Filling

Joint AraBERT – No CRF The training results on each epoch for fine-tuning AraBERT for joint intent classification and slot filling are shown in Figure 5.20. The least errors were reached at epoch 8 where a training loss of 0.426, an intent accuracy of 97.2%, an intent F1 of 86.9%, an intent precision of 86.8%, an intent recall of 88.3%, a slot accuracy of 97.9%, a slot F1 of 95.5%, a slot precision

of 95.4%, a slot recall of 95.5%, and a semantic frame accuracy of 85% were achieved. Results on the test dataset are shown in Figure 5.21. The test results achieved are an intent accuracy of 97.648%, an intent F1 of 86.186%, an intent precision of 88.114%, an intent recall of 86.953%, a slot accuracy of 97.107%, a slot F1 of 93.699%, a slot precision of 93.666%, a slot recall of 93.732%, and a semantic frame accuracy of 82.755%.

Figure 5.20: Joint AraBERT – No CRF - Train Results

Epoch	Loss	Intent Accuracy	Intent F1	Intent Precision	Intent Recall	Slot Accuracy	Slot F1	Slot Precision	Slot Recall	Semantic Frame Accuracy
1	0.418	0.970	0.921	0.919	0.946	0.971	0.929	0.919	0.938	0.792
2	0.405	0.972	0.867	0.871	0.877	0.975	0.946	0.944	0.947	0.822
3	0.462	0.966	0.868	0.881	0.870	0.976	0.946	0.943	0.948	0.840
4	0.396	0.972	0.871	0.871	0.883	0.978	0.952	0.951	0.953	0.844
5	0.418	0.972	0.869	0.868	0.883	0.978	0.953	0.953	0.954	0.844
6	0.429	0.970	0.867	0.871	0.877	0.979	0.956	0.956	0.956	0.854
7	0.427	0.972	0.869	0.868	0.883	0.979	0.953	0.952	0.954	0.844
8	0.426	0.972	0.869	0.868	0.883	0.979	0.955	0.954	0.955	0.850

Figure 5.21: Joint AraBERT – No CRF - Test Results

Error Measure	Score (%)
Intent Accuracy	97.648
Intent F1	86.186
Intent Precision	88.114
Intent Recall	86.953
Slot Accuracy	97.107
Slot F1	93.699
Slot Precision	93.666
Slot Recall	93.732
Semantic Frame Accuracy	82.755

Joint AraBERT – With CRF The training results on each epoch for fine-tuning AraBERT for joint intent classification and slot filling with the addition of the CRF layer are shown in Figure 5.22. The least errors were reached at epoch 8

where a training loss of 1.850, an intent accuracy of 97.2%, an intent F1 of 85%, an intent precision of 83.6%, an intent recall of 88.7%, a slot accuracy of 95%, a slot F1 of 95.3%, a slot precision of 95.1%, a slot recall of 97.7%, and a semantic frame accuracy of 84.4% were achieved. Results on the test dataset are shown in Figure 5.23. The test results achieved are an intent accuracy of 96.865%, an intent F1 of 86.205%, an intent precision of 91.730%, an intent recall of 86.248%, a slot accuracy of 97.018%, a slot F1 of 93.728%, a slot precision of 93.794%, a slot recall of 93.662%, and a semantic frame accuracy of 81.859%.

Figure 5.22: Joint AraBERT – With CRF - Train Results

Epoch	Loss	Intent Accuracy	Intent F1	Intent Precision	Intent Recall	Slot Accuracy	Slot F1	Slot Precision	Slot Recall	Semantic Frame Accuracy
1	1.805	0.958	0.728	0.726	0.740	0.971	0.936	0.930	0.941	0.796
2	1.738	0.968	0.767	0.779	0.794	0.972	0.939	0.937	0.941	0.810
3	1.823	0.966	0.794	0.793	0.810	0.972	0.943	0.943	0.943	0.822
4	1.696	0.970	0.802	0.796	0.820	0.975	0.947	0.945	0.948	0.824
5	1.640	0.972	0.812	0.817	0.817	0.977	0.952	0.951	0.953	0.846
6	1.780	0.972	0.811	0.811	0.821	0.977	0.952	0.950	0.954	0.850
7	1.843	0.970	0.867	0.868	0.880	0.977	0.951	0.949	0.953	0.840
8	1.850	0.972	0.850	0.836	0.887	0.950	0.953	0.951	0.977	0.844

