# A Comparison of Learning Approaches to Dialogue Management in Conversational Systems

Preprint

**David Griol, Zoraida Callejas**
Software Engineering Department, University of Granada, Spain
dgriol@ugr.es

## Abstract

Dialogue systems have an increasingly higher number of applications and so their development is raising interest both in academic and industrial setting. Dialogue management is a key aspect for the development of these systems, as it is in charge of the decision making processes and the identification of the most appropriate responses to the user inputs. In this paper we compare different statistical techniques to train dialogue managers using generally available corpora at the disposal for the scientific community. Our results show that the use of generative models and in particular an intent classifier with Neural Networks and Seq2Seq using GRU cells attain the best accuracy for dialogue management.

***K*eywords** Dialogue systems, Dialogue management, Statistical techniques, Deep Learning.

## 1 Introduction

Dialogue systems (also known as Conversational interfaces or Chatbots) are computer programs that are able to maintain a dialogue with humans using natural language [1]. They are currently used in a variety of applications including costumer services, education, e-government, healthcare, entertainment, etc [2].

Task-Oriented Spoken Dialogue Systems are designed to facilitate users to complete a specific task (e.g., order food in a restaurant, book a ticket, or schedule a meeting). Figure 1 shows the classical architecture to develop Task-Oriented Dialogue Systems, in which different modules emulate the same processes that we follow to maintain a conversation: recognize the words in the speech signal (automatic speech recognition, ASR), understand the meaning of these words (natural language understanding, NLU), decide the next system action (dialogue management, DM), translate these action into a sentence(s) in natural language (language generation, LG), and transmit this message by means of a speech signal (text-to-speech synthesis, TTS).

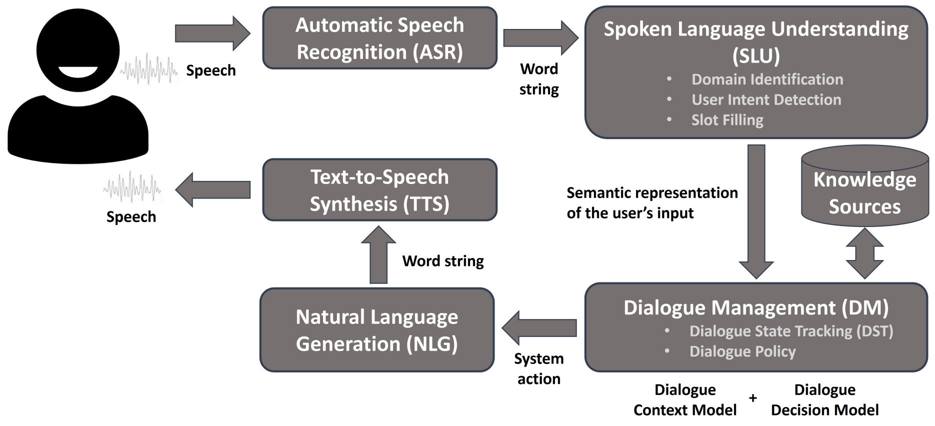
Speech
Automatic Speech Recognition (ASR)
Word string
Spoken Language Understanding (SLU)
Domain Identification
User Intent Detection
Slot Filling
Semantic representation of the user's input
Knowledge Sources
Text-to-Speech Synthesis (TTS)
Speech
Word string
Natural Language Generation (NLG)
System action
Dialogue Management (DM)
Dialogue State Tracking (DST)
Dialogue Policy
Dialogue Context Model
+
Dialogue Decision Model

Figure 1: Classical architecture to develop Task-Oriented Spoken Dialogue Systems

The focus of this paper is on the dialogue management task [3, 4]. This modules takes as input the information provided by the NLU module after processing each one of the users' utterances. This input usually informs the system about what the user really wants to obtain from their utterances (user intent detection) and possible entities (slots) with values required by the system to access the data repositories and generate the system response. To illustrate this, in a system designed for booking train tickets there would be intents such as *Buy_Ticket*, *Consult_Timetable* or *Modify_Reservation*. If the user utterance is "I want to buy a ticket from London to Manchester tomorrow", the system should recognize the *Buy_Ticket* intent and the entities *Origin* (London), *Destination* (Manchester), and *Date* (tomorrow). The statistical development of a dialogue manager involves two main processes [2]:

- State tracking, which updates the dialogue state in the current conversation after processing each user utterance, considering the complete set of slots provided during the dialogue history and the results of accessing the data repositories (e.g., there are not trains available for the values of the slots provided, there is only one train, or more than one train available).

