uc3m | Universidad Carlos III de Madrid

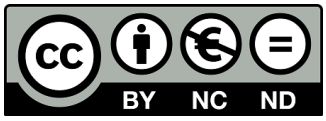
University Degree in Computer Science and Engineering
2018-2019

Bachelor Thesis

"Development of customized conversational interfaces with Deep Learning techniques"

Pablo Cañas Castellanos

Tutor: David Griol Barres
Co-Tutor: Juan Manuel Alonso Webber
Leganés, June 2019



SUMMARY

This Bachelor's thesis will cover the end-to-end process of developing a personalized conversational interface for a specific domain, using Deep Learning techniques. In particular, it will focus on the study of the Dialog Manager module, which is in charge of deciding the next system response based on the current dialog state.

Although there is plenty of literature about Machine Learning applied to the construction of dialog management models, there is very little reference to the utilization of Deep Learning for such task. As a result, this work analyzes the improvement that deep neural networks can bring to accuracy. Several models are created with TensorFlow, and comparisons are made with traditional Machine Learning solutions. Results show that Deep Learning is not the most recommended approach for this type of problems, yet further research is suggested for more complex datasets.

After this, one of the Deep Learning models, based on a train scheduling domain, is used for the implementation of the dialog manager inside a real spoken dialog system. To integrate the rest of required components of such technology (automatic speech recognizer, natural language understanding module and text-to-speech service), a modern framework is used: DialogFlow. With this platform, a complete chatbot is built in the form of an assistant in the train scheduling domain.

Evaluation of the spoken dialog system with real users generates a very positive feedback, demonstrating that a Deep Learning based dialog manager is a valid solution in commercial conversational interfaces.

Keywords: Conversational Interfaces, Spoken Dialog Systems, Machine learning, DialogFlow, Deep Learning, TensorFlow

DEDICATION

This thesis is the culmination of the greatest educational experience of my life. Not only I have learnt to become a competent Computer Science professional, but I have also grown up to a person that I can feel proud of and that is living accordingly to his major interests and principles.

I would like to dedicate my work to every person that has accompanied me during this wonderful journey. In special, I would like to thank my parents, María Jesús and Pedro, for supporting me in every decision that I took, shedding light on every problem that I encountered, and teaching me correct moral values; and my brothers, Jorge and Sergio, for being direct authors of the joy in my life.

The deepest gratitude goes to every relative, friend and person that brings me back to happy memories. There have been innumerable lifetime experiences that I will carry with me forever and that have shown me that the most important thing in life is the people whom you share your path with. There are way too many people to mention them all, but those include friends from my neighbourhood, university (LaVendiciónDevs), Ruta Quetzal, Ruta Inti, San Diego, CERN, travelling mates, and a large et cetera.

Finally, I would like to thank my tutors, David and Juan, for helping me with the construction of this project.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation of Work

Conversational Interfaces (CUI) are systems that simulate interactive conversations with humans [1]. These platforms are meant to display human-like characteristics and support the use of spontaneous natural language in order to make interactions with different purposes, such as performing transactions, responding to questions, or simply to chat [2].

Conversational interfaces have become a key research subject for many companies, as they have understood the potential revenue of introducing these devices in society's mainstream. Virtual Personal Assistants (VPAs), such as Google Now, Apple's Siri, Amazon's Alexa or Microsoft's Cortana, are the clearest examples of this. These VPAs are used for a wide variety of tasks, from setting an alarm to updating the calendar, passing through getting directions, finding nearest restaurants or stores, or even recipe planning and reporting the news. But personal assistants are not the only application of conversational interfaces. Companies can use CUIs to enhance their services and increase customers' satisfaction. They are currently being used for making appointments and reservations, secretarial work, and support service, saving companies the cost related to human labour and being even more efficient than humans in solving such issues.

Interactions with conversational interfaces are designed to be really simple. Figure 1.1 shows an example from an interaction with Google Now personal assistant, which can be activated from any Android smartphone by pronouncing the words "Ok Google".

User: What is the weather like in Madrid today?
Google Now: The weather in Madrid is 18 degrees and sunny.

Fig. 1.1. An example of user interaction with Google Now VPA

Figure 1.3 shows an example of a user requesting information about the weather in Madrid. As it can be observed, the assistant responds with the updated information about the weather. In addition to the spoken response, there is also a visual display of the recognized question, a visual representation of the weather forecast, and other related questions that could be asked (Figure 1.2).

Fig. 1.2. Result to weather request from Google Now

But not only virtual personal assistants can retrieve the information that is being asked. Besides, they can make several actions, such as buying a product or making an appointment in the calendar. Figure 1.3 displays an example of an interaction were the user is asking Google Now to call the contact 'Mom'. Together with the response, that confirms the understanding of the query, the smartphone automatically calls the contact, as it can be observed in Figure 1.4.

User: Call Mom
Google Now: Calling to Mom's mobile...

Fig. 1.3. An example of user request with Google Now VPA

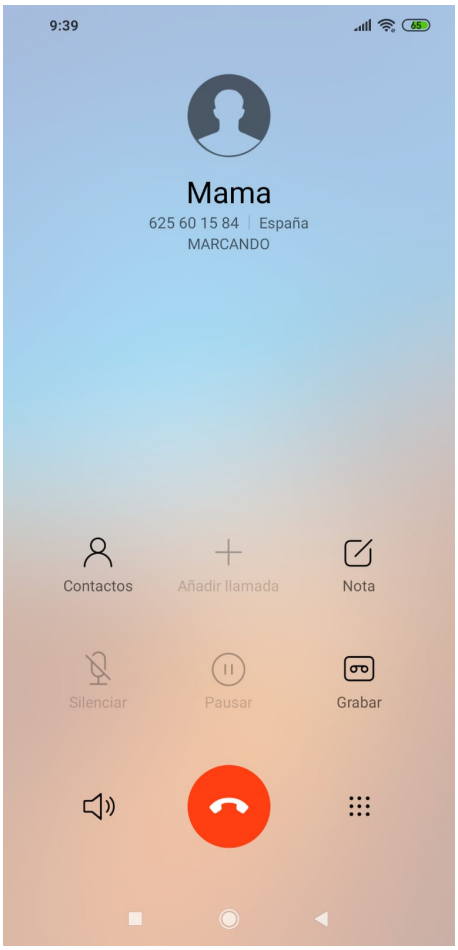


Fig. 1.4. Result to calling request from Google Now

On the other hand, we find a recently developed field in Artificial Intelligence (AI) named Deep Learning (DL) [3]. This area began to grow in the beginning of the decade as an alternative solution to tasks where traditional Machine Learning algorithms were reaching a saturation point. In particular, it had to deal with the treatment of large amounts of data. As a result, Deep Learning has become part of many of the current intelligent systems, such as video surveillance or biological data classifiers. It has also become very popular in the media due to its advantages for the development of self-driving cars. Since its rise, all the industry is strived towards the evolution of this technology, and it seems to keep offering plenty of potential for the future.

The merge of these two technologies is something inevitable and that is going to bring millions of euros in revenues in the years to come. For this reason, there will be a huge industry generated around it, with hundreds of research groups investigating in the area and thousands of jobs created. This is the main motivation of my Bachelor's thesis, to get in touch with the technical background required to understand the topic, and to make a small contribution to this exciting field.

1.2. Goals

The main goal of this bachelor project is to show the whole process of the development of a spoken dialog system with Deep Learning techniques. This process goes from the experimentation with different DL architectures and the analysis of performance for each model, to the implementation and evaluation of a commercial conversational interface solution.

In particular, the objectives of the thesis are:

- To make an study of the importance Deep Learning has for developing a modern conversational interface with respect to traditional Machine Learning techniques, using several domains.
- To learn a reliable approximation of a dialog manager for a specific domain, using Deep Learning techniques.
- To integrate such dialog manager with the rest of the components of a conversational interface.
- To implement a conversational interface as a commercial application that people could use from their devices.
- To evaluate and validate such application by executing a test plan with real users.

1.3. Document Structure

The content of each of the chapters of the report is now exposed, with the objective of showing the organization of the document:

- **Chapter 1: Introduction.** This chapter describes the main motivation behind the project, as well as the main goals pursued by the author. It will also make a brief description of the document structure.
- **Chapter 2: State of the Art.** This chapter makes a review of the literature and the concepts surrounding the development of the thesis. As such, it explains what a conversational interface is and its history, reviews the terms of Machine Learning, Artificial Neural Networks, Deep Learning, Automatic Speech Recognition, Natural Language Processing and Dialog Management, and shows the current tools and frameworks used for the development of such systems.
- **Chapter 3: Deep Learning analysis applied to CUI Dialog Manager.** In this chapter, the experimentation phase of the thesis is presented. Different validated datasets are taken, preprocessed and trained with a variety of Machine Learning and Deep Learning algorithms. Results will be compared and commented, hence

obtaining overall conclusions of the utility of Deep Learning methodologies for dialog management modelling.

- **Chapter 4: Implementation of the Conversational Agent.** This chapter describes the complete conversational agent implementation process. After choosing one of the models created in Chapter 3, the necessary parts for a basic spoken dialog system will be assembled, and a functional conversational agent will be created.

- **Chapter 5: Evaluation of the Conversational Agent.** In this chapter, the previously created system is validated with real experimentation. As a result, the system will be presented to the users for external assessment. Based on the results, an objective will be made, whereas users will also provide subjective feedback from their experience.

- **Chapter 6: Regulatory Framework.** This chapter approaches the thesis from the legal point of view. As a result, it makes an overview of the applicable regulations, as well as the possible issues concerning intellectual property. It also enumerates the technical standards of the project.

- **Chapter 7: Socio-economic Environment.** In this chapter, a detailed analysis of the socio-economic impact derived from the project will be crafted, as well as a justification for the planning and budgeting of the work.

- **Chapter 8: Conclusions.** This chapter will list the main ideas and conclusions extracted from the development of the thesis. It also analyses the future lines of investigation that could be followed from such work.

- **Bibliography.** This chapter is dedicated to the list the online resources, documents, papers and books reviewed and referenced for the development of the thesis.

- **Annexes.** This final chapter lists a glossary of common and technical abbreviations with the objective of facilitating the reading.

2. STATE OF THE ART

This thesis touches many different areas of Computer Science (CS) and Artificial Intelligence, as well as several toolkits and techniques that are commonly used for the development of conversational interfaces. As a result, this section provides an overview of all the concepts and tools used in the project, together with previous work done by other authors, in order to get a better understanding on the context of the thesis.

2.1. Prior Background

The main subject in which this project is based is Machine Learning (ML). Therefore, a small introduction to the topic is made so as to understand the basic aspects and objectives of ML algorithms. After this, Artificial Neural Networks (ANN) algorithm will be explained, as it is the main predecessor of Deep Learning. Finally, the major concepts and related work behind Deep Learning will be exposed.

2.1.1. Machine Learning

Machine Learning is a term that is extremely popular nowadays. We can always find news about several novel applications of this technique, and new ways about how it is improving people's lives. But, what is exactly Machine Learning?

According to Arthur Samuel [4], Machine Learning is the subset of Artificial Intelligence that studies the algorithms and statistical models which computer systems use to effectively perform a specific task without being explicit programmed. Instead, Machine Learning algorithms build mathematical models that are then used to solve a wide variety of tasks relying on patterns and inference [5].

Machine Learning paradigms are classified in three main groups [6]:

- **Supervised Learning**: based on training a model from data source with correct output already assigned. This can be divided in two types of tasks: classification, in which the output variable is a category (e.g. deciding whether to give a credit to an individual or not, detecting if an image is a dog or a cat...), and regression, in which the output variable is a real number (e.g. predicting the price of a house based on its characteristics, predicting the temperature for the next week...).
- **Unsupervised Learning**: based on training a model to identify hidden patterns from unlabelled input data. We can as well divide this group into two categories of algorithms: clustering, which tries to discover the inherent groups in data (e.g. grouping clients by purchasing behaviour), and association, which tries to find rules

that describe large proportions of your data (e.g. people who watch movie X may like also movie Y).

- **Reinforcement Learning (RL)**: based on training a model by maximizing the reward along a particular dimension over many steps. It differs from supervised or unsupervised learning methods in the learning process: while the others try to identify patterns or approximate functions in the data, reinforcement algorithms are goal-oriented, and focus on maximizing the reward. As a result, a reinforcement problem consists of an agent that has several actions to transit between states with an associated reward, and the objective is to arrive to a final state with the maximum possible reward. An example of application of this methodology is traffic light control in a smart city.

A typical Machine Learning work cycle involves the following subtasks:

1. **Generate example data**: in order to build a model, one needs to provide training examples so that the system identifies and learns patterns from them. Therefore, an appropriate sized training set must be collected to build an effective algorithm. This also includes the cleaning and preprocessing of data, so that the relevant information is optimized as much as possible.

2. **Train the model**: once the dataset is ready, the model is trained using a specific ML algorithm. The idea is to use a subset for training and another subset for evaluating the model. Therefore, this is an iterative process in which one tries several algorithms and configurations and keep refining the learning outcome until you manage to achieve the best possible result.

3. **Deploy the model**: this includes the implementation of the ML algorithm in the system for which it was created, and the validation of such product by results monitoring and data analytics.

A representation of the working flow is shown in figure 2.1.

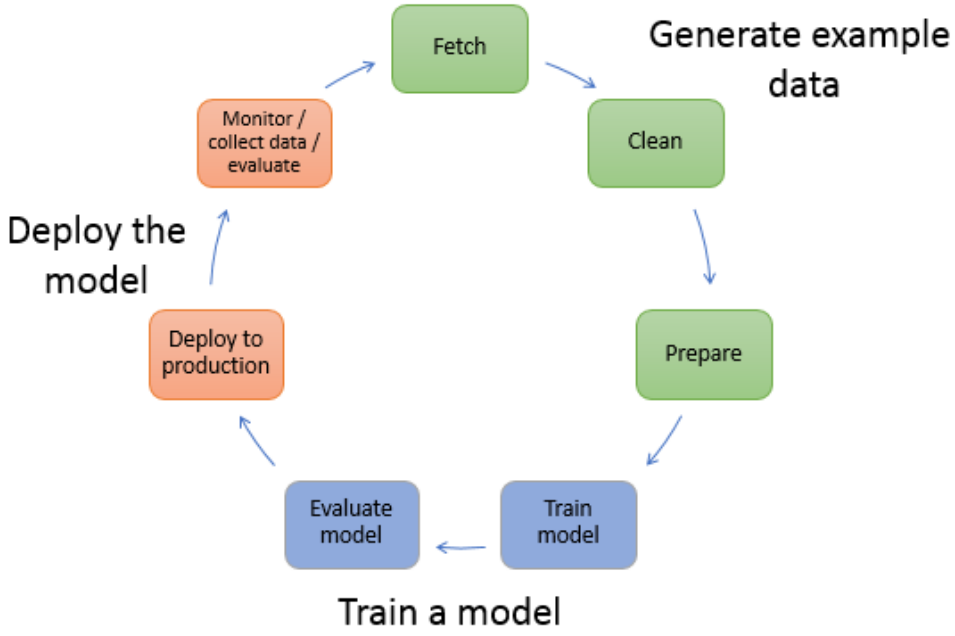


Fig. 2.1. Machine Learning typical working flow [7]

Machine Learning is currently used in a wide variety of areas, going from predicting the next value of a financial asset in the Stock Exchange Market (SEM) to Outer Space exploration [8]. Some of the most common applications from day-to-day life include [9]:

- **Malware filtering**: malicious software has always been a very important concern since the beginning of the digital era. Being affected by a virus could entail the public exposure of people's private data (as we have recently seen with various celebrities [10]) or a loss of billions of dollars for a company [11]. Fortunately, every new malware that is created has a high percentage of similarity with its previous versions, and thus there exists a pattern that a Machine Learning algorithm can learn to prevent those situations.

