A Spoken Goal-Oriented Dialogue System for Service Robots

João Nuno Marcelino Barroca
joao.barroca@tecnico.ulisboa.pt

Universidade de Lisboa, Portugal

December 2019

Abstract

In recent years, researchers have been trying to develop conversational systems capable of understanding and speaking in natural language, so that humans can interact with them easily and more naturally. Human-Computer dialogue systems, particularly goal-based dialogue systems, have been the most important component in conversational systems, since they allow users to speak naturally in order to accomplish tasks more efficiently. However, when deployed to production, a spoken dialogue system may encounter a variety of difficulties, such as a large variation in the users of the system and high noise environments. Thus, the dialogue system must be able to use statistical frameworks to handle the uncertainty in both the speech recognition and language understanding. In this work, we develop a spoken goal-oriented dialogue system capable of dealing with such uncertainties, by maintaining uncertainty about everything the user has said, and exploit the sequential nature of dialogue to disambiguate in the presence of errors. In particular, we design three main components: Natural Language Understanding (NLU), Dialogue State Tracking (DST) and Dialogue Management (DM). Furthermore, we implement the developed dialogue system in a service robot and test the complete system on a real world environment.

Keywords: Spoken Dialogue Systems, Natural Language Understanding, Dialogue State Tracking, Dialogue Management, Human-Robot Interaction

1. Introduction

Robotics hold tremendous potential to benefit humans in all aspect of life. These benefits have been widely demonstrated in industrial environments, where automation and robotics have boosted the efficiency and productivity of industries, while reducing the costs in labor and production. Nowadays, due to technological advances, particularly in the field of Artificial Intelligence, robots are starting to be deployed and integrated into our daily environment. Since, one of their main limitations is the inability to communicate with humans in a natural manner, the integration of robots into our daily lives has triggered the need to enhance Human-Robot Interaction (HRI) with speech and natural language.

In recent years, researchers have been trying to develop systems capable of understanding and speaking in natural language, so that humans can interact with them easily and more naturally. Those systems do not necessary have to be integrated in physical robots. In fact, conversational systems such as virtual personal assistants are increasingly becoming a part of daily life, with examples including Apples Siri, Google Now, Nuance Dragon Go, Xbox and Cortana from Microsoft, and numerous start-ups [16]. Spoken goal-based dialogue systems, have been the most important component in conversational systems, since they allow users to speak naturally in order to accomplish tasks more efficiently. A spoken dialogue system can be defined as a computer system able to interact with humans on a turn-by-turn basic, and in which spoken natural language interface plays an important part in the communication [1]. They are widely demanded in network information services, service, domestic and social robots, and so on.

When exposed to the public, a spoken dialogue system may encounter a variety of difficulties, such as a large variation in the users of the system and high noise environments. Therefore, the dialogue system must be able to use statistical frameworks to handle the uncertainty in both speech recognition and language understanding. Statistical dialogue systems are able to maintain uncertainty about everything the user has said, and exploit the sequential nature of dialogue to disambiguate in the presence of errors [6].

This work aims at developing a spoken goal-oriented dialogue system capable of dealing with uncertainty in speech recognition, to be further implemented in a service robot. To achieve this, we designed three main components of a typical dialogue system pipeline, namely the Natural Language Understanding (NLU), the Dialogue State Tracker (DST) and the Dialogue Management (DM).

2. Previous Work

Traditionally, dialogue systems can be split into two classes: the goal-driven or non-goal-driven dialogue systems. The former represent the interaction between a human and a computer in a task-oriented context, while the latter, usually called chatbots, are related to chat communication.

First research in task-oriented dialogue systems began in the early 1990s, when MIT (Massachusetts Institute of Technology) developed an automatic flight booking system [12]. Similar dialogue systems were developed in the following decade: HMIHY (How May I help You?), a spoken dialogue system based on call routing [5], and JUPITER, a conversational interface that allows users to obtain worldwide weather information over the telephone using spoken dialogue [18]. Non-goal-driven dialogue systems usually respond to user utterances without any specific goal. ELIZA [15] might be the first chatbot, and one of the first programs capable of attempting the Turing test.

The majority of modern task-based dialogue systems make use of the frame-based architecture, first introduced in the GUS system for travel planing [2]. In a frame-based architecture, the system uses a domain ontology, a knowledge structure representing the kinds of information the system can extract from user utterances [7]. The ontology not only defines one or more frames, each a collection of slots, but also the possible values each slot can take. However, frame-based dialogue systems are not capable of asking questions, giving orders, or making informational statements [7]. The dialogue-state architecture, like the GUS systems, is based on filling in the slots of frames. Nevertheless, the dialogue-state architecture has a different way of deciding what to say next than the GUS systems. Simple frame-based systems often just continuously ask questions corresponding to unfilled slots, while the dialogue-state based systems use a dialogue policy to decide what to say.

The typical dialogue-state architecture consists of the following components, arranged in a pipeline: Automatic Speech Recognizer (ASR), Natural Language Understanding (NLU), Dialogue State Tracking (DST), Dialogue Management (DM), Natural Language Generation (NLG), and Text-to-Speech (TTS). First, the computer needs to convert speech into text and then extract task related information from user utterances, which is done by the ASR and NLU, respectively. Second, the computer must be able to control the process of a dialogue. Initially, the DST estimates the state of the conversation, and then the DM generates actions based on the dialogue state. Finally, the computer needs to convert the system output actions to natural language sentences, and then to speech, which is the objective of NLG and TTS. This architecture is represented in Figure 1.