- Dialogue Policy, which decides the next system action for the corresponding dialogue state. This model can require the user to provide additional information, confirm the value(s) of a specific slot(s), complete a query to the data repositories and inform the user, etc.

The conventional processing chain shown in Figure 1 has also some limitations [5, 6, 7], such as some modules depend on others (DM depends on NLU, etc.). Secondly, since the system is composed of several modules, it may be difficult to determine the source of an error. Finally, the slots and features are defined in advance and cannot be changed easily without retraining again the complete set of modules.

Due to these limitations, recently, some end-to-end frameworks for Task-Oriented Dialogue Systems have been introduced. For instance, Bordes et al. [8] proposed an end-to-end dialogue system based on memory networks in which all components are trained from the dialogues themselves. Zhao and Eskenazi [7] developed an end-to-end framework that replaces the NLU, the state tracking and the dialogue policy modules with a single module that can be jointly optimized using a variant of Deep Recurrent Q-Networks (DRQN).

In this paper, we evaluate a set of generative and retrieval-based approaches for selecting the system responses in the dialogue management process. Retrieval-based models are easier to implement and use the user utterance and the context of the conversation for selecting the best response from a repository of predefined responses (candidate responses). The most important advantage of using these models is that they ensure that all the answers are grammatically correct and not inappropriate as they only rely on a database to search in. The heuristic used to select the most appropriate response based on the input and context can range from rule-based architecture to Machine Learning classifiers. For instance, Bartl et al. [9] presented a retrieval-based model with embeddings using Locality-Sensitive Hashing Forest (LSH Forest) and an Approximate Nearest Neighbour (ANN) model to find similar conversations in a corpus and rank possible candidates.

Generative models do not rely on predefined responses, whereas they generate utterances word by word from scratch. Nevertheless, some disadvantages of these models are that it is necessary a huge amount of data to train them and they are more difficult to build. In addition, their answers often have grammatical errors and are meaningless. Most of these systems are based on the RNN Encoder-Decoder, also known as Sequence to Sequence (Seq2Seq) architecture, for generating the text.

We provide a comparative assessment of both kinds of proposals using a set of corpus which have been used as common testbed for the task of dialogue state tracking[1]. The remainder of the paper is as follows. Section 2 describes the proposed statistical techniques used to develop the different proposals for dialogue management. Section 3 presents the main characteristics of the reference corpora used to implement and evaluate our proposals. Section 4 describes the different techniques and the results of their evaluation. Finally, Section 5 presents the conclusions and future research lines.

## 2 Proposed techniques

Recurrent Neural Networks (RNNs) are neural networks whose previous outputs are used as input to the current step with the use of hidden layers. RNNs are quite useful for many natural language processing and speech recognition tasks [2]. The main problem with these networks is that, in practice, they can only retain recent information from a sequence, and it is difficult for them to access information from a long time ago.

Long Short-Term Memory units (LSTMs) can deal with this problem to learn long-term dependencies. They can decide which information will be stored and which will be erased. Their cell state transfers the relevant information through the entire sequence chain. Gates can remove or add information to the cell state. These gates are composed of a sigmoid neural network layer and a pointwise multiplication operation. The sigmoid activations output a number between 0 and 1. The input gate decides the values that will be updated. Finally, the output gate determines the parts of the cell state to output. Gated Recurrent Units (GRUs) have a simpler structure because they do not have the cell state and use the hidden state to transfer information. They include an update gate, which is a combination of the forget and input gates. Thus, GRUs are usually more efficient and faster to train.