Figure 5.23: Joint AraBERT – With CRF - Test Results

Error Measure	Score (%)
Intent Accuracy	96.865
Intent F1	86.205
Intent Precision	91.730
Intent Recall	86.248
Slot Accuracy	97.018
Slot F1	93.728
Slot Precision	93.794
Slot Recall	93.662
Semantic Frame Accuracy	81.859

Evaluation and Analysis of Results

Following the implementation of the two approaches of training separate models for intent classification and slot filling and the training of a joint model for both tasks, the results have to be compared to conclude on the best approach. Figure 5.24 shows the results of both approaches for comparison.

Figure 5.24: NLU Results Comparison

Model	Intent				Slot				Semantic Frame Accuracy
	Accuracy	F1	Precision	Recall	Accuracy	F1	Precision	Recall	
AraBERT – Intent Classification	97.2%	88.947%	92.330%	89.795%	-	-	-	-	-
AraBERT – Slot Filling	-	-	-	-	97.040%	93.619%	93.471%	93.619%	-
Joint AraBERT	97.648%	86.186%	88.114%	86.953%	97.107%	93.699%	93.666%	93.732%	82.755%
Joint AraBERT + CRF	96.865%	86.205%	91.730%	86.248%	97.018%	93.728%	93.794%	93.662%	81.859%

Comparing the results with respect to the intent classification task, it can be seen that fine-tuning AraBERT on the intent classification task only results in better performance in terms of F1, accuracy, and recall compared with the joint approach. The model for intent classification only was able to achieve 2.76% and 2.74% higher F1, 4.22% and 0.6% higher precision, and 2.84% and 3.55% higher recall compared with the joint AraBERT without CRF and with CRF approaches, respectively. However, the joint approach is better in terms of accuracy. Figure 5.25 shows examples of the predicted intents using each approach using test samples. Each example is analyzed to better understand the performance of each model. Analyzing the results achieved using the first example where all the models predicted wrong results, the reason was that for the intent “atis.aircraft”, all the sentences in the training dataset don’t contain “عدد”, they just contain “أنواع” or “نوع”. The word “عدد” is more linked to the intent “atis_capacity” which is for questions about the capacity of airplanes and the intent “atis_quantity” which is for questions about the number of flights. For the second example, the reason for the wrong prediction is that the annotated actual intent is wrong, and it should be “atis_quantity” as predicted. For the third example, only the joint AraBERT without CRF predicted the intent correctly.

After analyzing the results, it was observed that the word “مدى”, was not used in the training dataset which made it difficult for the models to predict the intent, but the joint model was able to correctly predict it. For the fourth example, both joint models with and without CRF were able to correctly predict the intent but the model fine-tuned on intent classification only was not able to correctly predict it. The reason behind that might be a confusion in prediction as the sentence contains some abbreviation but the question is not about it.

Figure 5.25: Intent Classification Results

Model	في مطار شارلوت كم عدد أنواع مختلفة من الطائرات هناك على الطيران الأميركي	كم عدد الرحلات الجوية التي خطوط الاسكا لديها إلى بربانك	إلى أي مدى هو نيويورك لا غورديون من وسط المدينة	ما هو القدرة على الجلوس من 9
Actual	atis_aircraft	atis_flight	atis_distance	atis_capacity
No Joint	atis_capacity	atis_quantity	atis_city	atis_abbreviation
Joint – No CRF	atis_quantity	atis_quantity	atis_distance	atis_capacity
Joint – With CRF	atis_quantity	atis_quantity	atis_city	atis_capacity