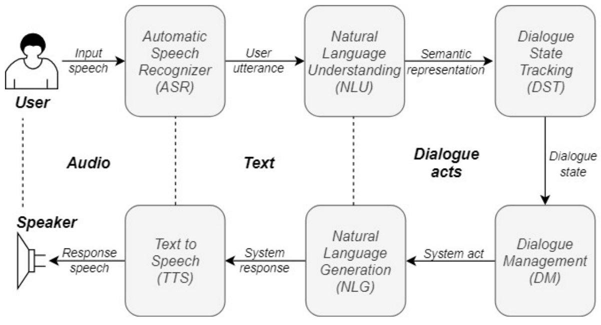

Figure 1: Dialogue-state architecture for dialogue systems.

2.1. Natural Language Understanding

Natural Language Understanding (NLU) allows the system to extract task related information from the user utterances. Traditionally, this information is extracted by three main NLU sub-tasks: *domain classification*, *intent detection* and *slot filling*. Domain classification is responsible for determine the domain of the conversation, while intent detection is aimed at determine what is the general task or goal that the user is trying to accomplish. Finally, slot filling extracts semantic concepts, in particular the slots and fillers that the user intends the system to understand from their utterance [7].

In recent years, deep learning models have been extensively explored in NLU, particularly, Recurrent Neural Networks (RNNs) and Long-Short Term Networks (LSTMs) [11]. Prior work have also shown that joint modeling the NLU tasks can exploit and model the dependencies between them and improve the performance over independent models [3]. In addition, enchanting those models with attention mechanisms can help with long-range dependencies [10].

However, recent breakthroughs in the NLP field introduced a variety of unsupervised techniques for training general purpose language models using a large amount of unannotated text. Those pre-trained models can be fine-tuned on NLP tasks, yielding a significant improvement over training on task-specific annotated data and reducing the time needed for training a specific model. One of those pre-trained models, BERT (Bidirectional Encoder Representations from Transformers) [4], has created state-of-the-art models for a wide variety of NLP tasks. Therefore, it triggered great interest in exploring the use of BERT in the natural language understanding tasks. Chen et al. [3] proposed a joint BERT-based model for both intent detection and slot filing, achieving significant improvements in both intent classification accuracy and slot filing F1, compared to the previous attention-based joint models. Besides that, they also demonstrate a large gain in the sentence-level semantic frame accuracy on the Snips dataset, which include multiple domains and has a large vocabulary, and therefore showing the strong generalization capability of using a general purpose language model [3].

2.2. **Dialogue State Tracking**

Dialogue State Tracking (DST) refers to the task of correctly estimating the state of the dialogue [16]. This task is particularly difficult because of the common ASR and NLU errors, which may cause the system to misunderstand the user. At the same time, the dialogue state tracking is the core of any dialogue system, because the dialogue policy relies on the estimated dialogue state to choose actions.

So far, three families of dialogue state tracking algorithms have been widely explored and implemented: hand-crafted rules, generative models, and discriminative models. Williams et al.[16] give a detailed review on the state-of-the-art in dialogue state tracking, comparing a variety of models in the Dialogue State Tracking Challenge (DSTC), which represents the benchmark challenge for dialogue state tracking.

2.3. **Dialogue Management**

Conventional dialogue systems typically maintain a single hypothesis for the dialogue state, effectively making the assumption that the state is known [6]. A variety of frameworks have been proposed to choose the best action to perform, taking into account the single state [8], [9]. These approaches introduce some advantages, such as bootstrapping, and an easy way of incorporating context knowledge. However, none of them suggest a way of learning which actions should be taken. More recently approaches for dialogue systems allows to maintain a probability distribution over states instead of tracking a single state hypothesis, and thereby include details of the uncertainty in the user inputs. Thus, a more theoretically well-founded framework is to cast the problem as a partially observable Markov decision process (POMDP). We direct the reader to Young et al. [17] for a detailed review of formulating dialogue as a POMDP.

Nowadays, the state-of-the-art approach is to use Reinforcement Learning (RL), in which the dialogue system is seen as a conversational agent, and the dialog system responses are interpreted as the agent actions . Thus, RL allows the agent to learn an optimal policy which can map the state of a dialogue conversation to a system response.

3. Problem Description and Solution

This thesis aims at developing a spoken goal-oriented dialogue system, which can be defined as a computer system able to interact with humans on a task oriented context. Our goal-oriented dialogue system is designed based on the dialogue-state architecture. In a dialogue-state architecture, the system uses a domain ontology $\mathcal{D}$, which represents the kinds of information the system can extract from the users. The ontology defines one or more frames, each a collection of slots, s, and defines the values, v, that each slot can take. The set of slots specifies what the system needs to know, and the filler of each slot is constrained to values that the slot can take.

3.1. **Domestic Domain**

The dialogue system will be implemented in a service robot, operating in a domestic environment. In a domestic environment, a robot must perform a variety of domestic/service tasks. One of the first problems the robot must solve is to discover which task it is being asked to perform and, hence, a slot of the utmost importance should be of the type INTENT. The second problem the robot must solve is to fill the slots of the task. Hence, other two important slots the system must fill are the SOURCE and DESTINATION. As the names suggest, these slots hold information about the beginning and ending of a specific task, respectively. Finally, three more slots are defined: the OBJECT, the PERSON and the WHAT-TO-SAY slots, which refer to the object and the person involved in the task, and the latter refers to what the robot must say.