Sequence-to-Sequence (Seq2Seq) models have two main components, an encoder and a decoder. There are multiple ways to implement these two components, but in this paper, we focus on LSTM and GRU cells [10] given the number of current applications in Natural Language Processing (dialogue systems, text summarization, machine translation, image captioning and more applications. As its name suggest, in this model, both the input (question) and the output (answer) are sequences. The encoder is an RRN that takes as input a sequence of words (sentence) and its goal is to convert this input into a fixed size vector. Each hidden state influences the next hidden state and the last hidden state is called the

[1]https://dstc9.dstc.community/ (Last access: June 2021)

context (also known as thought vector) and contains the summary of the sequence and represents its intention. Once the decoder receives the thought vector, the second RNN (decoder) generates another sequence computing one word by word. At each timestep, the decoder should be influenced by the context vector and the previous decoded words.

A significant limitation of the basic Seq2Seq model is that they do not consider the context of the conversation. For instance, it is necessary to know the previous dialogue turns in a conversation to answer a kind of question such as "Could you say that again?". Another weakness of this architecture is that it does not decode large sequences properly. However, this limitation can be fixed by adding the Attention Mechanism proposed in [11].

Using the Attention Mechanism, we allow the decoder of the model to focus only on some parts of the source sentence but not the whole text. With this mechanism, a bidirectional encoder (BiRNN) is used to encode the input sentence, word by word, into one forward and one backward RNN. The forward RNN reads the input as it is ordered, from the start to the end, while the backward does it in reverse, going from the last to the first word. The final bidirectional hidden cell state vector is the concatenation of the hidden state going forwards and the hidden state going backwards computed by both RNNs. The concatenated state contains an annotation for each word that contains the summaries of both the preceding and the following words. Then, the context vector is computed as a weighted sum of these annotations. Using this attention mechanism in the decoder, the encoder does not need to encode all information in the source sentence into a fixed-length vector and the information can be spread throughout.

## 3 Datasets

Three dialogue corpora have been used for training our Retrieval-based and Seq2Seq models: the data released for the Dialogue State Tracking Challenges DSTC2 [12] and DSTC3 [13], and the Self-dialogue Corpus [14].

DSTC2 is a labeled dialogue corpus of dialogues collected using Amazon Mechanical Turk (AMT) and consists of over 2.000 dialogues in a restaurant information domain. This system is able to offer the user information and recommendations about restaurants in the city of Cambridge based on the user's preferences (type of food, area, price range, etc.).

The dataset is distributed as a collection of calls (dialogues) and each call has two log objects with a unique ID, the date and a list of the turns, which give the utterance from the user and the output of the system. For each turn, it is provided the index of the turn, the transcription of what the user said, the dialogue-act representation of that utterance and a list with the slots requested by the user.

In DSTC2 there are eight types of slots: area, food, name, price-range, address, phone, postcode, and signature. Users can provide a value for these slots and they can also be requested by the system. There are 20 classes associated to the set of different system actions: 'affirm', 'any-area', 'any-food', 'any-price', 'anything-else', 'bye', 'greeting', 'inform-area', 'inform-area-food', 'inform-area-price', 'inform-food', 'inform-food-price', 'inform-food-price-area', 'inform-price', 'request-addr', 'request-phone', 'request-addr-phone', 'request-postcode', 'request-phone-postcode', 'request-addr-postcode'. DSTC3 has also 2.000 dialogues also collected using AMT, in the tourist information domain and it adds the following types of slot to the ones in DSTC2: 'children allowed', 'has internet', 'has tv', 'near', 'type'. This dataset has the same user and system dialogue acts than DSTC2.

The Self-dialogue Corpus was also collected using AMT. This dataset contains 24,165 dialogues (141,945 turns, 3,653,313 words) across 23 topics that include movies, music, sports, etc. A total of 2,717 users participated in the acquisition of the corpus.

## 4 Proposed methodologies

Section 4.1 and 4.2 describe our proposals for statistical dialogue management using respectively retrieval-based and generative models. In addition, we describe the results of the comparative assessment of these proposals using the reference corpora described in the previous section.