- **Online customer support**: lately, it has become extremely common for services that were traditionally provided physically to be used through an online platform. As a result, many of these websites offer the possibility to chat with a customer support representative in order to solve any issue that the client may have. However, it has been shown that using a chatbot for such functionality can multiply your investment return and increase the degree of customer satisfaction [12].

- **Product recommendations**: keeping the fidelity of customers in e-commerce sites is vital for the success of those business. Therefore, making predictions on the products that a user may like based on their previous purchases or searches has become a topic of great interest for every multinational company. It is considered to be a task of such importance that Netflix offered a prize of 1 million dollars to the team that improved their movie recommendation system by a 10% accuracy [13].

- **Videos surveillance**: security is one of the main concerns of society. Imagine a single person monitoring multiple cameras at the same time, trying to find suspicious movements from normality, for several hours. Certainly, it is a very difficult job to perform. On the other hand, thanks to new Machine Learning techniques like Convolutional Neural Networks (CNN), processing data from images has become a feasible task. Therefore, such technology is used to improve surveillance services, tracking unusual behaviour, recognizing faces, and ultimately detecting a crime before it happens.

- **Online fraud detection**: Machine Learning is also contributing to build a secure cyberspace for even the most delicate operations, such as online monetary transaction. For example, Paypal is using ML to detect money laundering (see [14]), and traditional banks are blocking credit cards when they spot irregular transactions in a small period of time.

- **Virtual personal assistants**: current birthday gift par excellence, Alexa, Siri or Google Now are some of the most popular virtual personal assistants in the market. They perform a wide variety of tasks, from setting an alarm to ordering a pizza, when asked over voice. These devices use ML to understand what the user is requesting, and they have a high level of accuracy.

Once we have had a brief overview of what Machine Learning is, let's move on to more specific concepts that are closely related to the main work of the thesis.

2.1.2. Artificial Neural Networks

The ML technique that is most commonly use nowadays is artificial neural networks. Haykin [15] describes ANN as a massively parallel combination of simple processing units which can acquire knowledge from environment through a learning process and store the knowledge in its connections.

Artificial neural networks design is inspired by the biological neural networks that constitute animal brains. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it [16].

The first studies related to the area were made by McCulloch and Pitts in 1943 [17]. They were seeing neurons as units that, depending on the connected inputs attached to them, will activate according to a threshold value. They only theorized about the architecture, with no mention to the learning process. However, in 1949 Donald Hebb appeared to introduce Hebbian learning [18]. He proposed that, when two neurons fire together, the connection between them is strengthened. As a result, this activity is one of the fundamental operations necessary for learning.

Some authors used these ideas to create the first known neural networks. Frank Rosenblatt introduced the **Perceptron** in 1958 [19], and Bernard Widrow the **Adaptative Linear Element (ADALINE)** in 1960 [20]. The basic architecture of their neuron is shown in figure 2.2.

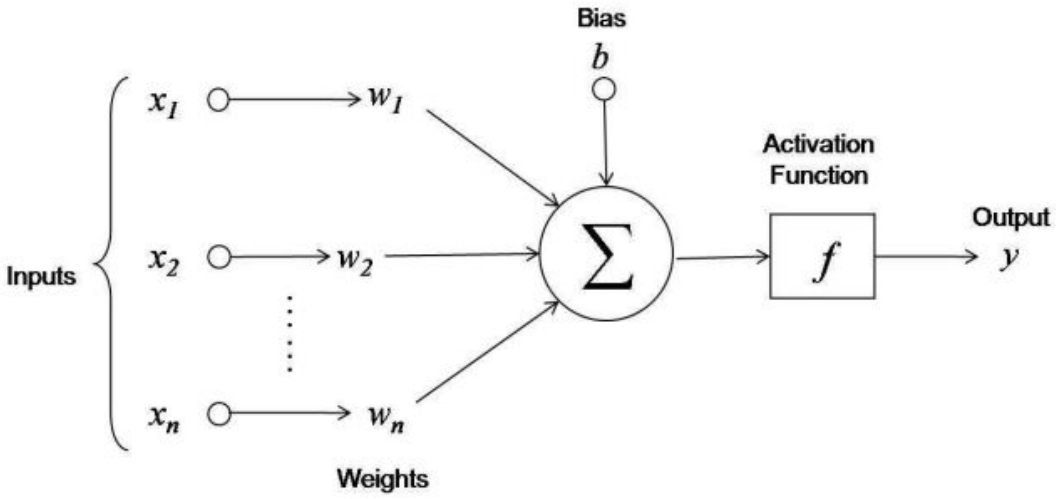


Fig. 2.2. Representation of a neuron

As we can observe, a neuron is essentially a mathematical function. It receives several inputs, and some weights that are associated to the connections between those inputs and the neuron. All these, together with a bias variable, are added up, in order to produce and output. The mathematical equation for the output is the following:

$$y = f(b + \sum_{i=1}^{n} x_i w_i) \tag{2.1}$$

The main difference between the two models is actually the activation function. The Perceptron uses a threshold function (2.2), while ADALINE uses the real sum of inputs (2.3). This is a big difference, as Perceptron can only be used for classification tasks, while ADALINE can also be used for linear regression problems.

$$f(x) = \begin{cases} 1 & \text{if} > 0 \\ -1 & \text{if} \leq 0 \end{cases} \tag{2.2}$$

$$f(x) = x \tag{2.3}$$

The objective of ADALINE is to minimize the error derived from the output. The error function specified for this neural network is the Mean Square Error (MSE):

$$E = \frac{1}{m}\sum_{p=1}^{m} E^p = \frac{1}{2m}\sum_{p=1}^{m}(d^p - y^p)^2 \tag{2.4}$$

where m is the total number of examples, d^p is the desired output of example p, and y^p is the real output of such pattern.

The learning process of ADALINE implies iteratively updating the weights of the connections between neurons in order to minimize the error function of the output. The learning rule used for such update is the Delta Rule, which tries to minimize the error of each individual example using a technique called **Gradient Descent**, summarized in equation 2.5.

$$w(t) = w(t-1) - \alpha\frac{\partial E^p}{\partial w} \tag{2.5}$$

Not entering into deeper mathematics, the idea of gradient descent method is to subtract the derivative of the error function. If we think of the error function as a n-dimensional surface, subtracting its derivative is equivalent to descending in that surface, until a minimum is found. As a result, we are reducing the error in each iteration. Figure 2.3 shows a representation of such logic in a 2-dimensional space, for a better understanding.

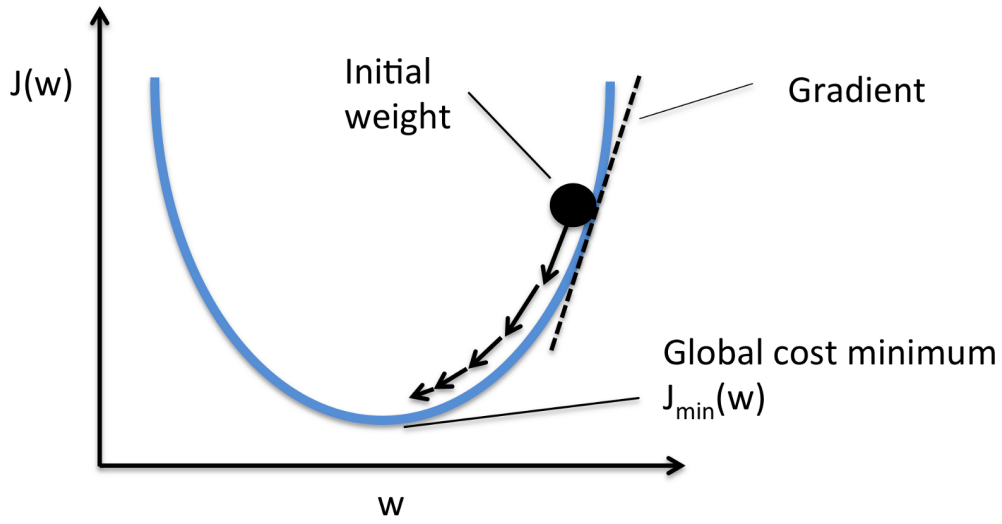


Fig. 2.3. Representation of how Gradient Descent works

These two neural network architectures became very popular for a few years and were the first indicator of what learning algorithms could achieve in the future. However, they had a huge drawback: they only were able to solve linear tasks. In the other hand, most of the problems that we find in the real world have a non-linear behaviour, for which current networks could not deal with.

In this context we arrive to the late 60s, in which the Artificial Intelligence field suffered from a huge depression in the research community. The lack of new advancements made many researchers think that eventually AI would only be a topic for science fiction, and investigation was paused by different groups. Artificial neural networks were knocked by a paper from M. Minsky and S. Papert, in which they proved that the Perceptron was not even able to learn an XOR function [21].

No progress was made until 1986, year in which Rumelhart et al. presented the **Generalized Delta Rule** [22]. Also named Backpropagation Rule, it consisted of updating the weights by distributing errors backwards throughout the network's layers. This allowed the creation of multiple layers of neurons, in which the output of one neuron could be connected to the input of another neuron, in order to build up a network that passes information from one neuron to another. All this idea was also based in gradient descent, and the objective was exactly the same: minimizing the error of the examples. However, with this new approach, non-linear problems were also solvable.

Together with backpropagation, a new ANN was introduced: the **Multilayer Perceptron (MLP)**. It consists of at least three layers of nodes: an input layer, a hidden layer and an output layer, but it may contain several hidden layers.

The most common activation function for the MLP is the sigmoid function, which is what inserts non-linearity into de algorithm. Its equation is the following:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

The architecture of the MLP is pictured in figure 2.4.

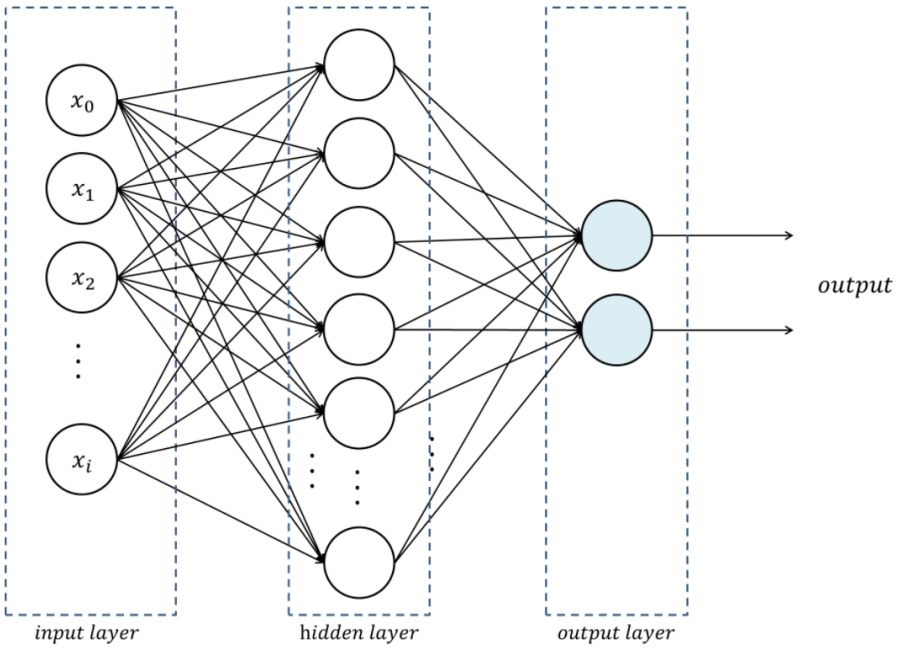


Fig. 2.4. Representation of a MLP [23]

Around the same time, it was shown that the MLP has the ability to approximate any function. For this reason, it became go to Machine Learning algorithm for every research lab. Unfortunately, this network was computationally expensive to use and, when more complex problems emerged, such as image recognition, MLP did not scale correctly. For this reason, they were eventually substituted by other learning algorithms, such as Support Vector Machines (SVM), leading to another cold period around the 2000s.

This second crisis was solved with the appearance of **Deep Learning**, main concept of this thesis, that will be explained in depth in the following section. Deep Learning is such important nowadays that recently the Association for Computer Machine (ACM) has awarded with the Turing Award to the main promoters of this revolutionary technology: Yoshua Bengio, Yann LeCun and Geoffrey E Hinton [24]. The Turing Award is considered to be the most prestigious honor in the field of Computer Science.

A brief summary of the history of artificial neural networks is shown in figure 2.5.

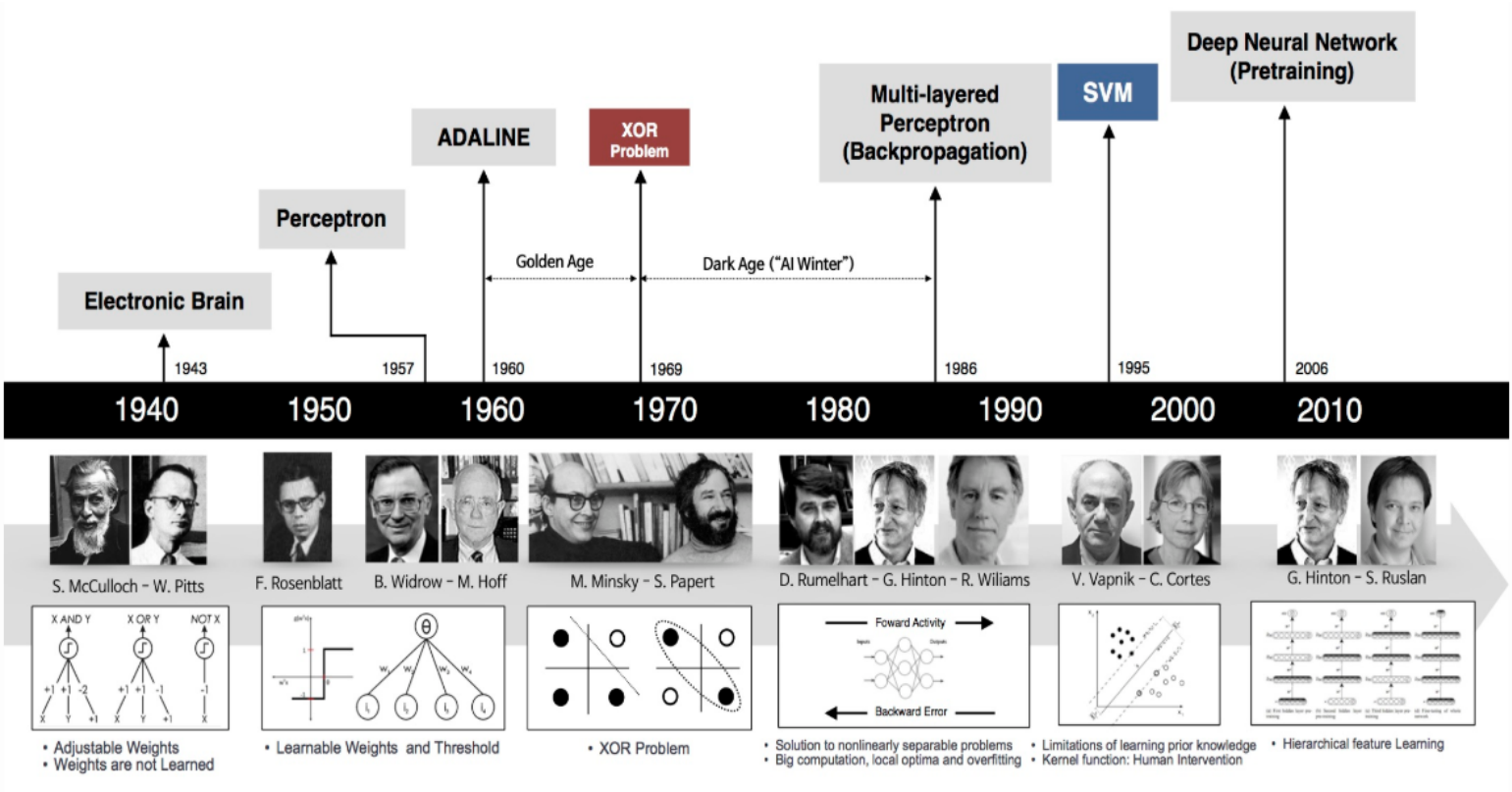


Fig. 2.5. Milestones in the Development of Neural Networks (Source: Andrew L. Beam [25])

2.1.3. Deep Learning

Deep Learning is a subset of the family of artificial neural networks algorithms, which consists of ANN with several hidden layers and neurons [26]. But, why is it important to add more hidden layers if one-hidden-layered networks work quite decently? There exist two ideas: on the theoretical side, it has been proven that some problems can be generalized better when increasing the number of hidden layers [27]. On the empirical

side, information processing in the brain works in a similar way, from the simpler ideas to the abstract concepts, and researchers believe in creating a mathematical model that emulates this.