3.1.1 GPSR and SciRoc

In order to evaluate the performance of service robots, there are two international robotics competitions, RoboCup and European Robotics League (ERL), which have benchmark challenges. One of these challenges is General Purpose Service Robots (GPSR), in which the robot is ordered to perform some domestic tasks. Table 1 gives a brief description of the GPSR tasks. Recently, another project was created to support the ERL tournament in the context of smart cities. The SciRoc challenge focus on smart shopping and is divided in a series of episodes. In one of these episodes, the robot assists people in a coffee shop, and takes care of customers, by taking orders and bringing objects to and from customers' tables. Table 2 describes the task of the SciRoc coffee shop challenge. The developed dialogue system is design for the GPSR task. However, it has been tested in the SciRoc coffee shop challenge, with a few modifications.

Table 1: Tasks and respective description for the GPSR challenge.

Task	Description
Motion	Moves to some place
Meet	Meets a person
Grasp	Grabs an object
Place	Places an object
Follow	Follows a person to a location
Tell	Tells something to someone
Find	Looks for an object or person
Guide	Guides a person to a location
Take	Takes an object from some place to another or gives it to someone

Table 2: Task and respective description for the SciRoc coffee shop challenge.

Task	Description
Order	Order three items (objects) from the coffee shop menu

3.2. Natural Language Understanding

Traditional natural language understanding (NLU) models perform three different sub-tasks: domain classification, intent detection and slot filing. However, when implemented in a dialogue system pipeline, the NLU must also be able to identify the dialogue act type, D-TYPE, which will help the dialogue system to extract some information about dialogue specific actions. Therefore, besides intent detection and slot filing, our NLU model will also perform the dialogue act type classification of the user utterances. The domain classification will be ignored, since our dialogue system will be implemented in a single-domain domestic environment.

Recent state-of-the-art models for the main NLP tasks are based on pre-trained general purpose language models. Because of the lacking of real-world labeled data for our task, we have decided to build a model based on a pre-trained general purpose language model, namely BERT, and exploit its generalization capability. The model architecture of BERT is a multi-layer bidirectional Transformer encoder based on the original Transformer[13], which consists of 12 encoder layers. The BASE version[1] uses vectors with a size of 768 and the multi-head attention mechanisms are composed of 12 heads. BERT takes the input representation of words as a concatenation of WordPiece, positional, and segment embeddings. In addition, a special classification embedding, [CLS], is inserted as the first token to provide a context-dependent representation of the full input sentence, $\boldsymbol{h}_{[\text{CLS}]}$.

We formalize the dialogue act type classification and intent detection as classification problems, while the slot filing task is considered as a sequence labeling task. Sequence labeling requires aligned data. Therefore we need to do aligned labeling in order to provide an alignment between the words in the input utterance and the target semantics, which can be achieved by using BIO tags. Table 3 represents and example of sequence labeling using BIO tags.

Table 3: Sequence labeling using BIO tagging. In this example, the slots are: PER(person)="me", OBJ(object)="book".

Tokens	Bring	me	a	book
Labels	O	B-PER	O	B-OBJ

We developed three independent models. Both dialogue type classification, intent detection and slot filing models are composed of a pre-trained English uncased BERT-BASE model, stacked with a classifier layer (linear layer + softmax). However, in the dialogue type classification and intent detection we classify the whole sentence, while in slot filing we classify each word of the input sentence. Both type of models are represented in Figures 2 and 3.

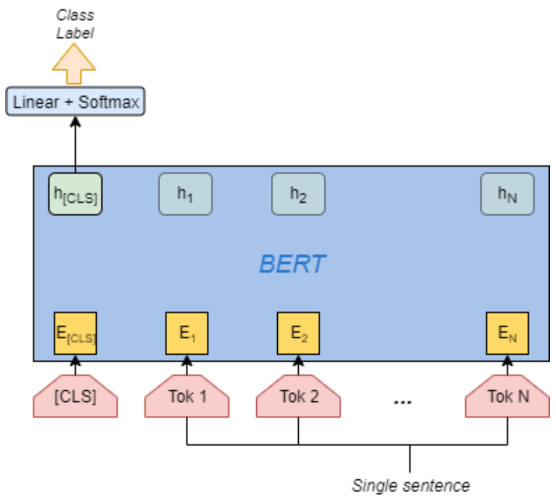

Figure 2: Classification model used for dialogue type classification and intent detection.

[1]BERT developers released two model architectures: BERT BASE and BERT LARGE.

Based on the hidden state of the first special token [CLS], denoted $\boldsymbol{h}_{[\text{CLS}]}$, both the dialogue act type and intent are predicted as:

$$y^d = \text{softmax}(\boldsymbol{W}^d \boldsymbol{h}^d_{[\text{CLS}]} + \boldsymbol{b}^d) \tag{1}$$

$$y^i = \text{softmax}(\boldsymbol{W}^i \boldsymbol{h}^i_{[\text{CLS}]} + \boldsymbol{b}^i) \tag{2}$$

where $\boldsymbol{W^d}, \boldsymbol{b^d}$ and $\boldsymbol{W^i}, \boldsymbol{b^i}$ are the weights and biases associated with the classifier layer, and $\boldsymbol{h}^d_{[\text{CLS}]}$, $\boldsymbol{h}^i_{[\text{CLS}]}$ are the hidden state representations of the special token [CLS], for the dialogue act type and the intent, respectively.