Comparing the results with respect to the slot filling task, it can be seen that using the joint approach results in better performance in terms of all error measures compared with fine-tuning AraBERT on the slot filling task only. Results of both joint approaches with and without CRF are very close. In terms of F1 and accuracy, the joint model with CRF achieves 0.029% higher F1, 0.128% higher precision, and 0.07% lower recall compared with the joint approach without CRF. Figure 5.26 shows an example of the slots predicted for one of the sentences in the test set. All the models predicted the wrong slot for the word “أو”. After analyzing the results, it was noticed that the annotated actual slot for that word in the dataset is wrong and should be 'B-or' as predicted by all the models. Another example is shown in Figure 5.27 where both joint models with and without CRF were able to correctly predict the slot for the word “لا” which is the first word in the airport name “لا غورديون”. However, the model fine-tuned on the slot filling task only had a wrong prediction for the slot of that word. A reason for that wrong prediction might be the confusion between the slots “airport_name” and “fromloc.airport_name” which both refer to airport names. However, the joint model was able to distinguish between them and the reason behind that is the strong link between the intents and the slots which is learned in the joint model that helped predict the slots more accurately. Another example is shown in Figure 5.28 where only the joint model without CRF was able to correctly predict the slot for the word “الخطوط”. After analyzing the data, the reason

behind the difficulty in the prediction for other models might be the confusion in the annotation as the word “الخطوط” is sometimes annotated as the beginning of the airline_name chunk and sometimes not. However, the joint model was able to correctly predict the slot and the main reason is the addition of both learnings of intents and slots in the prediction of the slots.

Figure 5.26: Slot Filling Results - 1

Sentence	أنا بحاجة إلى رحلة من تورونتو إلى نيويورك في اتجاه واحد ترك مساء الأربعاء أو صباح الخميس
Actual	'O', 'O', 'O', 'O', 'O', 'B-fromloc.city_name', 'O', 'B-toloc.city_name', 'B-round_trip', 'I-round_trip', 'I-round_trip', 'O', 'B-depart_time.period_of_day', 'B-depart_date.day_name', 'O', 'B-depart_time.period_of_day', 'B-depart_date.day_name'
No Joint	'O', 'O', 'O', 'O', 'O', 'B-fromloc.city_name', 'O', 'B-toloc.city_name', 'B-round_trip', 'I-round_trip', 'I-round_trip', 'O', 'B-depart_time.period_of_day', 'B-depart_date.day_name', 'B-or', 'B-depart_time.period_of_day', 'B-depart_date.day_name'
Joint – No CRF	'O', 'O', 'O', 'O', 'O', 'B-fromloc.city_name', 'O', 'B-toloc.city_name', 'B-round_trip', 'I-round_trip', 'I-round_trip', 'O', 'B-depart_time.period_of_day', 'B-depart_date.day_name', 'B-or', 'B-depart_time.period_of_day', 'B-depart_date.day_name'
Joint – With CRF	'O', 'O', 'O', 'O', 'O', 'B-fromloc.city_name', 'O', 'B-toloc.city_name', 'B-round_trip', 'I-round_trip', 'I-round_trip', 'O', 'B-depart_time.period_of_day', 'B-depart_date.day_name', 'B-or', 'B-depart_time.period_of_day', 'B-depart_date.day_name'

Figure 5.27: Slot Filling Results - 2

Sentence	كم هو خدمة الليموزين في لا غوردون
Actual	'O', 'O', 'O', 'B-transport_type', 'O', 'B-airport_name', 'I-airport_name'
No Joint	'O', 'O', 'O', 'B-transport_type', 'O', 'B-fromloc.airport_name', 'I-airport_name'
Joint – No CRF	'O', 'O', 'O', 'B-transport_type', 'O', 'B-airport_name', 'I-airport_name'
Joint – With CRF	'O', 'O', 'O', 'B-transport_type', 'O', 'B-airport_name', 'I-airport_name'