### 4.1 Retrieval-based model

The main objective of retrieval-based approaches for dialogue management is to estimate the next system action (i.e., system dialogue act) based on a classification process that considers the current dialogue state as input. The dialogue state includes the slots mentioned in the previous user utterances. For the Dialogue Manager to select the answer, we has assumed that the only information that these models consider to determine the next system response is the presence or not of slots and dialogue acts during the previous dialogue history, and the confidence scores assigned to each one of these information pieces.

Table 1: Accuracy results of the retrieval-based model for the DTSC2 and DSTC3 datasets

| **Model** | **Accuracy DSTC2** | **Accuracy DSTC2 & DSTC3** |
|---|---|---|
| Decision Trees | 0.614 | 0.602 |
| Random Forest | 0.712 | 0.687 |
| Support Vector Machines (SVM) | 0.685 | 0.614 |
| Neural Networks | 0.694 | 0.662 |

For the classification function we have evaluated four different approaches. Decision Trees predict the values of the class by learning simple decision rules inferred from the input variables and is used a baseline model. Random Forest consists of many Decision Trees that work as an ensemble. Each tree in the Random Forest outputs a class prediction and the class with the most votes becomes the prediction of the model. Support Vector Machines (SVM) have the main objective of finding a hyperplane in an N-dimensional space that distinctly classifies the data points. Finally, we used the Keras library to also evaluate neural Network-based classifiers, for which the rectified linear unit activation function (ReLU) was used on the first 3 layers and the Softmax activation function in the output layer. The Dropout method was also applied to prevent it from overfitting. We also defined the optimizer as the gradient-based algorithm Adagrad and used categorical cross entropy as the loss function.

The DSTC2 and DSTC3 datasets were divided into training (80%) and test (20% )subsets and a 5-fold cross validation was used to complete the evaluation process. Firstly, we used only the DSTC2 dataset (restaurant information domain), which consist of 13.692 instances with 259 features and 24 labels. Then, we used the data from DSTC3 (tourist information domain) plus the restaurant data from DSTC2 for training. This dataset has 27.709 instances with 390 features and 48 labels.

Table 1 shows the accuracy results obtained in both steps of the evaluation. As it can be observed, the best score for the first experiment was obtained with the Random Forest model (accuracy of 0.712). One of the main problems found was related to the differences in number of samples for each one of the classes (from 1,962 to 15 depending of the class). The performance of the different classifiers are slightly lower in the second experiment. Again, the best result is obtained using the Random Forest model, with an accuracy of 0.687.

### 4.2 Generative model

In the second proposal, we designed a dialogue management framework combining three main components: a user intent classifier, a finite state machine to consider the dialogue context, and a generative model to deal with non-task utterances.

#### 4.2.1 Intent classifier

When the user asks a question to the system, the intent classifier classifies the user utterance into one of the predefined intents (such as *request-phone*). In addition, intent matches have a confidence value in a range from 0.0 (completely uncertain) to 1.0 (completely certain). If the confidence value is higher than a rating threshold, a system response will be selected according to the intent detected.

We have evaluated different classifiers to compare their performance and select the one with the best accuracy: Naïve Bayes, Logistic Regression, Support Vector Machines and Neural Networks. Logistic regression is a predictive analysis used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Naïve Bayes classifiers are highly scalable probabilistic classifiers that apply Bayes' theorem considering strong independence assumptions between the features.

The intent classifiers were trained using the statements and their respective intents in the DSTC2 and DSTC3 corpora. As it can be observed in Table 2, SVM and 2-layer Neural Network classifiers achieved the best accuracy values. These algorithms have a very similar accuracy, so we have used both alternatives in the experiments where all the modules are used.

#### 4.2.2 Finite State Machine

A Finite State Machine (FSM) has also been trained for state management and keep track of the conversation. According to the user intent selected in the previous step, the FSM transits to a specific state. Figure 2 shows the different states of the finite state machine and the different transitions.