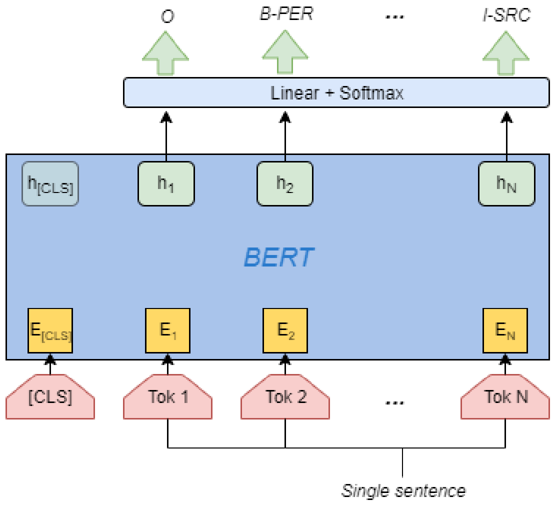

Figure 3: Sequence labeling model used for slot filling.

In addition, each model will have an independent learning objective, which is to maximize the conditional probability $P(y^d|\boldsymbol{x})$ and $P(y^i|\boldsymbol{x})$, respectively. The models are fine-tuned end-to-end via minimizing the cross-entropy loss.

For slot filling, as a sequence labeling task, we need the final hidden states for the tokens representing each of the input words, denoted $\boldsymbol{h}_x = (\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_N)$. We then feed each of the hidden representation into a softmax layer, yielding a label for each token, $y^s_n, n \in 1...N$, where N is the total number of tokens.

$$y^s_n = \text{softmax}(\boldsymbol{W}^s \boldsymbol{h}^s_n + \boldsymbol{b}^s) \tag{3}$$

where $\boldsymbol{W}^s$ and $\boldsymbol{b}^s$ represent the weights and biases of the classifier layer of the slot filing model, and $\boldsymbol{h}^s_n$ is the hidden state representation of the n-th input token.

The objective if formulated as:

$$P(y^s|\boldsymbol{x}) = \prod_{n=1}^{N} P(y^s_n|\boldsymbol{x}) \tag{4}$$

and the learning objective is to maximize the conditional probability $P(y^s|\boldsymbol{x})$. Similar to the previous models, the slot filing model is fine-tuned end-to-end via minimizing the cross-entropy loss.

3.3. Dialogue State Tracker

For the dialogue state tracker (DST), we chose to use a rule-based method, since we had no dialogue data for this particular domain, to train a statistical model. In addition, we also wanted a dialogue state tracker capable of tracking multiple hypothesis and maintain a belief over dialogue states. A particular rule-based method, that achieved significant results in the DSTC benchmark tasks, is a generic DST that maintains beliefs over user goals, particularly over individual slot-value pairs, based on a few simple domain independent rules [14]. These rules use basic heuristic operations, which are derived from some probabilistic mathematics.

Each turn, the dialogue system executes an action and receives an observation, which is an N-best list of user acts (or dialogue acts). First, the user acts with more than one slot-value pairs will be split into single slot-value user acts. Then, the user acts with the same slot-value pairs will be merged across the N-best list by summing their confidence scores, yielding marginal confidence scores for individual slot-value representations. Finally, the dialogue state tracker will use the single slot-value user acts to update the current dialogue state using a set of heuristic rules. Let $P_t(u, s, v)$ denote the marginal confidence score for a user dialogue act $u(s = v)$ at turn t. Then, the belief $b_t(s, v)$ for the slot-value pair (s, v) is updated as [14]:

- Rule 1: If $u = inform$, then $b_t(s, v) = 1 - (1 - b_{t-1}(s, v))(1 - P_t(u, s, v))$.

In addition, we also consider the effects of certain system actions on the belief updates as well. Let $a(h)$ be one of the system actions performed in turn t, where h stands for a set of n slot-value arguments taken by a, i.e. $h = (s_1, v_1), ..., (s_n, v_n)$ [14].

- Rule 2: If a is an implicit or explicit confirmation action, CONFIRM, and an AFFIRM or NEGATE user act u is observed with confidence score $P_t(u)$:
  - Rule 2.1: If $u = affirm$, then $b_t(s_i, v_i) = 1 - (1 - b_{t-1}(s_i, v_i))(1 - P_t(u)), \forall (s_i, v_i) \in h$.
  - Rule 2.2: If $u = negate$, then $b_t(s_i, v_i) = b_{t-1}(s_i, v_i)(1 - P_t(u)), \forall (s_i, v_i) \in h$.

Finally, we add another rule to restart the belief for all slot-value hypothesis when a restart action is requested by the user.

- Rule 3: If a RESTART user act u is observed with confidence score $P_t(u) \geq \tau_R$, then $b_t(s_i, v_i) = 0, \forall(s_i, v_i) \in \mathcal{D}$, where $\mathcal{D}$ is the domain ontology and τ_R is a RESTART THRESHOLD.

3.4. Dialogue Manager

Similarly to dialogue state tracking, we have decided to design the dialogue manager (DM) using a rule-based approach. Briefly, the DM will compute a task state, t_t, and then use a set of hand-crafted rules to choose the next system's action. This task state is computed using a set of task templates, defined in a task ontology, $\mathcal{T}$.

Consider the dialogue state at turn t, $s_t = (b_t(s_1, v_1), ..., b_t(s_M, v_M))$, where M denotes the total number of slot-value pairs, and a system response dialogue act generated in the same turn, $d_t^a = a(h)$, where a denotes the dialogue act type and $h = (s_1, v_1), ..., (s_k, v_k)$ is a set of k slot-value arguments taken by a.