Besides, as we previously studied, there existed an acceptable number of problems for which traditional neural network algorithms could not handle with successful results. Many of these issues were related to image processing, as the number of inputs (pixels) was so huge that parameters had been reduced to just a variety of image characteristics. This was not an effective approach anymore, and it was one of the main reasons why ANN lost popularity during the 2000s. However, with the appearance of Deep Neural Networks, the area enjoyed a new rise that continues in our days.

The first reference to the Deep Learning concept dates back to 2006, when Geoffrey E. Hinton introduced the idea of **Deep Belief Networks (DBN)** during his NIPS 2010 workshop: *Deep Learning and Unsupervised Feature Learning*. The idea was to use a composition of autoencoders, where each sub-network's hidden layer served as the visible layer for the next (see Figure 2.6). With this procedure, one could make an unsupervised pretraining of the parameters, and this would be used as an initialization of the parameters of a traditional neural network.

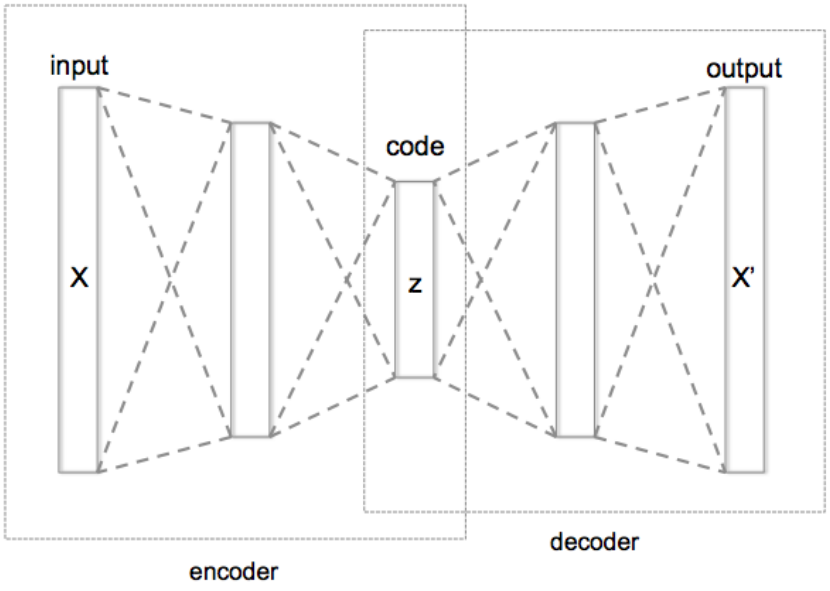


Fig. 2.6. Schema of an autoencoder (Author: Chervinskii [28])

After this first concept of DBN, interest was renewed and many neural networks research groups were publishing more and more papers on the area. However, the breakthrough did not arrive until 2012, when Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton, propose the **ImageNet** [29], a novel architecture that included three main improvements: the use of GPUs to train the model, the use of Rectified Linear Unit (ReLU) as

activation function, and the use of a method to reduce overfitting named dropout (see Figure 2.7) . From that moment, those innovations became mainstays in subsequent Deep Learning investigations.

ReLU function has the following equation:

$$f(x) = max(0, x) \tag{2.7}$$

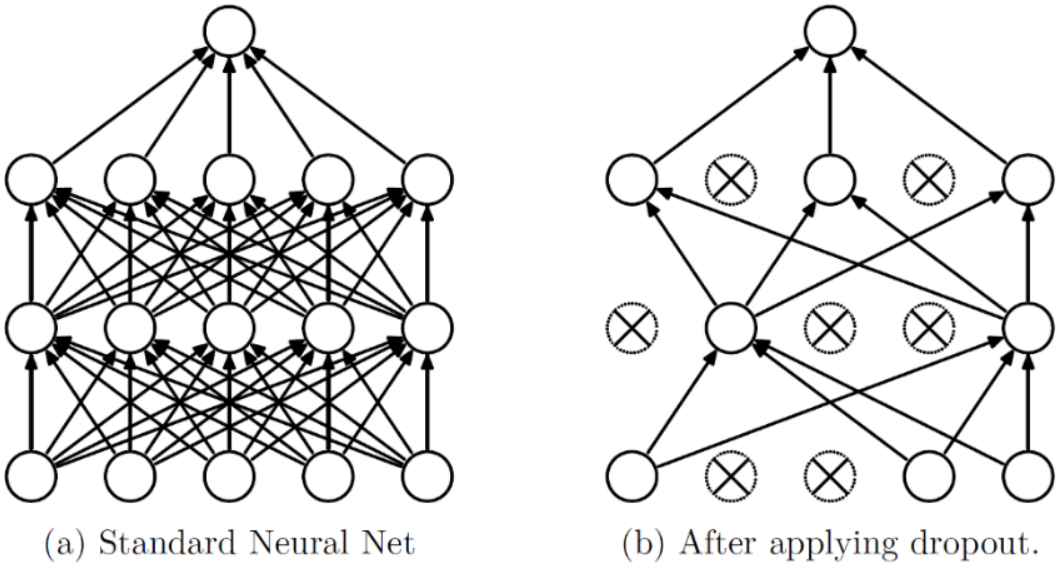


Fig. 2.7. Dropout method: with a probability p we remove a neuron from the network, both its incoming and outgoing connections. This helps to build more robust neurons, as they cannot always rely on receiving all the parameters. In the end, it helps to generalize better, preventing overfitting (Source: Srivastava, Hinton, Krizhevsky, Sutskever and Salakhutdinov [30]).

The most typical architecture used nowadays for Deep Learning is **convolutional neural networks**. They were first proposed in 1995 by Yann LeCun and Yoshua Bengio [31], but it was not until the previously commented improvements when they became a very effective learning algorithm.

CNNs unsupervised learning process consists of the combination of two types of layers: convolutional layers and pooling layers. Units in a convolutional layer are organized in feature maps, which will gather up the characteristics of each group of units. The reason why this is effective is that, in array data, such as images, local groups of values are often highly correlated, forming distinctive local motifs that are easily detected [32]. The role of the pooling layer is to merge semantically similar features into one. This is typically done by computing the maximum of a local patch of units in one feature map.

The result of this is that the dimension of the representation is reduced without losing important characteristics from the feature maps.

A CNN consists then of a series of stacked convolutional and pooling layers that are finally connected to a regular ANN (see Figure 2.8). Backpropagation through a CNN is as easy as through a regular neural network, making the learning process simple.

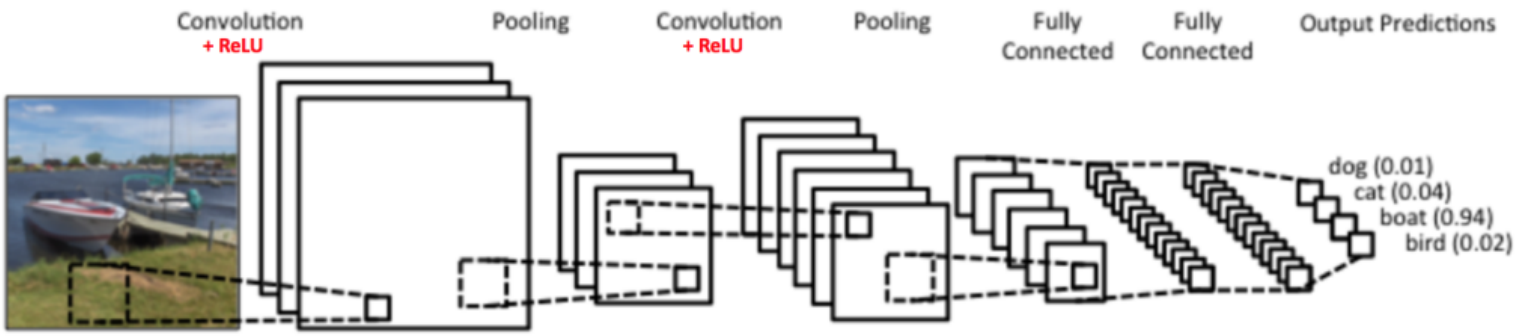


Fig. 2.8. Architecture of a Convolutional Neural Network (Credits to Clarifai [33])

The most direct beneficiary from convolutional neural networks has been image recognition. There are tens of applications for this technology in a wide variety of subjects, going from computer vision in videogames to segmentation of biological images [34]. Two very interesting examples of projects made with CNN include the following:

- **Self-driving cars AI platform**. Nobody doubts that self-driving cars is the future of automotive industry, and many companies are racing to be the first to build a fully automatic vehicle for the economic revenue that the achievement will mean. Nvidia is one of these companies, but they are going one step ahead: they have created a platform that helps other manufacturers make autonomous driving a reality. They have achieved this combining its excellent hardware (Nvidia's main industry is the design of GPUs) with a software that uses Deep Learning to identify elements in the road. Their solution has such potential that even Toyota, the world's largest vehicle manufacturer, has partnered with Nvidia for this venture [35].

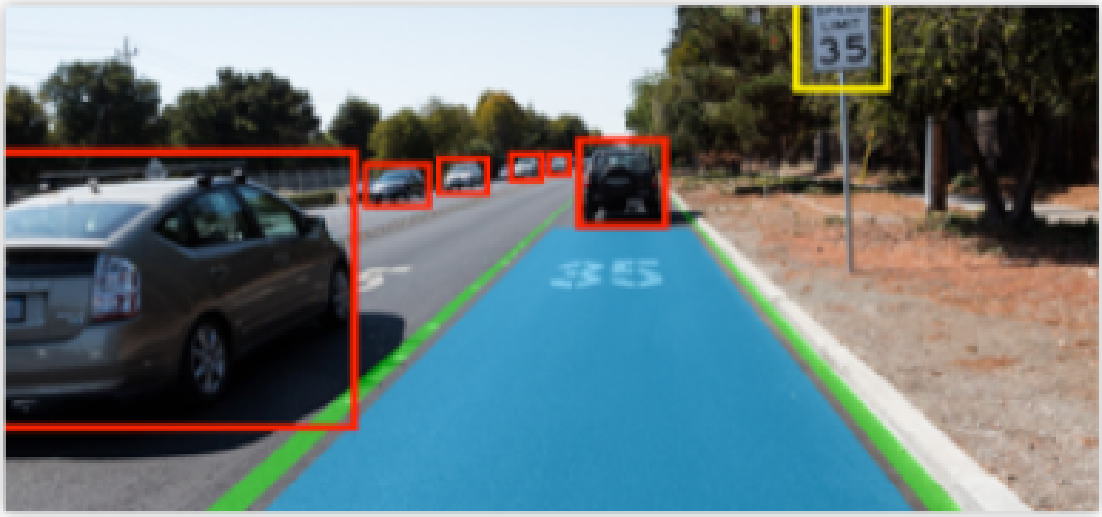


Fig. 2.9. Prototype of how a self-driving car will identify other objects in the road (Credits to NVidia [36])

- **Person Blocker**. Remember TV series Black Mirror's episode *White Christmas*, in which a new technology able to block people from your life was presented? Data scientist Max Woolf brought this vision to reality with the help of convolutional neural networks [37]. This software receives an image and it will produce the blocking effect on the elements specified by the user (see Figure 2.10). With the Person Blocker you cannot only block people, but also a wide variety of objects: animals, food, vehicles, or other domestic utensils. It is a very nice example of how accurate image recognition is.

Fig. 2.10. Person Blocker example (Author: Max Woolf [38]).

Besides image recognition related solutions, Deep Learning has also become a fundamental pillar in areas such as Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU). These two subjects are closely related with the system that we are going to build for this thesis, so we will make a brief overview of them in the following sections.

2.2. Conversational Interfaces

Conversational Interfaces are the main focus of this thesis. As a result, a whole section is given to study the history of the field, as well as the evolution of each of its main components (automatic speech recognizer, natural language understanding module, and dialog manager) through the work done by preceding authors.

2.2.1. Introduction

Since the 1980s, the idea of a conversational interface with which humans could interact for different purposes has been a focus of study for a wide variety of research groups and communities, that largely worked to make this technology come to reality, and not just an utopia of contemporary science fiction films and books, such as Star Wars or Blade Runner. Up to this point, conversational interfaces had taken the form of either text-based dialog systems, where the main emphasis was on applying techniques from linguistics, or chatbots, that simulated conversation using simple pattern-matching techniques.

Major advances in automatic speech recognition technology made it possible to recognize user's spoken input with a reasonably high degree of accuracy [39]. This led to the development of **Spoken Dialog Systems (SDS)**, which tried to go one step ahead in the simulation of "intelligence" that previous dialog systems had. Areas of Artificial Intelligence such as user modelling [40] and planning [41] were being used in order to deal with problematic aspects of communication and recognize intentions behind user's statements. The results were ATIS [42] and SUNDIAL [43], considered to be the first SDS projects.

On the other hand, we have **Voice-User Interfaces (VUI)**, which began around the 1990s with the realization that speech could be used to automate self-service tasks, such as call routing, directory assistance and information enquiries. Although they used the same spoken language technologies as SDS, VUI development was oriented towards business needs, such as return on investment, usability and user satisfaction. This technology was early adopted by telecommunications companies such as AT&T [44][45] to address the task of call routing, and it is still widely used in our days.