Table 2: Accuracy results of the intent classifier for the DTSC2 and DSTC3 datasets

| **Model** | **Accuracy DSTC2 & DSTC3** |
|---|---|
| Naïve Bayes | 0.711 |
| Logistic Regression | 0.749 |
| SVM | 0.614 |
| Neural Networks | 0.754 |

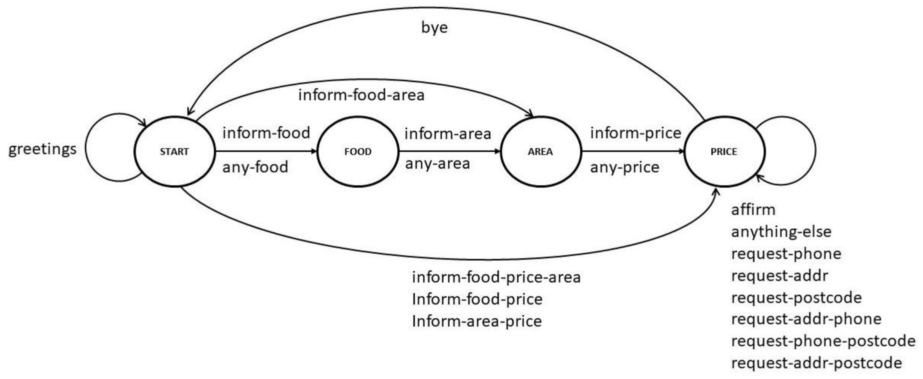

Figure 2: Finite State Machine designed for the generative model

For instance, when the user says, “I want an Indian food restaurant”, this utterance is classified to the intent “inform-food”. Then the system returns the predefined answer “What part of town do you have in mind?” and the state is changed to “FOOD”. Next, the user replies “In the north of the town”, which is classified to “inform-area”. The system will answer “Would you like something in the cheap, moderate, or expensive price range?” and the state will be updated to “AREA”. As it can be seen in the diagram, in order to request information from a restaurant we need to be at the state “PRICE”, that is to say, the user has already provided the type of food (“inform-food”), area (“inform-area”) and price of the restaurant (“inform-price”) that he/she wants to look for. Once this state has been reached, it is possible to request information about the restaurant recommended by the system, ask for another option (“anything-else”) or use the intent “bye” to clear the context and move to the state “START”.

### 4.2.3 Seq2Seq model with Attention Mechanism

If the confidence value given by the user intent classifier is lower than the predefined threshold, a generative model (Seq2Seq model with Attention Mechanism) is used to process the input and generate the text of the answer. This model has been trained with the Self-dialogue Corpus. Thus, the resulting statistical dialogue manager is able to answer questions that do not strictly fall within the DSTC2 and DSTC3 domains.

This model has been built using the PyTorch library. The encoder consists of an embedding layer and RNN layers. The model uses the embedding layer to go from an integer representation (that represent each word) to the dense vector representation (embedding vector) of the input. It looks up the input integers into the dictionary and returns the associated vectors. Next, these embeddings are used as input for the RNN and its output is the thought vector that the decoder will use. The decoder is an RNN, in our case a unidirectional 2-layer LSTM or GRU, that generates the response sentence producing one word at a time until it outputs an END token representing the end of the sentence. The attention layer after the RNN (LSTM or GRU) uses the previous decoder hidden state and the hidden state of the encoder at that time step to calculate the attention weights, which are then multiplied by the encoder outputs to get a context vector, which will be used to predict the next system response.

To evaluate the performance and effectiveness of combining the intent classifier with the finite state machine and the generative model, we designed 4 models to be compared:

- Model 1 (SVM+GRU): Intent classifier with SVM and Seq2Seq using GRU cells.
- Model 2 (NN+GRU): Intent classifier with Neural Networks and Seq2Seq using GRU cells.
- Model 3 (SVM+LSTM): Intent classifier with SVM and Seq2Seq using LSTM cells.
- Model 4 (NN+LSTM): Intent classifier with Neural Networks and Seq2Seq using LSTM cells.