First, the dialogue manager verifies if the dialogue state contains a value for the INTENT slot. If not, the system cannot know the user's goal. Therefore, it must ask for that specific slot, generating the system response dialogue act REQUEST(INTENT).

- Rule 1: if $(intent, v) \nexists s_t$, then $d_t^a = a(h) =$ REQUEST$(intent)$.

On the other hand, when the dialogue state contains a value for the INTENT slot (e.g. INTENT=TAKE), but its confidence is lower than a SLOT THRESHOLD, τ_s, then the system generates the dialogue act, CONFIRM(INTENT=TAKE).

- Rule 2: if $(s = intent, v) \exists s_t$, with $b_t(s = intent, v) \leq \tau_s$, then $d_t^a = a(h) =$ CONFIRM$(intent, v)$.

Otherwise, if the dialogue state has a value for the INTENT slot and is confidence enough about it, it chooses the task templates related to that specific INTENT. Each task template has a set of required slots, s_{req} (i.e. slots required to perform that specific task), and also a task confidence score c^t. In this case, the DM starts filling the task templates, using the slot-value pairs in the dialogue state. For each task template, the DM searches in the dialogue state for the slot-value pairs whose slot belongs to the required slots of the task, i.e. $(s, v) \in s_t, s \in s_{req}$. If the confidence score of those slot-value pairs is greater than the SLOT THRESHOLD, $b_t(s, v) > \tau_s$, the slot-value pairs are added to the task template. In addition, for each slot-value pair added to the task template, the task confidence is updated as $c_i^t = c_{i-1}^t \cdot b_t(s, v)$. The initial task confidence score is the confidence score of the intent slot, $c_0^t = b_t(intent, v)$.

In the end, at each turn t, we have a set of task templates related to a specific task, defined by the INTENT slots in the dialogue state. Each of these task templates contains slot-value pairs extracted from the dialogue state, whose slots are required to perform that specific task, and a task confidence score. Finally, the DM chooses the task template which contains lesser unfilled required slots. If there are two task templates with the same number of unfilled required slots, the one with the higher confidence score is chosen. Finally, the task state, t_t, is computed using the chosen task template.

Thus, at each turn t, a task state composed of j slot-value pairs, $t_t = (s_1, v_1), ..., (s_j, v_j)$, with a certain task confidence c^t, is computed. If the task confidence score is lower than a TASK THRESHOLD, τ_T, (i.e. the system is not entirely sure of all the information in the task state), the DM generates the system response dialogue act CONFIRM$((s_1, v_1), ..., (s_j, v_j))$.

- Rule 3: for $t_t = (s_1, v_1), ..., (s_j, v_j)$, if $c^t \leq \tau_T$, then $d_t^a = a(h) =$ CONFIRM$((s_1, v_1), ..., (s_j, v_j))$.

If the task confidence score is greater than the TASK THRESHOLD, then the DM will check if the task state (task template) has all the required slots full-filled, i.e. if all the required information is presented. If not, it must generate a system response dialogue act requesting one of the missing slots, REQUEST(s).

- Rule 4: for $t_t = (s_1, v_1), ..., (s_j, v_j)$, $c^t > \tau_T$, if $s_k \exists s_{req}, s_k \notin t_t$ then $d_t^a = a(h) =$ REQUEST(s_k).

4. Experiments and Analysis

In general, the main performance indicator for our dialogue system is the NLU performance, since the dialogue system relies entirely on the semantic information extracted by the NLU for computing the dialogue state, and then uses the dialogue state to generate responses. Thus, our analysis will be focused on the NLU models and their performance.

4.1. Natural Language Understanding

The major problem when developing the dialogue system was the non-existence of any real data related to the tasks we want the robot to perform. The adopted solution was to design our own data generator to generate the training and test datasets. However, for validating the model, we used the available GPSR generator[2], which is the official generator used in the GPSR competition. After generating the user utterances, they were manually labeled with intent labels and slot tags. Nevertheless,

[2]https://github.com/kyordhel/GPSRCmdGen

for dialogue act type classification, we were not able to generate any validation data, since the GPSR generator only generates text commands which correspond to a INFORM dialogue act type. The generated training, test, and validation sets contain 5000, 500, and 100 utterances, respectively. There are 11 slot labels, 10 dialogue act type labels, and 9 intent labels.

4.1.1 Metrics

The evaluation of the models is based on some standard classification metrics: *precision*, *recall* and F_1-*score*. The precision of a model reflects how precise the model is in identifying the true elements of a class, while the recall indicates how many of the relevant items are selected. Together, precision and recall metrics are capable of characterizing the performance of a classification model, since they can evaluate the proportion of items of a specific class that are being correctly predicted as belonging to that class. Usually, the precision and recall metrics can be represented together as the F_1 score, which computes their harmonic mean.

In addition, we also evaluate the model using the AUROC (Area Under the Receiver Operating Characteristic curve), which is a performance measurement commonly used in binary classification problems, and indicate how well the model is capable of distinguishing between classes. We extend this metric to a multi-class classification by using a one vs all methodology.

4.1.2 Results

The NLU models use the English uncased BERT-BASE model, which is pre-trained on a large corpus. Traditionally, two main transfer learning techniques can be chosen when using a pre-trained model. We can use the pre-trained model as feature-extractor or fine-tune it on the new task. We have decided to compare both approaches, and compare their performance.