The last precursor that we are going to talk about are chatbots. The main aim of these systems is to fool the user into thinking that they are conversing with another human. However, the interaction with chatbots is not oriented towards task solving as in SDSs and VUIs, but to generate small conversations. As a result, chatbots do not usually take

the initiative in conversations and they are limited to react to user's input. Chatbots have their origins in ELIZA [46], which simulates a Rogerian psychotherapist, and has been used until our days in areas such as education, e-commerce and information retrieval.

Some of the key technologies that made conversational interfaces a reality are the following:

- **Artificial Intelligence**: Since Alan Turing first proposed in 1950 a kind of system capable of learning [47], researchers have worked towards creating computers capable of demonstrating intelligent behaviour. At first, it was thought that intelligent behaviour could be extracted based on rules, and knowledge-based systems achieved spectacular results such as chess playing computer Deep Blue, which defeated the human world champion in 1996 [48]. However, other aspects of intelligent learning, such as image recognition or speech recognition, could not be solved using these methods and required inference-based knowledge in order to extract appropriate patterns in data. The field got stuck for a while until GPUs started to be used for computing purposes, allowing process vast amounts of data (*Big Data*). New algorithms like Deep Learning also contributed to improve the efficacy of AI systems.

- **Language technologies**: language understanding has enormously benefited from the newer Machine Learning advancements. Besides, the field has such an interest for big tech companies like Google, Apple, Amazon or Facebook, that they have promoted the research in various areas of speech and language technology, including speech recognition, text-to-speech synthesis, spoken dialog management, and natural language learning.

- **Semantic Web**: in 2001, Berners-Lee and colleagues [49] put forward a vision for a Web in which all the content will be structured and machine-readable, so that different agents could easily perform tasks such as checking calendars, making appointments, and finding locations. They called it Semantic Web. This research has enabled to better interpret the semantics of a user's query and to return structured responses to that query.

- **Device technologies**: smartphones have incorporated such powerful processors that they have the same computing power as large personal computers. This, combined with appearance of cloud computing, has enabled that resource-intensive operations such as speech recognition can be handled faster. Furthermore, smartphones' access to user's location, contacts or calendar, has enabled the increasing of customization of virtual personal assistants for each user.

Current conversational interfaces are made up of some basic components, represented in Figure 2.11. Those are the following:

- **Automatic speech recognition**: obtains the sequence of words uttered by a speaker. It is actually a very complex task, and has been a matter of research (see [50][51]) in order to overcome the issues of linguistics, transmission channel or interaction context.
- **Spoken Language Understanding (SLU)**: obtains the semantics from the recognized sentence. It involves morpholigical, lexical and syntactical analysis [52].
- **Dialog Management (DM)**: decides the next action of the system, interpreting the incoming semantic representation of the user input in the context of the dialog [53].
- **Natural Language Generation (NLG)**: usually considered part of the dialog manager, obtains sentences in natural language from internal representation of information handled by the dialog system [54].
- **Text-To-Speech Synthesis (TTS)**: transforms the generated sentences into synthesized speech [55].

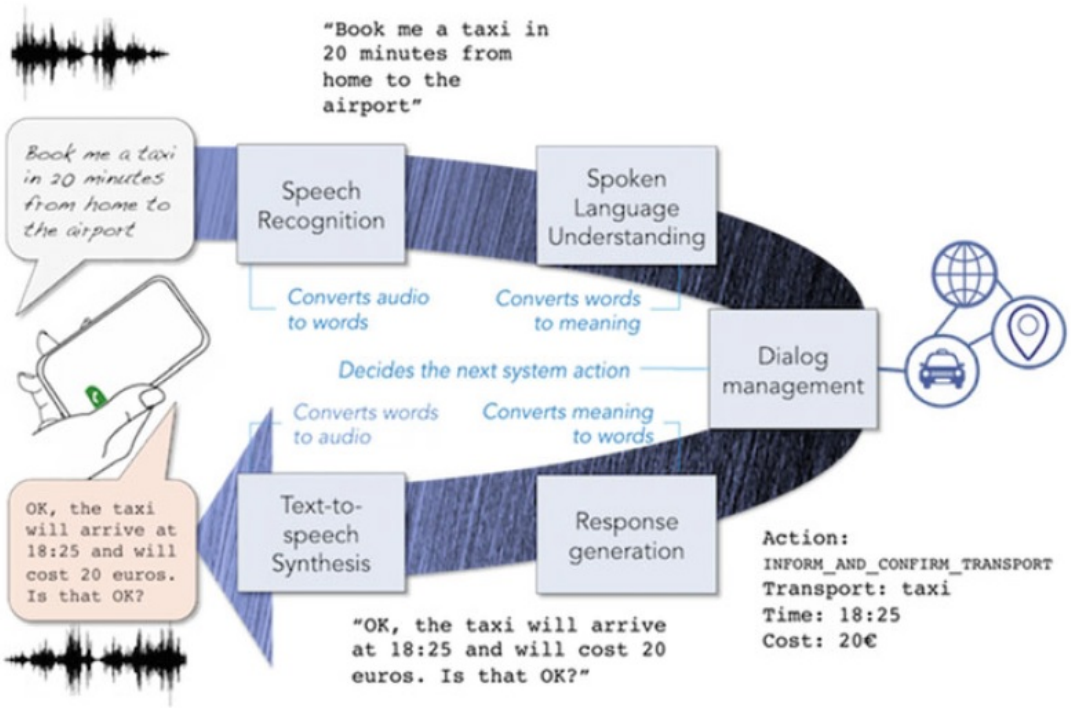


Fig. 2.11. The components of a spoken language conversational interface [2]

2.2.2. Automatic Speech Recognition

Speech recognition is the discipline that develops technologies to automatically recognize spoken language and translate that spoken input into text. It is the first main component that we find in a conversational interface, as the first interaction with the user is listening to what the user is saying to later understand it.

This task has an enormous complexity and requires a great amount of training, as the number of variables affecting the speech recognition is huge. A speech recognizer should know how to deal with differences in tone, volume or pitch; with possible speech disorders such as stuttering; or with environment sound and noise. It also requires an in-depth research in linguistics, in order to understand the sentence structure and to correct possible mistakes from the user.

In its early stages, speech recognition was an exercise of pattern recognition based on templates and the spectral distance of the sound that was recorded. The first system able to recognize speech was developed at Bell Laboratories in 1952. Scientists S. Balashek, R. Biddulph, and K. H. Davis, built **'Audrey'** [56], which was able to recognize numerical digits. In the following years, improvements to that system were made, including arithmetic operations, vowels and consonants recognition.

The U.S. government centered its interest in ASR, probably due to the advantage that they would obtain for their intelligence operations. As a result, U.S. Department of Defense's invested in the technology, driving considerable improvements in the field. The most remarkable system was Carnegie Mellon University's **Harpy**, which could recognize just over 1,000 words by 1976 [57].

In the mid 80s, the field was revolutionized with the popularization of **Hidden Markov Models (HMMs)**. This approach represented a shift towards a statistical method for speech processing, and meant a great improvement of accuracy. By the end of the decade, there were various companies competing for the best ASR system, and the 90s introduced consumers to speech recognition with the Dragon, a system that could identify over 80,000 words.

By the 2000s, speech recognition was still dominated by traditional approaches such as HMMs and ANNs, and funding was mainly coming from U.S. government initiatives. However, that changed soon, when companies with vast financial resources such as Microsoft, Apple and Google foresaw the potential of the technology in the future. Soon, they began to hire experts in the field to develop their own ASR systems.

Most recently, the field has benefited from advances in the Deep Learning area. The appearance of a DL method called **Long Short-Term Memory (LSTM)** marked the beginning of the ASR systems that we know today. Besides, there is a increasing interest from companies and institutions in speech recognition research, for all the applications that it has. Some examples of recent publications include Deep Learning applied to Audio-Visual Speech Recognition [58] or Distributed Deep Learning strategies applied to ASR [59].

2.2.3. Natural Language Understanding

Natural language understanding is the discipline that develops technologies to deal with machine reading comprehension. It is the conversational interface component that comes

after ASR, and its job is to use the context of the conversation and the ASR input to match the sentence with a previously learned series of options. These options, which are specific of each problem and represent the different intentions of the user, are called intents, and are the main concept of NLU systems. As a result, the component has an ontology around the product that is used to calculate the probability of some intent.

One of the earliest softwares developed for natural-language understanding dates from 1964. Daniel Brobow wrote the program **STUDENT** [60], which could solve simple algebra word problems. It was the first demonstration of how a computer could understand natural language input.

However, the greatest advancement came a year later with the creation of **ELIZA** [46] which, as previously mentioned, simulated a Rogerian psychotherapist. The system used simple parsing, grammar rules and substitution of keywords to understand phrases. It is the precursor of current commercial chatbots.

After several approaches of how to represent natural language input, in 1971 T. Winograd presented **SHRDLU** [61], a system aimed to direct a robotic arm to move objects. The system could only understand very simple English structures and had a very restricted vocabulary, but worked quite well, and served as a major influence for research undertaken at the time.

In the following years, various attempts for commercial NLU systems were made in from different authors. Some examples of those are Jabberwacky, a friendly and entertaining chatbot proposed with the objective of passing the Turing Test [62], and BORIS, an experimental story understanding and question answering system [63].

Nonetheless, the greatest advancements went hand in hand with the successes achieved with Machine Learning in the early 2000s. As a result, a new approach was taken, using Machine Learning for text classification. One of the most renowned examples is IBM's **Watson**, which is a question answering software powered by ML. In 2011, it participated in the famous American TV show Jeopardy! against the all-time best players, beating them [64]. In a more serious context, it is also used as a guidance in healthcare decisions, such as lung cancer treatment in the Memorial Sloan Kettering Cancer Center in New York City [65].

Lately, the appearance of **LSTMs** helped to improve language modeling and to refine the NLU systems. Currently these Deep Learning architectures are the ones that produce best results for natural-language understanding. There is a very high interest in the area due to its multiple applications, including automated reasoning, question answering, and large-scale content analysis. Hu et al. [66] proposed several CNN for the NLU task, and Narasimhan et al. [67] used Deep Learning for language understanding in videogames.

2.2.4. Dialog Management

The dialog manager is the main object of study in this thesis. It corresponds to the third process in the cycle of conversational interfaces, and it is in charge of analysing the information given by the NLU module (user intent, sentence keywords) in order to make a response. Therefore, we will revise the literature on the topic to understand previous work and current concerns on the field.

One widespread methodology to implement dialog management systems is applying **Reinforcement Learning**. The reason for this is that the main issue current learning approaches to dialog management face is related to the vast space of possible dialog states and strategies. As we know, Machine Learning algorithms learn from trial and error, so they need a sufficiently big corpus of data to achieve a valid training. Getting such information from dialogs between real users and the system is unfeasible, due to the high volume of resources needed to do so (money and time). To solve this issue, some researchers proposed to train a statistical user model that will simulate real user responses. This could then be used to learn the dialog strategy between interactions with the simulated user [68]. As a result, a large number of investigations are being carried out on user modelling with statistical approaches, and it is a very common technique for the development of dialog managers.

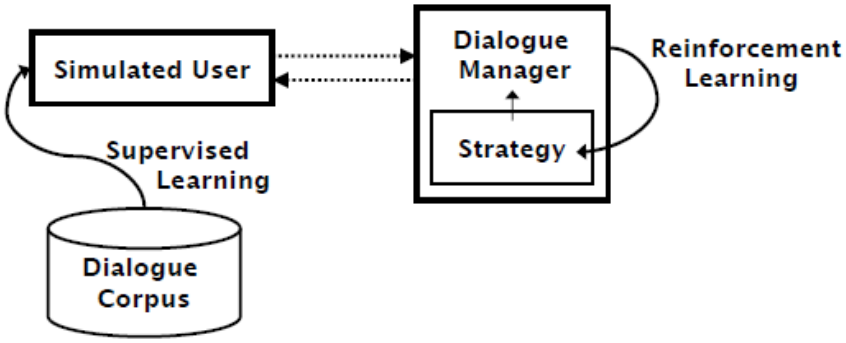


Fig. 2.12. Learning Dialog Strategies with a Simulated User (Source: [69])

The process will work as illustrated in Figure 2.12. First, the user model is trained with the real dialog corpus in order to learn what to respond to a given dialog state. During the second phase, the simulated user is used to interact with the dialog manager so that it optimizes dialog strategy based on the feedback given. This allows automatic training with simulated dialogs, and it even enables the exploration of dialog strategies that were not present in the original dialog corpus.

Early models for dialog strategies learning were implemented with **Markov Decision Processes (MDPs)**. MDPs allow forward planning and hence were used as statistical framework for dialog policy optimization by several authors, such Levin et al [70]. However, MDPs assume that the entire space of states is observable, and it does not work when

uncertainty is introduced in the system.

As a result, S. Young et al. proposed a new dialog system based on **Partially Observable Markov Decision Processes (POMDPs)**, which could account for uncertainty [71]. The idea is to retain a full and rich state representation but only maintain probability estimates over the most likely states. Results were really powerful, concluding that POMDPs build more robust spoken dialog systems.

In recent years, new proposals have appeared using what is called **Reinforcement Deep Learning (RDL)**. This technology merges Deep Learning with Reinforcement Learning algorithms in order to create more efficient solutions to current problems. A small amount of publications have reported good results [72][73], but it is not a standardized technique yet.

In 2013, Microsoft created the **Dialog State Tracking Challenge (DSTC)** [74]. It consists of a series of dialog state tracking tasks, designed for the research community so that they can investigate their new models and methodologies within the dialog management field. Each year, they publish a corpus of over 15k human-computer labelled dialogs, which is a dataset very difficult to build individually, in order to enhance research on the area.

Besides RL techniques, other authors have researched on alternative learning algorithms for dialog management generation. D. Griol et al. evaluated up to six different classifier: multinomial naive Bayes, n-grams, decision trees, support vector machines, grammatical inference and artificial neural networks [75][76][77]. Best results were obtained using the MLP, and a highly accurate dialog management system was developed using this ANN.

There is, however, very little reference in the literature of dialog management systems based on deep neural networks. For this reason, this thesis aim is to experiment with different configurations of Deep Learning architectures in the search of improvements from other ANN approaches. As a result, this work is considered to be a novel research in the area, and it is meant to serve as a reference for future research.

2.3. Tools Analysis

There exist a wide variety of tools and frameworks to develop Machine Learning algorithms, perform models analysis, and create spoken dialog systems. As a result, an overview of the most relevant ones has been made, arguing their advantages and drawbacks so as to conclude which is the best option. Note that not all the available tools are going to be listed, but only the ones that we have been considered and evaluated.

2.3.1. Tools for Machine Learning

Before getting into the Deep Learning analysis, the first part of the project involves performing supervised learning in the corpora, so that valuable comparisons can be made between traditional models and the proposed ones. Although it is not difficult to manually implement the basic supervised learning algorithms, we have decided to use a platform in order to save time in this task, as it is not the main one for the research.

Weka

Weka is a suite of Machine Learning software developed at the University of Waikato, New Zealand [78]. It contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions. Weka supports several standard tasks, such as data preprocessing, clustering, classification, regression, visualization, and feature selection.