Comparing the results in terms of the semantic frame accuracy which can only be evaluated for the joint approaches, the joint model without CRF was able to

Figure 5.28: Slot Filling Results - 3

Sentence	يرجى قائمة الرحلات من سينسيناتي إلى بربانك على الخطوط الطيران الأميركي
Actual	'O', 'O', 'O', 'O', 'B-fromloc.city_name', 'O', 'B-toloc.city_name', 'O', 'O', 'B-airline_name', 'I-airline_name'
No Joint	'O', 'O', 'O', 'O', 'B-fromloc.city_name', 'O', 'B-toloc.city_name', 'O', 'B-airline_name', 'B-airline_name', 'I-airline_name'
Joint – No CRF	'O', 'O', 'O', 'O', 'B-fromloc.city_name', 'O', 'B-toloc.city_name', 'O', 'O', 'B-airline_name', 'I-airline_name'
Joint – With CRF	'O', 'O', 'O', 'O', 'B-fromloc.city_name', 'O', 'B-toloc.city_name', 'O', 'B-airline_name', 'I-airline_name', 'I-airline_name'

achieve 0.896% better accuracy compared with the joint model with CRF. Finally, considering accuracy as the evaluation criteria for the intent classification task and the F1 as evaluation criteria for the slot filling task, the joint AraBERT without CRF is the best for intent classification and the joint AraBERT with CRF is the best for slot filling. However, considering the very slight difference in the F1 of the approaches with and without CRF (0.029%) and considering the higher semantic frame accuracy of the joint AraBERT without CRF, this model will be chosen to be the best one.

Following the comparison of the different approaches for both tasks, it is essential to compare the results achieved in Arabic to the ones achieved in English using the same joint approach and using the same ATIS dataset in English. Figure 5.29 shows the results achieved using our approach compared with the ones achieved in [21] using BERT and using the English ATIS dataset. Comparing

Figure 5.29: NLU Results Compared with SOTA

Dataset	Model	Intent Accuracy (%)	Slot F1 (%)	Sentence-level semantic frame accuracy (%)
Arabic	Joint AraBERT	97.648	93.699	82.755
	Joint AraBERT + CRF	96.865	93.728	81.859
English	Joint BERT	97.5	96.1	88.2
	Joint BERT + CRF	97.9	96	88.6

the results, it can be seen that our approach achieved comparable results with the approach using BERT on the English ATIS dataset. This assures that the translated ATIS dataset serves as a good dataset for NLU training and that the

approach of using joint AraBERT achieves comparable results with the English.

Chapter 6

Future Work

In this work, two approaches were suggested which are the end-to-end approach and the pipeline approach. For the pipeline approach, the Natural Language Understanding (NLU) component have been developed using AraBERT. However, the remaining components of the pipeline approach still need to be implemented and this will be part of the future work. In the future work, both the Dialogue Manager and the Natural Language Understanding (NLG) components will be implemented.

For implementing the Dialogue Manager component, the suggested methodology is to use reinforcement learning to train this component in an end-to-end method. An approach similar to the one suggested in [70] could be implemented. In this approach, a rule-based agent is used as a warm-start for the system, and this is done using supervised learning. Then the Dialogue Manager component is trained in an end-to-end way using reinforcement learning where the Deep Q-network reinforcement learning method was used. Training in an end-to-end way can be done by adding the NLU and NLG components trained separately or without adding the NLU and NLG components and replacing them by an error model controller. The error model controller can simulate and add noises from the NLU and NLG components which exist naturally in any communication between the user and the system.

For implementing the NLG component, the suggested methodology is to use GPT2 as proposed in the work in [48]. In this work, GPT2 was pretrained on large amounts of data containing dialogue acts and system responses. Then the model was fine-tuned on data from specific domains to adapt the NLG to domain-specific labels.