Therefore, we trained both the intent detection (classification task) and slot filing (sequence labeling task) models using both approaches, and for each training epoch we evaluated the models on the test set. We have decided not to analyze the dialogue act type classification model, since we already have a classification task represented in the intent detection analysis. Figure 4 shows the difference between the performance of the slot filing model, using the feature-extraction (EX) and fine-tuning (FT) approaches. The results for the intent detection model are not presented in this paper, but are similar to the slot filing analysis. The results were obtained by training the models several times using different combinations of hyper-parameters.

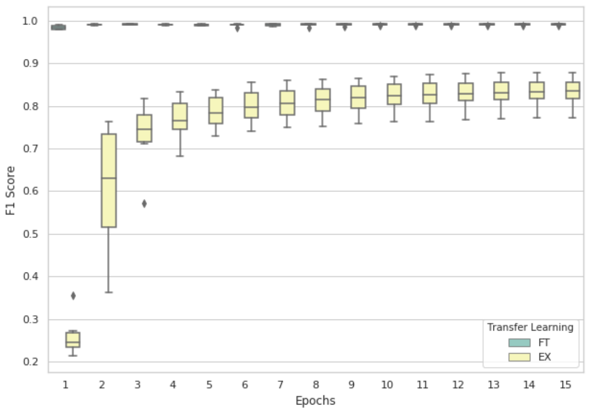

Figure 4: Fine-tuning (FT, green) vs feature extraction (EX, yellow) for slot filing.

It is clear that fine-tuning BERT during training leads to a significant increase in models performance, over the feature-extraction approach. Not only the models are able to achieve a greater F1-Score, but they also require a fewer number of training epochs. Intuitively, the fine-tuning approach enables the adaptation of the pre-trained representations to get better features for the new tasks, which may lead to a better overall performance.

After the previous analysis, we have decided to implement the fine-tuning technique for training the NLU models. For fine-tuning BERT, Devlin et al.[4] keeps most model hyper-parameters the same as in pre-training, with the exception of the batch size, learning rate and number of training epochs. Although task-specific, Devlin et al.[4] suggests a range of possible values to work well across all tasks.

Since fine-tuning is typically very fast, it is reasonable to run an exhaustive search over the above parameters. Therefore, we have trained the models using combinations of different hyper-parameters, and random parameters' initializations using different seeds, for a total number of 15 epochs[3]. The training and validation results will be shown further.

First, we trained the dialogue act type classification model. Figure 5 shows the performance of the model on the test set during training. It is trivial to see that the model is able to predict the data really easy, with a perfect F_1 score after the first epoch. Intuitively, this results may be explained because of the BERT pre-training. BERT represents the input sentence as a context-dependent feature vector. Since BERT was already pre-trained on a large text corpus, it only needs some slightly adaptations on the pre-trained representations for the final classifier layer to be able to easily split the different

[3]Each epoch, we save the model and run an evaluation on the test set.

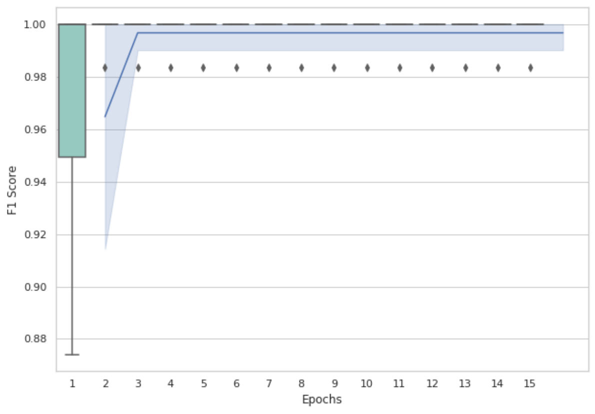

Figure 5: Dialogue act type classification model evaluation on test set during training

classes.

As referred earlier, we were not able to generate a validation set for validating the dialogue act type classification model. However, several tests were performed to check the model's performance, where we tested different type of user commands and checked the model's predictions. Since usually the user commands reflecting confirmations, negations, repetitions, and greetings are fairly simple, the model was able to correctly identify the different dialogue act types of user commands.

Next, we trained the intent detection model. Similarly to dialogue act type classification, this model was also able to achieve really good results after only one epoch (see Figure 6).

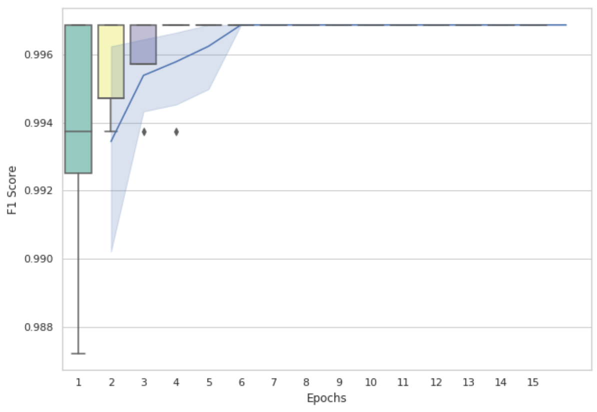

Figure 6: Intent detection model evaluation on test set during training.

In contrast with the dialogue act type classification, for intent detection we were able to validate the model on a validation dataset. As referred before, this validation data was generated using the official GPSR command generator, and then manually labeled with the different intent types. The validation results are shown on Table 4 and Figure 7.

Table 4: Validation results for the intent detection.