The main advantages that found to use this software were:

- Model portability, as you can download any model and export it to any other platform.
- Contains an extensive collection of data preprocessing and modeling techniques.
- No learning curve. You can implement your first models in minutes.
- Simplicity in the analysis, thanks to its graphical user interfaces.
- Contains a broad community of users and documentation to reference to.
- Contains a wide variety of add-on tools that increases empowers the platform.

On the other hand, Weka has two important disadvantages:

- Consumes a large amount of resources, making it unfeasible for large datasets.
- Complex algorithms can take days to execute.

BigML

BigML is a company that provides a selection of robust Machine Learning algorithms to solve real-world problems [79]. It includes supervised learning algorithms, such as trees or neural networks, and unsupervised ones, like clustering, anomaly detection or association discovery.

BigML main advantages include the following:

- It runs on large prepared servers, so it executes very fast.

- Contains a wide collection of modeling techniques.
- Very easy to interpret results, several plot-charts and analysis are shown after every prediction.

However, it has great disadvantages that led to dismiss the platform:

- One does not really control the parameters of the different algorithms to try out. The platform assumes little knowledge on the area, and does much of the configuration automatically, so it is hard to understand what you are trying out.
- The platform does not come free of charge. Although there is a trial version, it does not contain full potential of the Machine Learning software.
- It does not have an extended documentation or a community of users in the internet to solve your doubts.

RapidMiner

RapidMiner is a commercial data science software platform that provides an integrated environment for data preparation, Machine Learning, text mining, and predictive analytics [80]. It has more than 500 operators oriented towards data analysis, including information preprocessing and visualization. It contains very similar characteristics to Weka software, so we could put them in the same level. However, RapidMiner offers some upgrades:

- RapidMiner offers the vast majority of Weka algorithms for Machine Learning, and it provides the necessary tools to integrate Weka models in their platform. As a result, we could define RapidMiner as an evolution of Weka, with Data Science methodologies.
- The community of users around the platform is growing faster than their competitors, with all that this implies: larger and better documentation, more add-on tools are developed for RapidMiner, and more authors include the software in their researches.

On the other hand, it may be a bit more difficult to learn the foundations of the platform. As a result, it requires time to get used to the user interface, and to search for the necessary operators to achieve the desired results. Besides, interpretation of output is a bit more complex yet it provides more complete insights.

TensorFlow

TensorFlow is the most popular open-source library for Machine Learning applications [81]. It was developed by the Google Brain team, and it is actively used for research

and production in Google, sign of the tremendous power of the framework. It has stable APIs for Python and C languages, but it has lately released versions for Java, JavaScript or Swift.

The main advantages of using TensorFlow are:

- It focuses on the design of neural networks, providing a broad variety of layer types with many configurable parameters.
- It has a great computational power even for complex networks or large datasets.
- It has a vast community of developers and companies supporting it, with a well defined documentation and forums to solve any issue concerning the framework.
- It has a very nice network visualization tool integrated.

Nonetheless, this amount of power and control for the developer comes with a series of drawbacks:

- Greater curve of learning. Indeed, it is hard to first understand what is happening or how to program the network. Also, such large configuration set is a bit confusing at the beginning.
- Preprocessing has to be done in an earlier stage by the user. This feature is not incorporated in the framework, so the scientist must manually treat the data or use another software before passing it to TensorFlow.
- It does not include advanced analysis tools. To give a very clear example, confusion matrices are not included, and must be programmed separately. That increases the level of uncertainty and the development time.

Based on the former analysis, it has been decided to use the following tools: Weka and TensorFlow. Although RapidMiner is a more complete software than Weka, the extra functionalities that it provides are related to data science and are not really needed for this project. As a result, authors' prior experience in several projects with Weka makes them feel more comfortable, accounting that results will be very similar, if not the same. As Weka has difficulty when it comes with large datasets, the power of TensorFlow will be used in those situations. Regarding TensorFlow, despite its harsh curve of learning, it will also be used for Deep Learning as it is explained below, so it does not involve an extra time to use the library for simpler neural networks too.

2.3.2. Tools for Deep Learning

After the first simple models, Deep Neural Networks will be created to evaluate the suitability of such algorithm to dialog manager implementations. As a result, specialized

tools are needed to implement Deep Learning models. The main ones scrutinized are the following:

Keras

Keras is an open-source neural-network library written in Python [82]. It focuses on being user-friendly, modular, and extensible. This is achieved by offering a more intuitive set of abstractions that make it easy to develop DL models regardless of the computational backend used.

As a result, Keras is more of a high-level interface than a proper tool for complex Deep Learning analysis, where it becomes more limited. Therefore, we will study if any of the studied frameworks provides support for Keras.

TensorFlow

Besides simple neural networks, TensorFlow is capable of building any Deep Learning based neural network, thanks to its simple layering system. As a result, once you know how to use the library, it is very fast to implement more complex networks and test the results.

The biggest drawback for TensorFlow has to do with programming the exact desired algorithm, because it can become really complicated to write the code for a specific configuration. For this reason Google incorporated Keras in the library, making it extremely simple to adjust to the developer necessities while keeping the powerful TensorFlow backend.

PyTorch

PyTorch is an open-source Machine Learning library for Python, with a high-level interface for deep neural networks [83]. It is supported by top tech companies like Facebook, and it is a very popular tool for natural language processing.

PyTorch's main advantage is that it is a dynamic computational graph, that is, the configuration and architecture of the network can be adjusted in every iteration. This provides a great flexibility to implement fairly complex designs. Besides, PyTorch merged with the popular framework Caffe2, hence supporting complex deep neural network architectures such as CNNs, recurrent CNNs or LSTMs.

When it comes to compare PyTorch with TensorFlow, there are slight differences. On the one hand, building ML models with PyTorch building more intuitive and straightforward. On the other hand, it is a relatively less known framework. it has a smaller community, not many integrations and less papers referencing PyTorch implementations. Moreover, with TensorFlow you can visualize your ML models directly, whereas for PyTorch you need an external tool.

As anticipated before, it has been decided to use the following framework: TensorFlow + Keras. Although PyTorch seems to have a less steep curve of learning, we consider that TensorFlow saves this drawback by incorporating Keras interface. In addition to this, PyTorch's dynamic modeling feature is not necessary for this project, and TensorFlow offers a much powerful backend. Besides, documentation and community is larger for TensorFlow, fact that was also valued for the decision.

2.3.3. Tools for SDS implementation

Once the dialog management system has been modelled, we will have to integrate it with the other components to form a spoken dialog system that we can test on real users. There are some platforms that facilitate this task by providing pre-built components.

Nearly ten leading tools for SDS implementation were evaluated, including: Chatfuel, the leading chatbot platform for Facebook Messenger; Twilio, a commercial platform that enables real-time AI SMS text, email, WhatsApp or chat; and Business Chat, Apple's bot framework for conversational interfaces in their devices. However, the ones assessed as more complete were the following:

DialogFlow

DialogFlow is the human-computer interaction technology developed by Google to create natural language conversations [84]. It sets the foundation for some of the Google products, such as Google Assistant or Google Home.

DialogFlow provides a framework to implement conversational interfaces really easily. Both the speech-to-text and text-to-speech modules are implemented using their technology. The NLU component can be easily implemented, as the developer only needs to define the different possible inputs for each user intent, and their pre-trained algorithms will easily identify the intent. The dialog manager can also be implemented with predefined answers to each of the intents, but it also provides a method to add your own trained system.

DialogFlow also supports more than 14 languages (including the main spoken ones), can be integrated in wearables, cars, smart devices, web applications or any other mobile application, and has a considerable community supporting the product.

Amazon Lex

Amazon Lex is a service for building conversational interfaces into any application using voice and text, developed by Amazon [85]. It is the system powering Alexa, Amazon's VPA.

As with DialogFlow, Amazon Lex has already integrated the ASR and TTS modules,

utilizing Amazon's Deep Learning technologies, and the NLU component uses the same intent identification idea. This allows us to focus on designing the dialog manager response. Besides, it supports easy deployment of the chatbot into mobile applications and other messaging services such as Facebook Messenger or Slack.

A very nice feature is that responses for the CUI can be defined directly from the Amazon Web Services (AWS) console, allowing a very simple integration of the dialog management system into the SDS.

The two downsides found for this technology is that there is no free version (only trial version), and that the community is not as extended as with DialogFlow.

Microsoft LUIS

Acronym for Language Understanding Intelligent Service, LUIS is Microsoft's Machine Learning-based service to build natural language into apps, bots, and IoT devices [86]. It is used by big multinational companies such as UPS or the actual Microsoft team, for their internal and external interactions.

As well as its competitors, it allows the developer to define user intents in any of the top 15 spoken languages, and it has a pre-defined Machine Learning algorithm that predicts very accurately the user response. It also identifies entities, and other context variables.

There is, however, a huge drawback: LUIS only contains the NLU component of a Dialog System, so we would need to integrate it with other frameworks to provide the full spoken conversational experience. For that reason, this alternative was dismissed.

After considering these options, the conclusion was that Amazon Lex and DialogFlow offered very similar solutions, and both could be used. Finally, it was decided to use DialogFlow for the following reasons:

- It has a free version of the product for non-commercial solutions.
- It has a greater community and more extended documentation, facilitating the development of the CUI.
- We are also using other Google's solution for the Deep Learning model prototyping, TensorFlow, so we predict that we will have less issues integrating the model with DialogFlow. Besides, we will probably use Firebase (another Google product) to host the model in a server, and there is plenty of documentation on how to work with DialogFlow and Firebase together.

3. DEEP LEARNING ANALYSIS APPLIED TO CUI DIALOG MANAGER

This section discusses the work done to generate a Deep Learning model for a conversational interface dialog manager implementation. As we anticipated before, the main objective of this part of the thesis is to provide some reference work for future research, as there is very little reference of Deep Learning applied to dialog management models.

For this reason, traditional Machine Learning models (neural networks, decision trees, bayesian networks and k-nearest neighbors) will be compared with several Deep Learning architectures, to study if there is any improvement in the accuracy of the model. Weka will be used to analyse the traditional Machine Learning results, while TensorFlow is accounted for the Deep Learning task.

3.1. Analysis for pizza ordering domain

The first corpus that was evaluated is based in the pizza ordering domain, in which users would ask the conversational interface about their requests and orders, and the system will process the order. It was extracted from M. McTear et al.'s book [2], and it is an extremely simple corpus that was used a first contact with the analysis tools and models.

The corpus consists of 968 training examples labelled with each of the actions that the conversational interface should retrieve to the inputs. Those are the following:

1. Opening.
2. Ask the type of order.
3. Ask the number of pizzas.
4. Ask the types of pizzas.
5. Ask the sizes of pizzas.
6. Ask the types of dough.
7. Ask the drinks.
8. Ask to confirm the type of order.
9. Ask to confirm the number of pizzas.
10. Ask to confirm the types of pizza.
11. Ask to confirm the sizes of pizzas.
12. Ask to confirm the types of dough.
13. Ask to confirm the drinks.
14. Closing.

In order to predict each action, the examples have 10 input attributes:

- PREV_SYSTEM_ACTION, which stores the previous action taken by the system, as a context.
- Six task-dependent attributes, which will be codified with the values 0, 1, 2 according to the following criteria: 0, if the value of the slot is unknown; 1, if the value is known with a high confidence; and 2, if the value is known with a low confidence and needs confirmation from the user. These attributes are:
    1. Type of order.
    2. Number of pizzas.
    3. Types of pizzas.
    4. Sizes of pizzas.
    5. Types of pizza dough.
    6. Drinks.
- Three task-independent attributes, which will be codified with the same criteria as before, and will provide important information to build a more complete system. The slots are:
    1. Acceptance: if the user has confirmed some information.
    2. Rejection: if the user has denied some information.
    3. Not-understood: if the system has not identified the user's input.

An example of interaction is shown in figure 3.1:

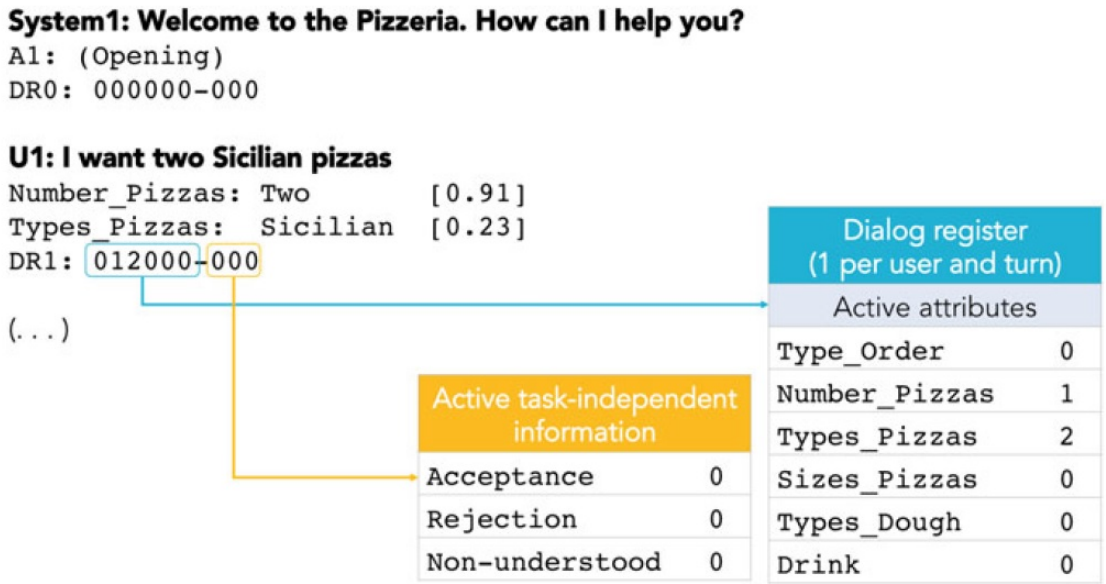


Fig. 3.1. Example of interaction between the user and the CUI within the pizza ordering domain (Ref: [2])

As it can observed, after the welcoming action, the user asks for two Sicilian pizzas. The number of pizzas is identified with high confidence, so the value is stored and the slot changes its value to 1. However, the pizza type is not identified with high confidence, so the corresponding slot gets a value of 2. As no more information is provided, the rest of the task-dependent attributes keep the value of 0. On the other hand, the task-independent information does not variate, as the user is not providing either confirmation or denial. Therefore, the value that will be passed to the model will be:

$$Opening - 012000 - 000$$

The corpus given was already cleaned and formatted as a *.csv* file, so no preprocessing was needed. This was one of the main reasons why this corpus was chosen, to start the analysis in Weka and TensorFlow right away. However, one data transformation was applied:

- Training instances order was shuffled. This is relevant because it avoids biases during the learning process, especially when the dataset is built with a particular sequence.

Consequently, let's describe the different algorithms that have been used for such analysis.

Machine Learning traditional algorithms

After experimenting with all the Weka algorithm set, the following models have been chosen for the final analysis:

- **M.W1:** Bayesian Network, with K2 search algorithm and a simple estimator.
- **M.W2:** Linear logistic regression.
- **M.W3:** K-nearest neighbors algorithm, with k=1.
- **M.W4:** Logistic Model Tree (LMT).