Table 3: Accuracy results of the generative model for the DTSC2 and DSTC3 datasets

| **Model** | **Accuracy DSTC2 & DSTC3** |
|---|---|
| Model 1 (SVM+GRU) | 0.793 |
| Model 2 (NN+GRU) | 0.849 |
| Model 3 (SVM+LSTM) | 0.814 |
| Model 4 (NN+LSTM) | 0.837 |

Table 3 shows the results of the evaluation of the four models. As it can be observed, Neural Networks are a better option than the SVM algorithm for building the intent classifier. With regard the Seq2Seq model, the results using GRUs are slightly better than using LSTMs. GRUs are also computationally more efficient and take much less time to train.

## 5 Conclusions and Future Work

In this paper we have proposed two main proposal to develop statistical dialog manager for conversational systems. The first approach is based on a retrieval-based model based on a classification process that considers the encoded dialog state as input to select the next system response (i.e., system dialogue act). Different definitions of the classification function has been evaluated using the corpora provided in the DSTC2 and DSTC3 dialogue state tracking challenges. The best accuracy results for this comparative assessment has been obtained using a Random Forest classifier (0.712).

The second approach is based on a generative model that combines an intent classifier, a finite state machine to manage goal-oriented conversations and a Seq2Seq model to deal with informal conversations. For the intent classifier we have also evaluated different classification functions with the same corpora, obtaining the best accuracy results using a Neural Network. For the Seq2Seq model, we have concluded that the results using GRUs are slightly better than the ones obtained using LSTMs (0.849).

As future work, we want to extend the evaluation of our proposals using corpora with a larger number of samples. We also want to evaluate alternatives for coding the dialogue state considering the context information compiled during the dialogue history. Finally, we want to also consider additional deep learning techniques based on Transformers models.

## 6 Acknowledgments

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823907 (MENHIR project: https://menhir-project.eu).

## References

[1] Michael McTear, Zoraida Callejas, and David Griol. *The Conversational Interface: Talking to Smart Devices*. Springer, 2016.

[2] Michael McTear. *Conversational AI. Dialogue systems, Conversational Agents, and Chatbots*. Morgan and Claypool Publishers, 2020.

[3] L. Mateju, D. Griol, Z. Callejas, J.M. Molina, and A. Sanchis. An empirical assessment of deep learning approaches to task-oriented dialog management. *Neurocomputing*, 439:327–339, 2021.

[4] David Griol, Zoraida Callejas, Ramón López-Cózar, and Giuseppe Riccardi. A domain-independent statistical methodology for dialog management in spoken dialog systems. *Computer Speech & Language*, 28:743–768, 2014.

[5] Qingbin Liu, Guirong Bai, Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. Heterogeneous relational graph neural networks with adaptive objective for end-to-end task-oriented dialogue. *Knowledge-Based Systems*, 227:107186, 2021.

[6] Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *Proc. of ACL'20*, pages 583–592, 2020.

[7] Tiancheng Zhao and Maxine Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proc. of SIGDIAL'16*, pages 1–10, 2016.

[8] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. In *Proc. of ICLR'17*, pages 1–15, 2017.

[9] Alexander Bartl and Gerasimos Spanakis. A retrieval-based dialogue system utilizing utterance and context embeddings. In *Proc. of ICMLA'17*, pages 1120–1125, 2017.

[10] Zhiqiang Ma, Baoxiang Du, Ji Shen, Rui Yang, and Jianxiong Wan. An encoding mechanism for seq2seq based multi-turn sentimental dialogue generation model. *Procedia Computer Science*, 174:412–418, 2020.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *Proc. of ICLR'15*, 2015.

[12] Matthew Henderson, Blaise Thomson, and Jason D Williams. The second dialog state tracking challenge. In *Proc. of SIGDIAL'14*, pages 263–272, 2014.

[13] Matthew Henderson, Blaise Thomson, and Jason D Williams. The third dialog state tracking challenge. In *Proc. of SLT'14*, pages 324–329, 2014.

[14] Joachim Fainberg, Ben Krause, Mihai Dobre, Marco Damonte, Emmanuel Kahembwe, Daniel Duma, Bonnie Webber, and Federico Fancellu. Talking to myself: self-dialogues as data for conversational agents. *arXiv:1809.06641*, 2018.