For the implementation of the Dialogue Manager and NLG components, data is required which is different for each of these components. The translated MultiWOZ dataset done part of this work contains the required data needed to train both the Dialogue Manager and NLG components and hence it can be used.

An additional step after training all the separate components is to integrate them. This is a challenging part as the output of one component should be

compatible with the input of the next component in terms of slots and domains and hence all components have to be trained on data with the same ontology to be able to integrate them or an additional integration component can be added between each of the developed components to reformat the data as needed by the next component.

After implementing and integrating the three components in the pipeline approach, the results would be compared to those generated using the end-to-end approach to evaluate and conclude on the better approach to be used. As for now, the two approaches could not be compared as only one component of the pipeline approach was implemented. The two approaches will be compared as part of the future work when the other components of the pipeline approach are implemented and based on that the two approaches will be compared in terms of performance.

Chapter 7

Conclusion

In this work, we have proposed the development of an end-to-end task-oriented dialogue system in Arabic using the pretrained language generation model AraGPT2. A multi-domain human-to-human conversational dataset was created by translating the MultiWOZ to Arabic and it was used for fine-tuning AraGPT2. The developed model was able to achieve results comparable with the state-of-the-art results in English. The developed end-to-end task-oriented dialogue was evaluated on three different aspects which are the response generation, the policy optimization, and a whole end-to-end system. Results were analyzed and the chatbot was able to achieve accurate results in terms of translating the user utterance to a belief state, and then generating the system act and system response based on the resulting belief state and database results. This proves that the proposed approach achieves good results on the Arabic language and also that the created conversational dataset is of good quality and can be used to train task-oriented dialogue systems.

We also proposed the development of the NLU component, which is one of the components in the pipeline approach, using the pretrained language model AraBERT. A dataset for training the NLU component was created by translating the ATIS dataset to Arabic and it was used to fine-tune AraBERT on joint intent classification and slot filling. Results achieved were analyzed by comparing the results achieved using the joint approach and using separate models for each task and it was concluded that the joint results were better as the intents and slots are highly correlated and modeling them jointly helps in sharing the information between them. In addition to that, the achieved results prove the high quality of the created dataset which is the first Arabic dataset which can be used for training the NLU component.

As part of the future work, the dialogue manager and NLG components have to be developed and combined with the NLU component to implement the complete pipeline approach. Results achieved using the pipeline approach have to be compared with the ones achieved using the developed end-to-end approach to conclude on the better approach.

Bibliography

- [1] R. Khan and A. Das, “Introduction to chatbots,” in *Build better chatbots*, pp. 1–11, Springer, 2018.
- [2] K. Ramesh, S. Ravishankaran, A. Joshi, and K. Chandrasekaran, “A survey of design techniques for conversational agents,” in *International conference on information, communication and computing technology*, pp. 336–350, Springer, 2017.
- [3] Z. Zhang, R. Takanobu, Q. Zhu, M. Huang, and X. Zhu, “Recent advances and challenges in task-oriented dialog systems,” *Science China Technological Sciences*, pp. 1–17, 2020.
- [4] H. Chen, X. Liu, D. Yin, and J. Tang, “A survey on dialogue systems: Recent advances and new frontiers,” *Acm Sigkdd Explorations Newsletter*, vol. 19, no. 2, pp. 25–35, 2017.
- [5] D. A. Ali and N. Habash, “Botta: An arabic dialect chatbot,” in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pp. 208–212, 2016.
- [6] W. Antoun, F. Baly, and H. Hajj, “Aragpt2: pre-trained transformer for arabic language generation,” *arXiv preprint arXiv:2012.15520*, 2020.
- [7] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gašić, “Multiwoz—a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling,” *arXiv preprint arXiv:1810.00278*, 2018.
- [8] W. Antoun, F. Baly, and H. Hajj, “Arabert: Transformer-based model for arabic language understanding,” *arXiv preprint arXiv:2003.00104*, 2020.
- [9] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, “The atis spoken language systems pilot corpus,” in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.