We have also built some multilayer perceptron networks with TensorFlow, each of them with a increasing number of hidden layers. The main purpose of this is to investigate the behaviour of the model when increasing the number of hidden layers, to check if the accuracy improves or downgrades. As a result, five more models are presented:

Model	M.T1	M.T2	M.T3	M.T4	M.T5
Input Layer	dense (256)	dense (256)	dense (256)	dense (256)	dense (256)
Hidden Layers		dense (256)	dense (256) dense (256)	dense (256) dense (256) dense (256)	dense (256) dense (256) dense (256) dense (256)
Output Layer	dense (13)	dense (13)	dense (13)	dense (13)	dense (13)

Table 3.1. PIZZA DOMAIN TENSORFLOW MULTILAYER PERCEPTRON MODELS

The dense layer is just a regular connected ANN layer, as was represented in figure 2.2. Therefore, its output equation will be described by the dot product between the weight and input vector adding the bias term, as shown in equation 2.1. The resulting network is then a MLP.

All these models were trained with the *adam* optimizer, which proved to be the most effective option for this domain, and *sparse_categorical_crossentropy* loss function. This loss function just indicates that the target labels are represented as integers.

Deep Learning algorithms

The design comprises a total of 6 different convolutional neural network architectures, with different complexity, so as to have a complete analysis of the capabilities of this algorithm. Some of the architectures were referenced from other that proved to work very good in well-known datasets, such as Vasudevan's CIFAR-10 classifier [87]. Other architectures were chosen for what has worked considerably good in previous CNN research done by the author.

Table 3.2 shows the different architectures chosen. Please note that, as studied in the State of the Art chapter, CNNs consist of a series of stacked convolutional and pooling layers that are finally connected to a MLP. For simplicity purposes, the perceptron part will be omitted. Dropout layers have also been omitted.

Model	M.T6	M.T7	M.T8	M.T9	M.T10	M.T11
	Conv (16, kernel=2)	Conv (64, kernel=3)	Conv (32, kernel=3)	Conv (32, kernel=3)	Conv (32, kernel=3)	Conv (40, kernel=5)
	Conv (32, kernel=3)					
	Pooling (pool=2)			Pooling (pool=2)	Conv (32, kernel=3)	
	Conv (64, kernel=4)					Pooling (pool=2)
	Pooling (pool=2)	Conv (32, kernel=3)	Pooling (pool=2)	Conv (64, kernel=3)	Pooling (pool=2)	
	Conv (128, kernel=4)					Conv (70, kernel=3)
	Pooling (pool=2)			Pooling (pool=2)	Conv (64, kernel=3)	
MLP	MLP	MLP	MLP	MLP	MLP	MLP

Table 3.2. PIZZA DOMAIN TENSORFLOW CNN MODELS

All these models were also trained using the *adam* optimizer and *sparse_categorical_crossentropy* loss function.

As the data corpus is small and we do not want to miss any valuable training examples, we have decided to evaluate the model using cross validation with 5 folds. This means that the model is trained with the full data corpus but, in order to provide relevant evaluation metrics, the corpus is divided in 5 partitions. After this, the corpus is trained during 5 iterations, taking 4 of the partitions as training set, and leaving the other one for testing. The results after all the iterations are then averaged, giving the final metric. Figure 3.2 displays a representation of how a 5-fold cross validation would work.

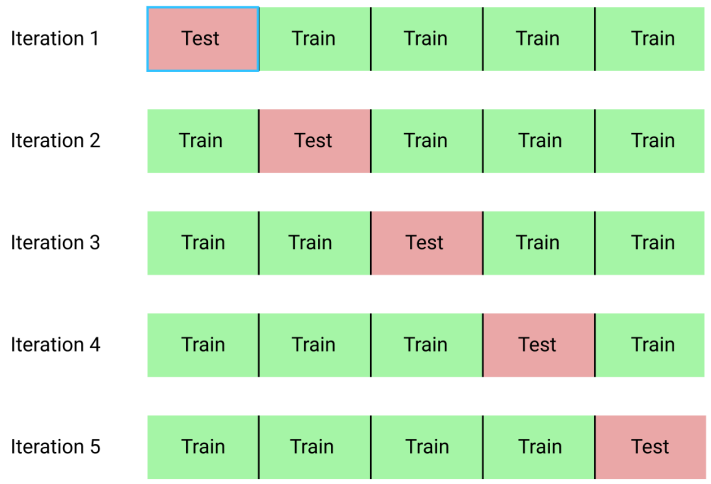


Fig. 3.2. Representation of a 5-fold cross validation

Results

After running the algorithms, these are the results obtained for each of the models:

Traditional Models	Options	Accuracy
M.W1	search=K2; estimator=simple	97.93%
M.W2	heuristicStop=50; maxBoostingIt=500	**99.02%**
M.W3	k=1, linearNNSearch	98.03%
M.W4	minNumInstances=15; numBoostingIt=-1	99.00%
M.T1	epochs=40; optimizer=adam; α=0.001	98.86%
M.T2	epochs=40; optimizer=adam; α=0.001	98.34%
M.T3	epochs=40; optimizer=adam; α=0.001	98.34%
M.T4	epochs=40; optimizer=adam; α=0.001	98.55%
M.T5	epochs=40; optimizer=adam; α=0.001	97.72%

Table 3.3. PIZZA DOMAIN TRADITIONAL MODELS RESULTS

Having a look to the table, it can concluded that the models have very similar results, ranging from 97.72 to 99.02 percent. Therefore, we could say that all of these models will deliver a very acceptable outcome for the studied domain.

The best results are obtained by the **linear logistic regression** model, which achieves an accuracy of **99.02%**. However, as it has been explained, performance is very similar with other algorithms such as LMT or MLP.

As can be seen, the best MLP result is achieved with the simplest model, M.T1, which has no hidden layers. And the accuracy tendency when increasing the number of hidden layers is downwards. This is insightful discovery because, although theoretically the larger a network it is the better approximation function it would make, the truth is that many times simpler is actually better. In the case of the pizza domain, which is not very complicated, results are favorable to smaller networks.

Coding process included the programming of several representational diagrams in order to facilitate interpretation of results. As a result, taking as a reference model M.T1, plots are shown for its accuracy evolution and confusion matrix of some of its folds.

Accuracy evolution per number of epoch is presented in figures 3.3, 3.4 and 3.5.

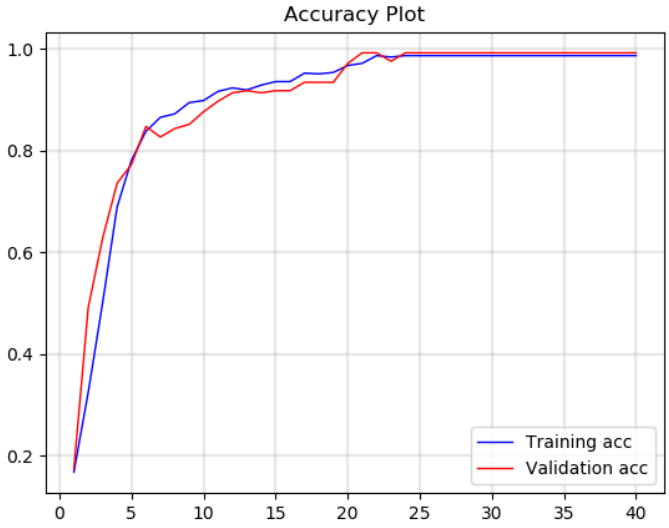


Fig. 3.3. Pizza Domain model M.T1 accuracy evolution (Fold 1)

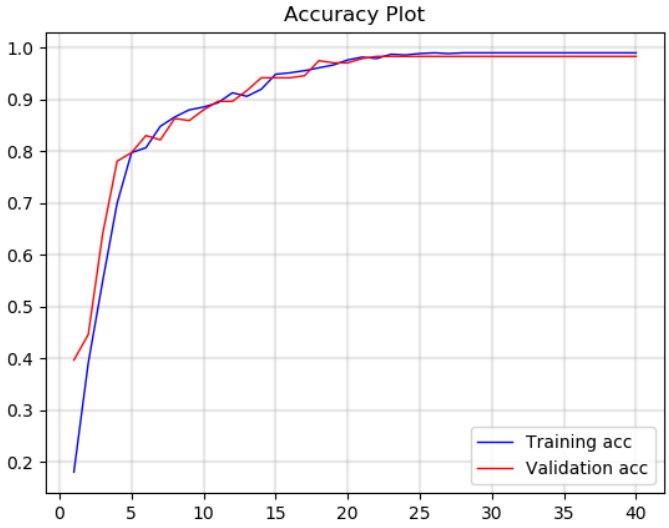


Fig. 3.4. Pizza Domain model M.T1 accuracy evolution (Fold 2)

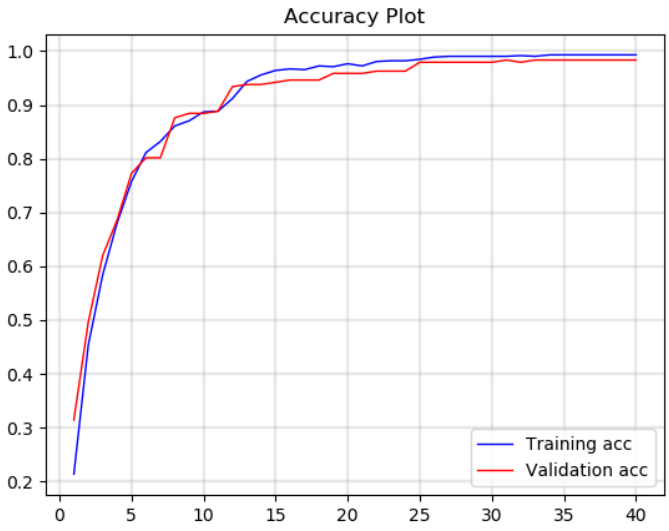


Fig. 3.5. Pizza Domain model M.T1 accuracy evolution (Fold 3)

As can be observed, training evolution is gradual and it converges at around epoch 35. Overfitting is not found, as the validation accuracy rises proportionally to the training accuracy, and it does not decrease at any moment, showing good generalization. Another relevant detail is that results between folds behave extremely similar, with very few variations in the final precision obtained (at most, 0.8 percentage points). Therefore, a successful learning process can be confirmed.

Tables 3.4 and 3.5 represent the confusion matrices made for folds 2 and 3, respectively. For portrayal reasons, only the numeric value of the class will be plotted. The numbering ordered is the same as the one specified when previously identifying each of

the labels. For instance, number 1 does not appear (Opening state), number 2 will correspond to asking the type of order, and so on.

Accuracies for each of the classes are very well condensed, with no class apparently being excessively weak in the predictions. Therefore, it can concluded that errors found are due to small outliers in the data corpus, and there is not an specific class with a high misclassification rate.

		Predicted Label												
		2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
	2	**14**	0	0	0	0	0	0	0	0	0	0	0	0
T	**3**	0	**10**	0	0	0	0	0	0	0	0	0	0	0
r	**4**	0	0	**17**	1	0	0	0	0	0	0	0	0	0
u	**5**	0	0	0	**21**	0	0	0	0	0	0	0	0	0
e	**6**	0	0	0	0	**34**	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	**48**	0	0	0	0	0	0	0
L	**8**	0	0	0	0	0	0	**6**	0	0	0	0	0	0
a	**9**	0	0	0	0	0	0	0	**3**	0	0	0	0	0
b	**10**	1	0	0	0	0	0	0	0	**11**	0	0	0	0
e	**11**	0	0	0	0	1	0	0	0	0	**6**	0	0	0
l	**12**	0	0	0	0	0	0	0	0	0	0	**3**	0	0
	13	0	0	0	0	0	0	0	0	0	0	0	**14**	0
	14	0	0	0	0	0	0	0	0	0	0	0	1	**51**

Table 3.4. PIZZA DOMAIN MODEL M.T1 CONFUSION MATRIX (FOLD 2)

		Predicted Label												
		2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
	2	**14**	0	0	0	0	0	0	0	0	0	0	0	0
T	**3**	0	**9**	0	0	0	0	0	0	0	0	0	0	0
r	**4**	0	0	**23**	0	0	0	0	0	0	0	0	0	0
u	**5**	0	0	0	**31**	0	0	0	0	0	0	0	0	0
e	**6**	0	0	0	0	**49**	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	**33**	0	0	0	0	0	0	0
L	**8**	0	1	0	0	0	0	**5**	0	0	0	0	0	0
a	**9**	0	0	0	0	1	0	0	**8**	0	0	0	0	0
b	**10**	0	0	0	0	0	0	0	1	**4**	0	0	0	0
e	**11**	0	0	0	0	0	0	0	0	0	**3**	0	0	0
l	**12**	0	0	0	0	0	0	0	0	0	0	**5**	0	1
	13	0	0	0	0	0	0	0	0	0	0	0	**13**	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	**41**

Table 3.5. PIZZA DOMAIN MODEL M.T1 CONFUSION MATRIX (FOLD 3)

Results for the designed CNN models are presented in table 3.6.

CNN Models	Options	Accuracy
M.T6	epochs=40; optimizer=adam; α=0.001	98.26%
M.T7	epochs=40; optimizer=adam; α=0.001	**98.96%**
M.T8	epochs=40; optimizer=adam; α=0.001	98.74%
M.T9	epochs=40; optimizer=adam; α=0.001	96.79%
M.T10	epochs=40; optimizer=adam; α=0.001	98.55%
M.T11	epochs=40; optimizer=adam; α=0.001	98.85%

Table 3.6. PIZZA DOMAIN CNN MODELS RESULTS

The results obtained are very acceptable, ranging from 96.79 to 98.96 percent. However, there is a significant conclusion to make: these models are **not improving traditional architectures results**.

The best results are obtained by model **M.T7**, which achieves an accuracy of **98.96%**. Although we could argue that this model is equivalent to the best traditional model, M.W2, by no means we can claim that the CNN is offering better results. As a result, a Deep Learning solution is not suitable for this domain.

An important comment to make is regarding the complexity of the network. As it was previously anticipated with the MLP results, a more complex network has no correlation with a better approximation of the result. This appreciation appears again when dealing with CNNs: while the networks (M.T7, M.T8, M.T11) are achieving the highest performances, architectures with a larger number of convolutional and pooling layers are decreasing the performance.

Taking as a reference model M.T7, plots are shown for its accuracy evolution and confusion matrix.