	Precision	**Recall**	**F1**
TELL	1.0000	1.0000	1.0000
FIND	1.0000	1.0000	1.0000
GO	1.0000	1.0000	1.0000
TAKE	0.9412	0.9412	0.9412
GUIDE	0.9286	0.8667	0.8966
FOLLOW	1.0000	1.0000	1.0000
ANSWER	1.0000	1.0000	1.0000
PLACE	1.0000	1.0000	1.0000
Micro Average	0.9770	0.9659	0.9714
Macro Average	**0.9765**	**0.9659**	**0.9710**

Table 4 shows that the model is performing really well, with an overall F_1 score of 97.1%. However, it is also clear that the intent labels TAKE and GUIDE are the cases where the model is performing worse, with an F_1 score of 94.1% and 89.7%, respectively. A deeper analysis of the validation dataset indicates that the problem could be related to the fact that both TAKE and GUIDE commands can be represented with the verb take, e.g. *"take the book from the table"* and *"take John to the living room"*. However, by analyzing the model predictions on the validation set, we verified that the model was capable of distinguish between both types of intents, taking into account that a TAKE intent command is usually highly correlated with an OBJECT, while a GUIDE intent command is correlated with a PERSON.

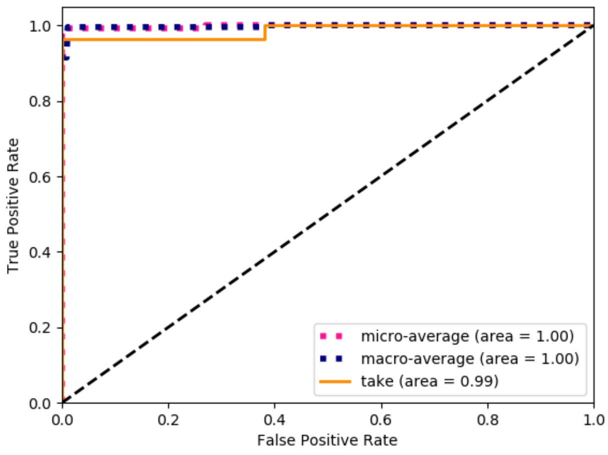

Figure 7: ROC for the intent detection model performance on validation data using a one vs all methodology. Classes with a AUROC of 1.0 are omitted for visualization

Intuitively, this means that the self-attention mechanism from BERT is able to model these correlations. Nevertheless, since the validation set is generated using a more complex generator than the training and test sets, it contains more general and unusual commands, such as *"take the pointing right person to the kitchen cabinet"*. We noted that, in this specific command and others similar to it, the model was not capable of labeling them as a GUIDE intent, because it was not interpreting *"the pointing right person"* as a word sequence describing a PERSON.

Figure 7 shows the ROC curve. For every class, the AUROC is 1.0 (or real close to 1.0), which means that the model has an excellent measure of separability.

Finally, we trained the last model, the slot filing model, whose training results are shown on Figure 8. After the first/second epochs, the model is already performing really well. We saved the model correspondent to the 10th epoch (purple), since it is when the F_1 score stabilizes. Next, we validated the model, yielding the results represented on Table 5 and Figure 9.

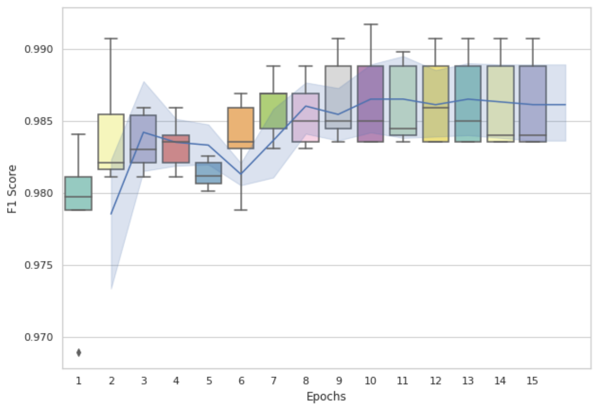

Figure 8: Slot filing model evaluation on test set during training.

Table 5: Slot filing model results on validation data.

	Precision	**Recall**	**F1**
PERSON	0.9792	0.8704	0.9216
OBJECT	0.9444	1.0000	0.9714
SOURCE	1.0000	0.9355	0.9667
DESTINATION	0.9577	1.0000	0.9784
WHAT-TO-TELL	0.9000	0.9000	0.9000
Micro Average	0.9639	0.9492	0.9565
Macro Average	**0.9650**	**0.9492**	**0.9558**

Table 5 shows that the model is having some difficulty labeling the words that correspond to the slot WHAT-TO-TELL. Usually, in the user commands that ask the robot to tell something, the sequence of words that indicate what the robot must say have an high length. For example, consider the command "please tell me the day of the month". To the slot WHAT-TO-TELL, must correspond the value "the day of the month", which has a length of 5. However, the model predicts this by labeling each word individually, using BIO tagging. Sometimes, the model is predicting the label B-WHAT_TO_TELL instead of I-WHAT_TO_TELL, which is causing the performance degradation, since the model considers them as different labels. In addition, we can also verify that the slot PERSON has a lower recall compared with the model overall performance. This is due to the same problem described in the intent detection model, with user utterances similar to *"take the pointing right person to the kitchen cabinet"*, where the model is not predicting the tokens with the label PERSON, and hence the recall drops. Similar to the intent detection model, Figure 9 shows the slot filing model also has an excellent measure of separability.