- [10] J. Weizenbaum, “Eliza—a computer program for the study of natural language communication between man and machine,” *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.
- [11] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” *Advances in neural information processing systems*, vol. 28, pp. 649–657, 2015.
- [12] H. B. Hashemi, A. Asiaee, and R. Kraft, “Query intent detection using convolutional neural networks,” in *International Conference on Web Search and Data Mining, Workshop on Query Understanding*, 2016.
- [13] S. Ravuri and A. Stolcke, “Recurrent neural network and lstm models for lexical utterance classification,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [14] P. Liu, X. Qiu, and X. Huang, “Adversarial multi-task learning for text classification,” *arXiv preprint arXiv:1704.05742*, 2017.
- [15] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, “Recurrent neural networks for language understanding,” in *Interspeech*, pp. 2524–2528, 2013.
- [16] N. T. Vu, “Sequential convolutional neural networks for slot filling in spoken language understanding,” *arXiv preprint arXiv:1606.07783*, 2016.
- [17] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, “Spoken language understanding using long short-term memory neural networks,” in *2014 IEEE Spoken Language Technology Workshop (SLT)*, pp. 189–194, IEEE, 2014.
- [18] L. Zhao and Z. Feng, “Improving slot filling in spoken language understanding with joint pointer and attention,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 426–431, 2018.
- [19] D. Hakkani-Tür, G. Tür, A. Celikyilmaz, Y.-N. Chen, J. Gao, L. Deng, and Y.-Y. Wang, “Multi-domain joint semantic frame parsing using bi-directional rnn-lstm,” in *Interspeech*, pp. 715–719, 2016.
- [20] C.-W. Goo, G. Gao, Y.-K. Hsu, C.-L. Huo, T.-C. Chen, K.-W. Hsu, and Y.-N. Chen, “Slot-gated modeling for joint slot filling and intent prediction,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 753–757, 2018.
- [21] Q. Chen, Z. Zhuo, and W. Wang, “Bert for joint intent classification and slot filling,” *arXiv preprint arXiv:1902.10909*, 2019.

- [22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [23] N. Mrkšić, D. O. Séaghdha, B. Thomson, M. Gašić, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young, “Multi-domain dialog state tracking using recurrent neural networks,” *arXiv preprint arXiv:1506.07190*, 2015.
- [24] P.-H. Su, M. Gasic, N. Mrksic, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.-H. Wen, and S. Young, “Continuously learning neural dialogue management,” *arXiv preprint arXiv:1606.02689*, 2016.
- [25] H. Lee, J. Lee, and T.-Y. Kim, “Sumbt: Slot-utterance matching for universal and scalable belief tracking,” *arXiv preprint arXiv:1907.07421*, 2019.
- [26] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai, “A form-based dialogue manager for spoken language applications,” in *Proceeding of Fourth International Conference on Spoken Language Processing. IC-SLP’96*, vol. 2, pp. 701–704, IEEE, 1996.
- [27] Z. Wang and O. Lemon, “A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information,” in *Proceedings of the SIGDIAL 2013 Conference*, pp. 423–432, 2013.
- [28] J. D. Williams, “Web-style ranking and slu combination for dialog state tracking,” in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pp. 282–291, 2014.
- [29] V. Zhong, C. Xiong, and R. Socher, “Global-locally self-attentive encoder for dialogue state tracking,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1458–1467, 2018.
- [30] M. Henderson, B. Thomson, and S. Young, “Deep neural network approach for the dialog state tracking challenge,” in *Proceedings of the SIGDIAL 2013 Conference*, pp. 467–471, 2013.
- [31] N. Mrkšić, D. O. Séaghdha, T.-H. Wen, B. Thomson, and S. Young, “Neural belief tracker: Data-driven dialogue state tracking,” *arXiv preprint arXiv:1606.03777*, 2016.
- [32] N. Mrkšić and I. Vulić, “Fully statistical neural belief tracking,” *arXiv preprint arXiv:1805.11350*, 2018.
- [33] L. Ren, K. Xie, L. Chen, and K. Yu, “Towards universal dialogue state tracking,” *arXiv preprint arXiv:1810.09587*, 2018.