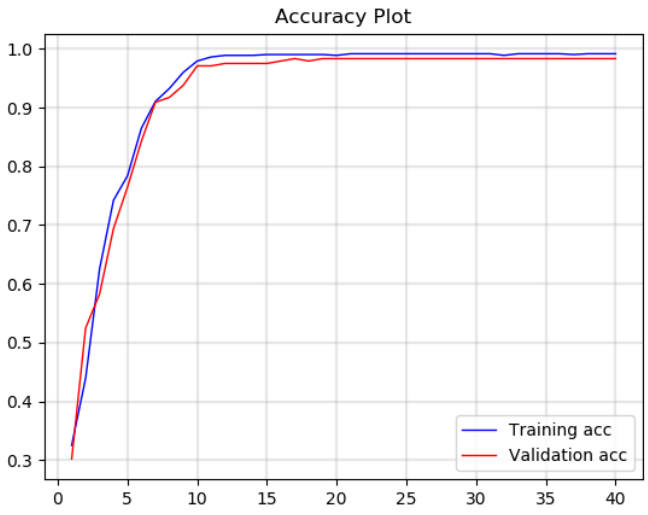


Fig. 3.6. Pizza Domain model M.T7 accuracy evolution (Fold 1)

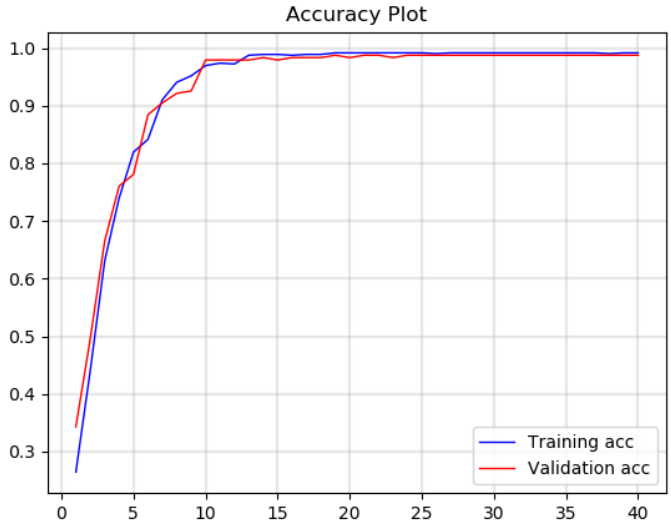


Fig. 3.7. Pizza Domain model M.T7 accuracy evolution (Fold 4)

Training process with convolutional networks is faster than with traditional methods, achieving a convergence at around epoch 15. Good generalization is also shown, with no sign of overfitting in the model. Therefore, a satisfactory learning can be affirmed.

Table 3.7 represents the confusion matrix made for fold 1. As with model M.T1, there is no sign of a high misclassification rate in any specific class. As a result, the Deep Learning model is completely capable of making accurate predictions in this domain, with very few error probability for any of the system scenarios.

		Predicted Label												
		2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
	2	**18**	0	0	0	1	0	0	0	0	0	0	0	0
T	**3**	0	**10**	0	0	0	0	0	0	0	0	0	0	0
r	**4**	0	0	**22**	0	0	0	0	0	0	0	0	0	0
u	**5**	0	0	0	**21**	0	0	0	0	0	0	0	0	0
e	**6**	0	0	0	0	**44**	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	**39**	0	0	0	0	0	0	0
L	**8**	0	2	0	0	0	0	**9**	0	0	0	0	0	0
a	**9**	0	0	0	0	1	0	0	**7**	0	0	0	0	0
b	**10**	0	0	0	0	0	0	0	0	**9**	0	0	0	0
e	**11**	0	0	0	0	0	0	0	0	0	**6**	0	0	0
l	**12**	0	0	0	0	0	0	0	0	0	0	**3**	0	0
	13	0	0	0	0	0	0	0	0	0	0	0	**17**	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	**33**

Table 3.7. PIZZA DOMAIN MODEL M.T7 CONFUSION MATRIX (FOLD 1)

CNNs have not brought any improvement from traditional ML algorithms. It is also true that a domain that generalizes so well with simple architectures can be hard to refine. Hence, more complex domains will be tried as they could provide more insights of the validity of Deep Learning for this type of problems.

As a last resource, one more experiment is going to be performed, building an ensemble. An ensemble is basically a group of N trained networks, in which results are combined to produce an overall result. Theoretically, the combination of several networks should give better results, as the weaknesses of one network can be compensated by the others.

In this case, a majority-vote ensemble has been crafted. It takes the output labels for every network, and it returns the one that has appeared the most, that is, the mode. An example is shown in the figure 3.8.

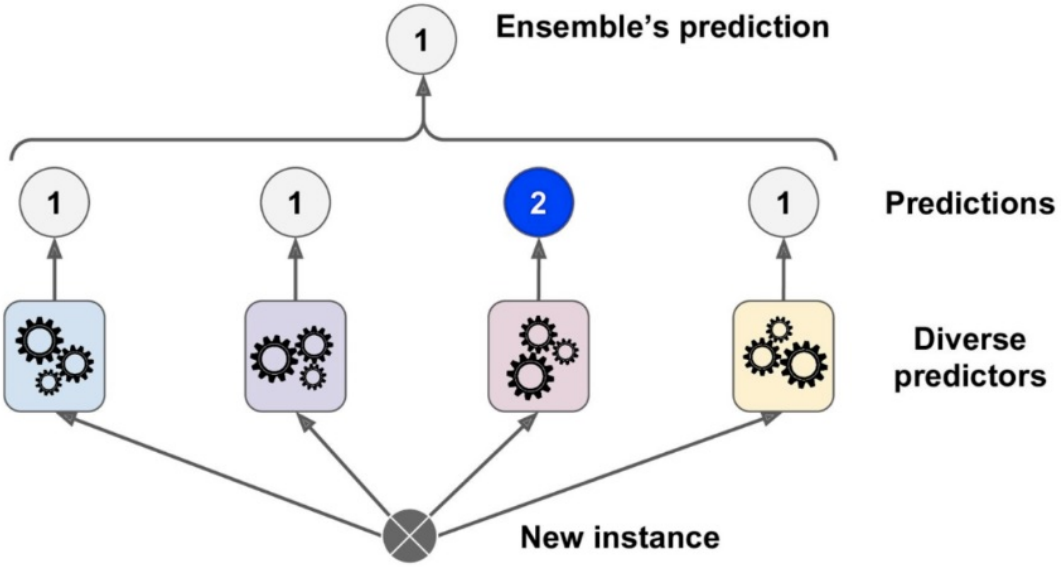


Fig. 3.8. Representation of a majority-vote ensemble (Source: [88])

For the pizza domain case, a total of six networks were chosen, combining two MLP and four CNN architectures: M.T1, M.T5, M.T6, M.T7, M.T10 and M.T11. Results obtained by the ensemble are shown in table 3.8.

Ensemble	Options	Accuracy
M.Ensemble	epochs=40; optimizer=adam; α=0.001	99.05%

Table 3.8. PIZZA DOMAIN ENSEMBLE MODEL RESULTS

Results show a rise in accuracy when combining all the models than for each model individually. Therefore, building an ensemble is recommended. However, the improvement is not significant, confined only to few training examples. As a result, it can be

confirmed that the percent point that any algorithm is capable of modelling corresponds to a small number of outliers in the dataset.

Finally, the confusion matrix for ensemble's fold 3 is shown in table 3.9, which achieves almost a perfect classification rate.

		Predicted Label												
		2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
	2	**9**	0	0	0	0	0	0	0	0	0	0	0	0
T	**3**	0	**6**	0	0	0	0	0	0	0	0	0	0	0
r	**4**	0	0	**22**	0	0	0	0	0	0	0	0	0	0
u	**5**	0	0	0	**23**	0	0	0	0	0	0	0	0	0
e	**6**	0	0	0	0	**31**	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	**49**	0	0	0	0	0	0	0
L	**8**	0	1	0	0	0	0	**4**	0	0	0	0	0	0
a	**9**	0	0	0	0	0	0	0	**6**	0	0	0	0	0
b	**10**	0	0	0	0	0	0	0	0	**8**	0	0	0	0
e	**11**	0	0	0	0	0	0	0	0	0	**3**	0	0	0
l	**12**	0	0	0	0	0	0	0	0	0	0	**5**	0	0
	13	0	0	0	0	0	0	0	0	0	0	0	**14**	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	**61**

Table 3.9. PIZZA DOMAIN MODEL M.ENSEMBLE CONFUSION MATRIX (FOLD 3)

3.2. Analysis for train scheduling domain

The second corpus that was evaluated is based in the train scheduling domain, in which users would ask the conversational interface about information regarding various train routes (e.g. origin and destination, departure time, price...), and the system would process the petition and retrieve the required information. This corpus is a cession from Dr. Griol Barres that was used as corpus for the development of his PhD. thesis [89]. In such publication, the author experiments with different ML algorithms, but Deep Learning proposals are not presented, so this work will try to complete such investigation.

The corpus consists of 4006 training instances, each of the labelled with the possible actions that the conversational interface could retrieve to each input. Those are the following:

1. Opening.
2. Ask the destination city of the train.
3. Ask the departure date of the train.
4. Ask to confirm the departure date of the train.

5. Retrieve answer for the train schedule.
6. Closing.
7. Ask if the request is for train schedules.
8. Ask to confirm the destination city of the train.
9. Retrieve answer for the train price.
10. Ask to confirm the origin city of the train.
11. Ask to confirm the type of train.
12. Retrieve answer for the train type.
13. Ask to confirm the departure hour of the train.
14. Ask to confirm the destination city and departure hour of the train.
15. Ask to confirm the origin city and price of the train.
16. Ask to confirm the type of ticket class for the train passenger.
17. Ask to confirm the arrival hour of the train.
18. Ask if the request is for train types.
19. Retrieve answer for the train services.
20. Ask if the request is for train prices.
21. Retrieve answer for the train travel time.
22. Ask to confirm the departure date and hour of the train.
23. Ask to confirm the origin city and departure hour of the train.

The codification applied for this corpus is based on the one authors followed in [77], which is very similar to the one for the pizza domain. In order to keep an active dialog state, it is not relevant to know the exact piece of information that the user has given, but only if the type of information that has been mentioned during the dialog. As a result, attributes will be codified with the values 0, 1, 2 according to the following criteria: 0, if the concept is unknown or value has not been given yet; 1, if the concept is known with a confidence higher than a given threshold; and 2, if the concept is known with a confidence lower than a given threshold.

In order to predict each action, the examples have 19 input attributes:

- prev_response, which stores the previous action taken by the system, as a context.
- Five task-dependent attributes, which will denote the type of request the user is asking for. These attributes are:
    1. Ask for timetable.
    2. Ask for price.
    3. Ask for train type.
    4. Ask for order number.
    5. Ask for services list.

- Ten task-dependent attributes, which will denote the type of information that has been mentioned in the dialog. These attributes are:

    1. Origin city.
    2. Destination city.
    3. Departure date.
    4. Arrival date.
    5. Departure hour.
    6. Arrival hour.
    7. Ticket class.
    8. Train type.
    9. Order number.
    10. Services list.

- Three task-independent attributes, which will be codified with the same criteria as before, and will provide important information to build a more complete system. The slots are:

    1. Acceptance: if the user has confirmed some information.
    2. Rejection: if the user has denied some information.
    3. Not-understood: if the system has not identified the user's input.

An example of interaction is shown in Figure 3.9:

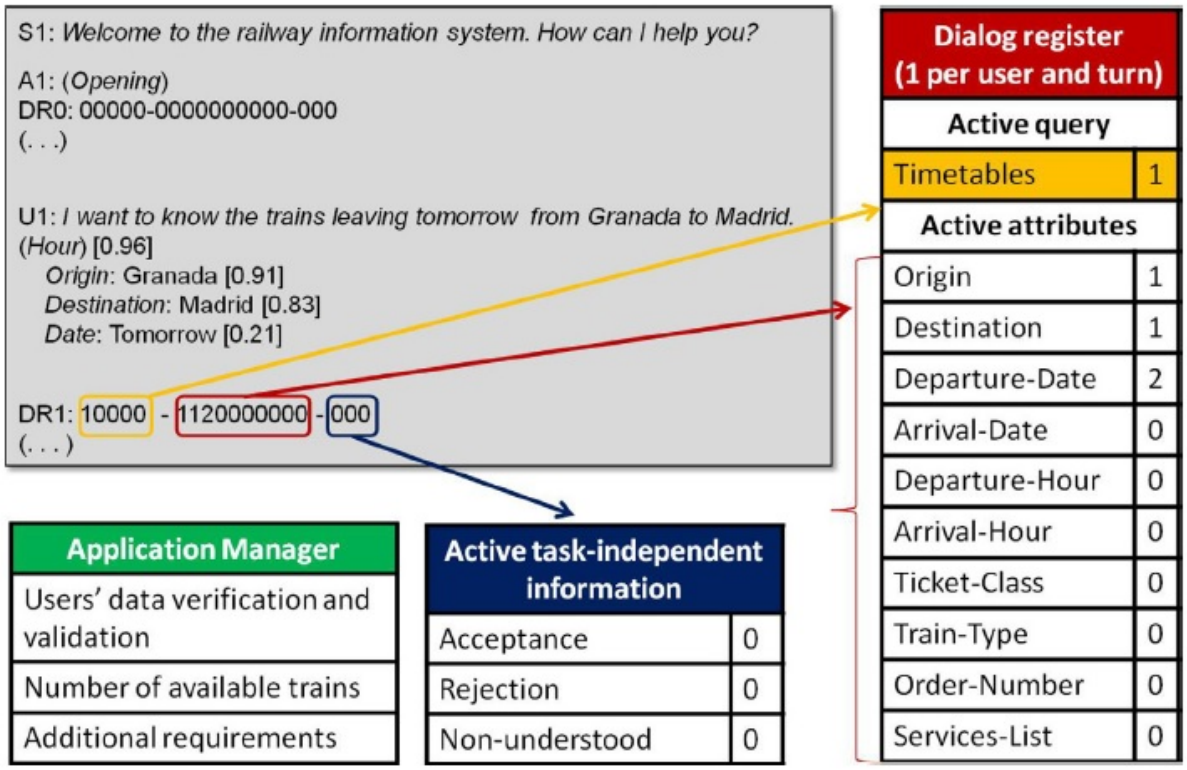


Fig. 3.9. Example of interaction between the user and the CUI within the train scheduling domain (Source: [77])

As can be observed, after the opening action, the user asks for the trains leaving tomorrow from Granada to Madrid. Both the origin and destination cities are identified with a high level of confidence, changing the value in their respective slot to 1. However, the departure date is identified with a very low level of confidence, so it will be stored with a value of 2. The type of request is also identified as a train schedule query, so its corresponding slot gets a value of 1. On the other hand, the task-independent information does not variate, as the user is not providing either confirmation or denial. Therefore, the value that will be passed to the model will be:

$$Opening - 10000 - 1120000000 - 000$$

The corpus was originally given as a *.txt* file, consisting of separated dialogs, each of them containing a list of interaction turns with the following structure:

$$<prev\_response>-<inf\_mentioned>-<type\_query>-<indep\_attr>-<response>,$$

where:

- *prev_response* is a string denoting the previous system response,
- *inf_mentioned* is a string of 10 digits denoting the type of information that has been mentioned in the dialog,
- *type_query* is a string of 5 digits indicating the type of request the user is asking for,
- *indep_attr* is a string of 3 digits representing the acceptance, rejection and misunderstandings states, as explained before, and

- *response* is a string denoting the system response.

There were originally 28 types of responses. Some of them were representing the same dialog state, and others had at most 1 instance representing them. As such, the corpus was cleaned, merging similar states and omitting underrepresented exceptional dialog situations. The resulting label set was the one explained above.

After this, all the labels were transformed into integer numbers. The main reason for this is that TensorFlow does not operate with non-integer entities, such as strings, so the parsing was necessary. The corpus was also shuffled because, as it has been previously noted, it prevents any possible biased inferred from the data order. Finally, the file was formatted as a *.csv* for its correct parsing in Weka and TensorFlow.