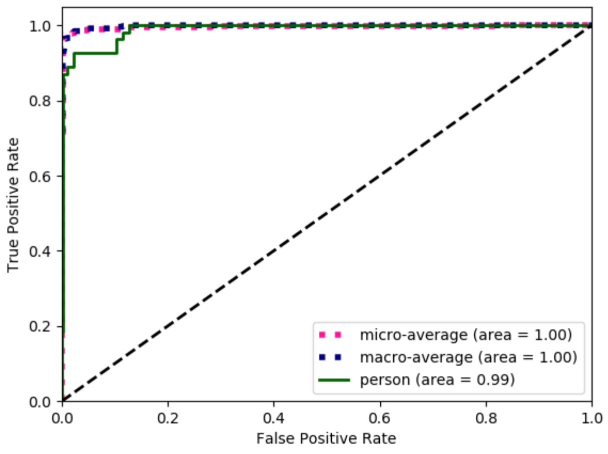

Figure 9: ROC curve for the slot filing model performance on validation data using a one vs all methodology. Classes with an AUROC of 1.0 are omitted for visualization.

The previous results show that, when evaluated on the test set, the models achieve really good results, which was already expected since they were evaluated on a dataset generated with the same generator of the training set. However, the validation set allowed us to check if the model was not overfitting to the training data (since it has been generated using the official GPSR generator). The results on the validation set are still very good, which shows that the model is performing and generalizing well. Nevertheless, the validation data cannot be seen as a representation of the real world data distribution. Thus, the validation of the NLU models on a real world dataset would represent a better indicator for the performance and generalization capability of the model.

4.2. SciRoc

After developing the dialogue system, we implemented it on a service robot that was used to compete in the SciRoc coffee shop task, where the robot must be able to serve customers in a coffee shop, and therefore recognize their orders. For each table, the robot must annotate the orders, composed of three different objects. In summary, during the trials, the dialogue system was able to take the orders from the customers[4]. This reflects the good generalization capability from the NLU models, particularly the slot filing model, which is able to label multiple objects in the same utterance, without even seeing this type of sentences during training. In addition, it is also capable of labeling objects that were not part of the training data. Furthermore, the DST and DM were able to track multiple recognized hypothesis and use the sequential nature of the dialogue to request more information from the users or confirm some of the orders.

5. Conclusions

The main purpose of this work was to design a spoken goal-based dialogue system capable of dealing with the uncertainty in speech recognition. This objective is achieved by three main contributions: an NLU component which takes advantage of using BERT to exploit its great performance and generalization capability; a DST capable of tracking multiple recognized hypothesis and maintain beliefs over slot-value pairs; a DM that generate responses based on the dialogue state, which enables the system to use the sequential nature of dialogue to disambiguate in the presence of errors. As final remark, we were able to test the dialogue system in a real world environment where, in general, it was capable of handling most of the conversations and was able to successfully take the orders.

5.1. Current Limitations and Future Work

In general, the developed spoken dialogue system performs well in simple environments. Nevertheless, it is still restricted by the NLU performance, since it relies entirely on the semantic information extracted by the NLU for computing the dialogue state. Although performing and generalizing well, the NLU still holds some limitations, particularly because the models were trained using generated datasets and not real world data. In addition, including the system actions from previous dialogue turns in the input features, together with the user utterances, may help overcoming some of those limitations. Jointly modeling the three tasks would also enable the model to exploit the relationships between the tasks, increasing the performance. Finally, the use of a structured predictor in the slot filing model, such as a conditional random field (CRF), can help modeling the relationships between the labels.

[4]A video of a real conversation from the competition is available at https://youtu.be/FGTSXI_By7s

References

[1] S. Arora, K. Batra, and S. Singh. Dialogue system: A brief review. *CoRR*, abs/1306.4134:2–5, 2013.

[2] D. G. Bobrow, R. M. Kaplan, M. Kay, D. A. Norman, H. Thompson, and T. Winograd. GUS, a frame-driven dialog system. *Artificial Intelligence*, 8:155–173, 1977.

[3] Q. Chen, Z. Zhuo, and W. Wang. Bert for joint intent classification and slot filling. 02 2019.

[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[5] A. L. Gorin, G. Riccardi, and J. H. Wright. How may I help you? *Speech Communication*, 23(1):113–127, 1997.

[6] M. Henderson. *Discriminative Methods for Statistical Spoken Dialogue Systems.* PhD thesis, University of Cambridge, 2015.

[7] D. Jurafsky and J. H. Martin. *Speech and Language Processing- An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall, 3 edition, 2008.

[8] S. Larsson and D. Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 5(3/4):323–340, 2000.

[9] S. Lee and M. Eskenazi. "Voicexml". *Communications of the ACM*, 43(9):53, 2000.

[10] B. Liu and I. Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. *In Interspeech, San Francisco, CA, USA*, pages 685–689, 2016.

[11] P. Martins. Human-robot speech interaction for service robots. Master's thesis, Instituto Superior Tecnico, Lisboa, 2017.

[12] S. Seneff, L. Hirschman, and V. W. Zue. Interactive problem solving and dialogue in the ATIS domain. *Proceedings of the Workshop on Speech and Natural Language, Pacific Grove, California*, pages 354–359, 1993.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017.

[14] Z. Wang and O. Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. *Proc SIGdial Conf on Discourse and Dialog, Metz, France*, 2013.

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

[16] J. Williams, A. Raux, and M. Henderson. The dialog state tracking challenge series: A review. *Proc SIGdial Conf on Discourse and Dialogue*, 7(3):4–33, 2016.

[17] S. Young, M. Gasic, B. Thomson, and J. D. William. POMDP- based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, PP(99):1–20, 2013.

[18] V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington. JUPITER: A telephone-based conversational interface for weather information. *IEEE Transactions on Speech and Audio Processing*, 8(1):85–96, 2000.