- [34] J. Hu, Y. Yang, C. Chen, L. He, and Z. Yu, “Sas: Dialogue state tracking via slot attention and slot information sharing,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6366–6375, 2020.
- [35] F. Ye, J. Manotumruksa, Q. Zhang, S. Li, and E. Yilmaz, “Slot self-attentive dialogue state tracking,” in *Proceedings of the Web Conference 2021*, pp. 1598–1608, 2021.
- [36] L. Zhou, K. Small, O. Rokhlenko, and C. Elkan, “End-to-end offline goal-oriented dialog policy learning via policy gradient,” *arXiv preprint arXiv:1712.02838*, 2017.
- [37] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, JMLR Workshop and Conference Proceedings, 2011.
- [38] Z. Lipton, X. Li, J. Gao, L. Li, F. Ahmed, and L. Deng, “Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.
- [39] H. Cuayáhuitl, S. Keizer, and O. Lemon, “Strategic dialogue management via deep reinforcement learning,” *arXiv preprint arXiv:1511.08099*, 2015.
- [40] X. Li, Z. C. Lipton, B. Dhingra, L. Li, J. Gao, and Y.-N. Chen, “A user simulator for task-completion dialogues,” *arXiv preprint arXiv:1612.05688*, 2016.
- [41] Y. Wu, X. Li, J. Liu, J. Gao, and Y. Yang, “Switch-based active deep dyna-q: Efficient adaptive planning for task-completion dialogue policy learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7289–7296, 2019.
- [42] K. Lu, Y. Cao, X. Chen, and S. Zhang, “Efficient dialog policy learning with hindsight, user modeling, and adaptation,” *IEEE Transactions on Cognitive and Developmental Systems*, 2021.
- [43] T.-H. Wen, M. Gasic, D. Kim, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young, “Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking,” *arXiv preprint arXiv:1508.01755*, 2015.
- [44] T.-H. Wen, M. Gasic, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young, “Semantically conditioned lstm-based natural language generation for spoken dialogue systems,” *arXiv preprint arXiv:1508.01745*, 2015.

- [45] T.-H. Wen, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, P.-H. Su, D. Vandyke, and S. Young, “Multi-domain neural network language generation for spoken dialogue systems,” *arXiv preprint arXiv:1603.01232*, 2016.
- [46] H. Zhou, M. Huang, and X. Zhu, “Context-aware natural language generation for spoken dialogue systems,” in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 2032–2041, 2016.
- [47] O. Dušek and F. Jurčiček, “Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings,” *arXiv preprint arXiv:1606.05491*, 2016.
- [48] B. Peng, C. Zhu, C. Li, X. Li, J. Li, M. Zeng, and J. Gao, “Few-shot natural language generation for task-oriented dialog,” *arXiv preprint arXiv:2002.12328*, 2020.
- [49] W. Lei, X. Jin, M.-Y. Kan, Z. Ren, X. He, and D. Yin, “Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1437–1447, 2018.
- [50] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [51] B. Peng, C. Li, J. Li, S. Shayandeh, L. Liden, and J. Gao, “Soloist: Few-shot task-oriented dialog with a single pretrained auto-regressive model,” *arXiv preprint arXiv:2005.05298*, 2020.
- [52] D. Ham, J.-G. Lee, Y. Jang, and K.-E. Kim, “End-to-end neural pipeline for goal-oriented dialogue systems using gpt-2,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 583–592, 2020.
- [53] Y. Yang, Y. Li, and X. Quan, “Ubar: Towards fully end-to-end task-oriented dialog systems with gpt-2,” *arXiv preprint arXiv:2012.03539*, 2020.
- [54] A. Shawar and E. Atwell, “An arabic chatbot giving answers from the qur’an,” in *Proceedings of TALN04: XI Conference sur le Traitement Automatique des Langues Naturelles*, vol. 2, pp. 197–202, ATALA, 2004.
- [55] B. A. Shawar and E. Atwell, *A comparison between Alice and Elizabeth chatbot systems*. University of Leeds, School of Computing research report 2002.19, 2002.

- [56] B. A. Shawar and E. Atwell, “Accessing an information system by chatting,” in *International Conference on Application of Natural Language to Information Systems*, pp. 407–412, Springer, 2004.
- [57] B. A. Shawar, “A chatbot as a natural web interface to arabic web qa,” *International Journal of Emerging Technologies in Learning (iJET)*, vol. 6, no. 1, pp. 37–43, 2011.
- [58] B. A. Shawar and E. Atwell, “Arabic question-answering via instance based learning from an faq corpus,” in *Proceedings of the CL 2009 International Conference on Corpus Linguistics. UCREL*, vol. 386, pp. 1–12, 2009.
- [59] O. G. Alobaidi, K. A. Crockett, J. D. O’Shea, and T. M. Jarad, “Abdullah: An intelligent arabic conversational tutoring system for modern islamic education,” in *Proceedings of the World Congress on Engineering*, vol. 2, 2013.
- [60] M. Hijjawi, Z. Bandar, K. Crockett, and D. Mclean, “Arabchat: An arabic conversational agent,” in *2014 6th International Conference on Computer Science and Information Technology (CSIT)*, pp. 227–237, IEEE, 2014.
- [61] M. Hijjawi, H. Qattous, and O. Alsheiksalem, “Mobile arabchat: An arabic mobile-based conversational agent,” *Int. J. Adv. Comput. Sci. Appl. IJACSA*, vol. 6, no. 10, 2015.
- [62] M. Hijjawi, Z. Bandar, and K. Crockett, “The enhanced arabchat: An arabic conversational agent,” *International Journal of Advanced Computer Science and Applications*, vol. 7, 2016.
- [63] S. S. Aljameel, J. D. O’Shea, K. A. Crockett, A. Latham, and M. Kaleem, “Development of an arabic conversational intelligent tutoring system for education of children with asd,” in *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pp. 24–29, IEEE, 2017.
- [64] A. Moubaidin, O. Shalbak, B. Hammo, and N. Obeid, “Arabic dialogue system for hotel reservation based on natural language processing techniques,” *Computación y Sistemas*, vol. 19, no. 1, pp. 119–134, 2015.
- [65] A. M. Bashir, A. Hassan, B. Rosman, D. Duma, and M. Ahmed, “Implementation of a neural natural language understanding component for arabic dialogue systems,” *Procedia computer science*, vol. 142, pp. 222–229, 2018.
- [66] A. Joukhadar, H. Saghergy, L. Kweider, and N. Ghneim, “Arabic dialogue act recognition for textual chatbot systems,” in *Proceedings of The First International Workshop on NLP Solutions for Under Resourced Languages (NSURL 2019) co-located with ICNLSP 2019-Short Papers*, pp. 43–49, 2019.

- [67] A. Elmadany, S. Abdou, and M. Gheith, “Improving dialogue act classification for spontaneous arabic speech and instant messages at utterance level,” *arXiv preprint arXiv:1806.00522*, 2018.
- [68] A.-H. Al-Ajmi and N. Al-Twairesh, “Building an arabic flight booking dialogue system using a hybrid rule-based and data driven approach,” *IEEE Access*, vol. 9, pp. 7043–7053, 2021.
- [69] C. Parsing, “Speech and language processing,” 2009.
- [70] X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz, “End-to-end task-completion neural dialogue systems,” *arXiv preprint arXiv:1703.01008*, 2017.
- [71] S. Young, “Cued standard dialogue acts,” *Report, Cambridge University Engineering Department, 14th October*, vol. 2007, 2007.