As it can be noticed, this domain has an added difficulty compared to the previously analyzed problem. The corpus is four times larger, while the input attribute set is twice as extensive. Therefore, it is expected to be a more difficult corpus to approximate, where Deep Learning can show its potential.

Let's now describe the different algorithms used for the analysis in the train scheduling domain.

Machine Learning traditional algorithms

For the Weka analysis, we experimented with the same algorithms as for the pizza set, as they proved to be the most flawless ones:

- **M.W1:** Bayesian Network, with K2 search algorithm and a simple estimator.
- **M.W2:** Linear logistic regression.
- **M.W3:** K-nearest neighbor algorithm, with k=1.
- **M.W4:** Logistic model tree.

TensorFlow MLP modelling was also very similar. A progressive process was followed, in which models with an increasing number of layers were designed. The main objective pursued by this approach is to see the differences when increasing the complexity of the MLP. Hence, five models were built:

Model	M.T1	M.T2	M.T3	M.T4	M.T5
Input Layer	dense (256)	dense (256)	dense (256)	dense (256)	dense (256)
Hidden Layers		dense (256)	dense (256) dense (256)	dense (256) dense (256) dense (256)	dense (256) dense (256) dense (256) dense (256)
Output Layer	dense (13)	dense (13)	dense (13)	dense (13)	dense (13)

Table 3.10. TRAIN DOMAIN TENSORFLOW MULTILAYER PERCEPTRON MODELS

All these models were trained with the *adam* optimizer, which proved to be the most effective option for this domain, and *sparse_categorical_crossentropy* loss function.

Deep Learning algorithms

We have designed a total of 6 different convolutional neural networks architectures, with different complexity, so as to have a complete analysis of the capabilities of this algorithm. Some of them are very similar to the ones built for the pizza ordering domain (thus, extracted from [87] and previous experimentation), while others have incorporated and added complexity in the form of extra layers.

Table 3.11 shows the different architectures chosen. For simplicity purposes, Dropout and MLP layers have been omitted.

Model	M.T6	M.T7	M.T8	M.T9	M.T10	M.T11
	Conv (16, kernel=2)	Conv (64, kernel=3)	Conv (32, kernel=3)	Conv (32, kernel=3)	Conv (32, kernel=3)	Conv (40, kernel=5)
	Conv (32, kernel=3) Pooling (pool=2) Conv (64, kernel=4)			Pooling (pool=2) Conv (64, kernel=3)	Conv (32, kernel=3) Pooling (pool=2) Conv (64, kernel=3)	Pooling (pool=2) Conv (70, kernel=3) Conv (500, kernel=3)
	Pooling (pool=2)	Conv (32, kernel=3)	Pooling (pool=2)	Pooling (pool=2)		Pooling (pool=2)
	Conv (128, kernel=4) Pooling (pool=2)			Conv (128, kernel=3)	Conv (128, kernel=3) Pooling (pool=2)	Conv (1024, kernel=3) Pooling (pool=2)
MLP	MLP	MLP	MLP	MLP	MLP	MLP

Table 3.11. TRAIN DOMAIN TENSORFLOW CNN MODELS

All these models were also trained using the *adam* optimizer and *sparse_categorical_crossentropy* loss function.

The model evaluation methodology chosen is cross validation with 5 folds. As with the pizza ordering domain, we do not want to miss any valuable piece of information from the dataset, so the final model will be built from the complete corpus, but the evaluation metrics will be obtained from the average of the 5 folds results.

Results

After running the algorithms, these are the results obtained for each of the models:

Traditional Models	Options	Accuracy
M.W1	search=K2; estimator=simple	90.06%
M.W2	heuristicStop=50; maxBoostingIt=500	92.26%
M.W3	k=1, linearNNSearch	89.59%
M.W4	minNumInstances=15; numBoostingIt=-1	91.24%
M.T1	epochs=150; optimizer=adam; α=0.0005	**92.34**%
M.T2	epochs=150; optimizer=adam; α=0.0005	91.79%
M.T3	epochs=150; optimizer=adam; α=0.0005	91.39%
M.T4	epochs=150; optimizer=adam; α=0.0005	91.46%
M.T5	epochs=150; optimizer=adam; α=0.0005	91.14%

Table 3.12. TRAIN DOMAIN TRADITIONAL MODELS RESULTS

In this domain, results have a greater variance, ranging from 89.59 to 92.44 percent. While some estimators, such as Bayesian networks or k-nearest networks, are proved to perform more poorly, neural networks and regression clearly achieve better outcomes.

The best results are obtained by the first **Perceptron** architecture, which has no hidden layers. It achieves an accuracy of **92.34%**. Following with the pizza domain hypothesis, we also find that simpler networks are producing better approximation functions than the more complex architectures. As a result, it could be argued than this domain is also a bit easy to approximate and that CNNs may not be necessary.

Model M.T1 has been taken as a representative as it has the highest accuracy. Plots will be shown for its accuracy evolution and confusion matrix of some of its folds.

Accuracy evolution per number of epoch is the following:

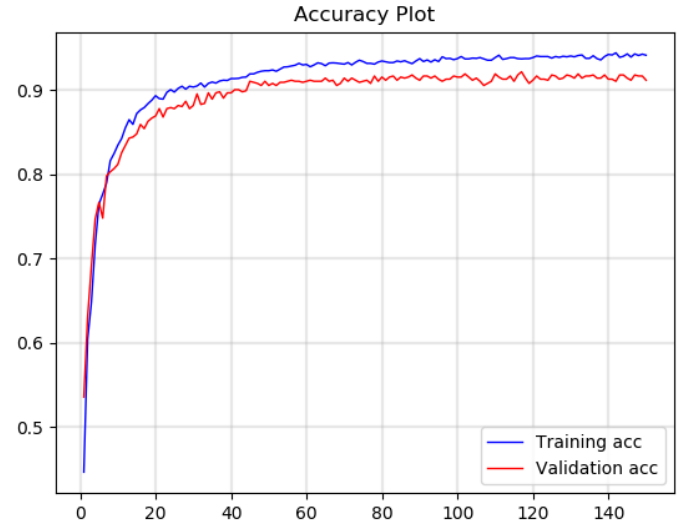


Fig. 3.10. Train Domain model M.T1 accuracy evolution (Fold 1)

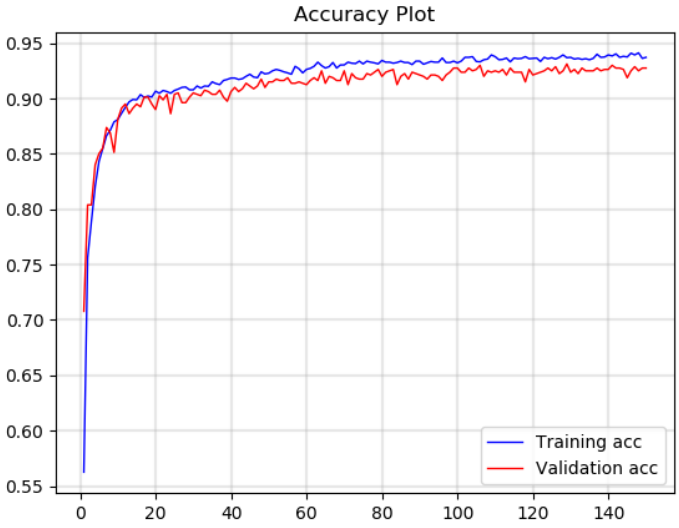


Fig. 3.11. Train Domain model M.T1 accuracy evolution (Fold 2)

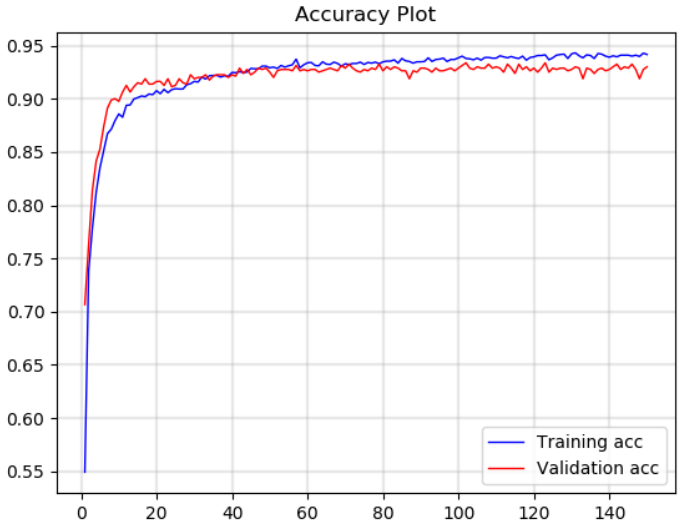


Fig. 3.12. Train Domain model M.T1 accuracy evolution (Fold 4)

As can be observed, training for this problem is a bit harder, and it takes more epochs to approximate. However, eventually it still converges at around epoch 75. As we can see, the training accuracy continues rising while the validation accuracy stays steady. As this could be the principle of overfitting, it has been decided to stop the learning in epoch 150. The model shows good generalization, and fold results are quite similar, so a successful learning process can be confirmed.

Tables 3.13 and 3.14 represent the confusion matrices made for folds 4 and 5, respectively. For portrayal reasons, only the numeric value of the class will be plotted. The numbering ordered is the same as the one specified when previously identifying each of the labels. For instance, number 1 does not appear (Opening state), number 2 will correspond to asking the destination city of the train, and so on.

As it can be observed from the predicted versus actual plot, the diagonal is the one were all the results are, confirming good predictions. We can also observe that the classes from the lower area of the graph seem to be a bit empty. The reason for this is that there was a very small number of instances labelled with this classes. However, they still classify correctly and are relevant in the context of building a more complete spoken dialog system, so those are the reason why they have been kept.

		Predicted Label																					
		2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
True Label	**2**	**48**	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	**68**	0	0	0	0	0	1	7	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	**60**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	1	0	**217**	5	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	3	**140**	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	7	0	1	1	0	0	**29**	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
	8	0	1	1	0	0	0	**23**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	9	0	0	0	1	4	0	0	**106**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10	10	0	0	0	0	0	0	0	**27**	0	0	0	0	0	0	0	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0	0
	12	0	1	0	0	0	0	0	0	0	0	**13**	0	0	0	0	0	0	0	0	0	0	0
	13	0	0	1	0	1	0	0	0	0	0	0	**8**	0	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0	0
	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0
	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0
	20	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0
	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**

Table 3.13. TRAIN DOMAIN MODEL M.T1 CONFUSION MATRIX
(FOLD 4)

		Predicted Label																					
		2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
True Label	**2**	**27**	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	**75**	0	1	1	0	1	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	**45**	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	5	0	**206**	3	1	0	12	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	3	**130**	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	1	0	**39**	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	1	0	1	0	0	**15**	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	4	1	0	0	**110**	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	10	2	1	2	1	0	0	0	0	**39**	0	0	0	0	0	0	0	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	**6**	0	0	0	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	**20**	0	0	0	0	0	0	0	0	0	0	0
	13	0	0	0	0	0	0	0	0	1	0	0	**9**	0	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
	17	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	**3**	0	0	0	0	0	0
	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0
	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0
	20	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0
	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**4**	0	0
	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0
	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

Table 3.14. TRAIN DOMAIN MODEL M.T1 CONFUSION MATRIX
(FOLD 5)

Results for the designed CNN models are presented in table 3.15.

CNN Models	Options	Accuracy
M.T6	epochs=150; optimizer=adam; α=0.0005	91.94%
M.T7	epochs=150; optimizer=adam; α=0.0005	91.66%
M.T8	epochs=150; optimizer=adam; α=0.0005	**91.98%**
M.T9	epochs=150; optimizer=adam; α=0.0005	91.69%
M.T10	epochs=150; optimizer=adam; α=0.0005	80.68%
M.T11	epochs=150; optimizer=adam; α=0.0005	91.39%

Table 3.15. TRAIN DOMAIN CNN MODELS RESULTS

Results for the CNN models range from 80.68 to 91.98 percent. As with the pizza domain case, we arrive to the same conclusion: **Deep Learning outcomes are not getting better than traditional architectures**.

Best results are obtained by model **M.T8**, which achieves an accuracy of **91.98%**. It is relevant to notice that this architecture is the simplest one, with just a convolutional and a pooling layer. However, other complex architectures achieve the same results, such as M.T6, which obtains a 91.94%.

Let's stop to comment the case of M.T10. While all the models tried usually are in the same accuracy range, this model falls apart by about eleven percent points. Figure 3.13 plots its accuracy evolution, and it can be observed that the model converges at around 80% and keeps the same result for almost 90 epochs. There is no apparent explanation of it, since other models are even mode complex, like M.T11. For this reason, it has been dismissed for further analysis.

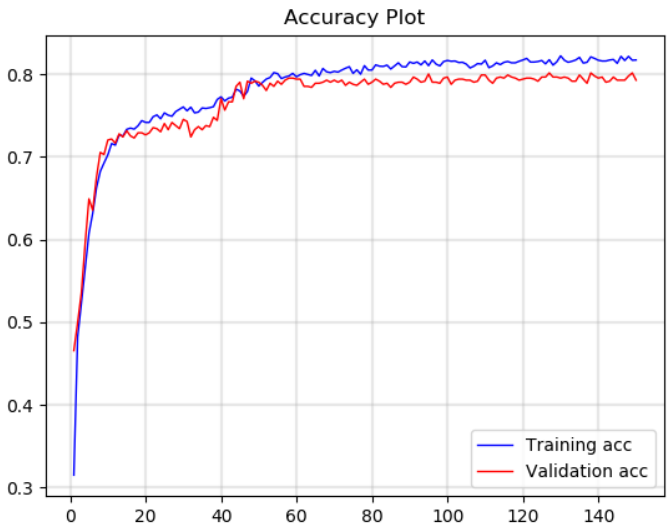


Fig. 3.13. Train Domain model M.T10 accuracy evolution (Fold 4)

Taking as a reference model M.T8, which is the best model, plots are shown for the accuracy evolution and confusion matrix.

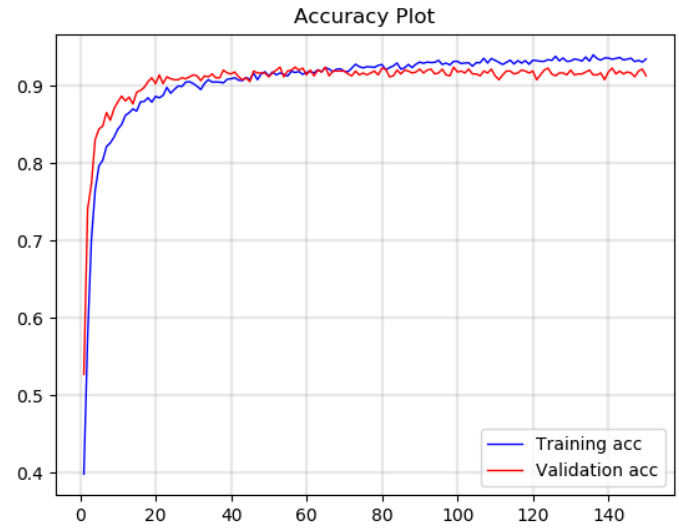


Fig. 3.14. Train Domain model M.T8 accuracy evolution (Fold 1)

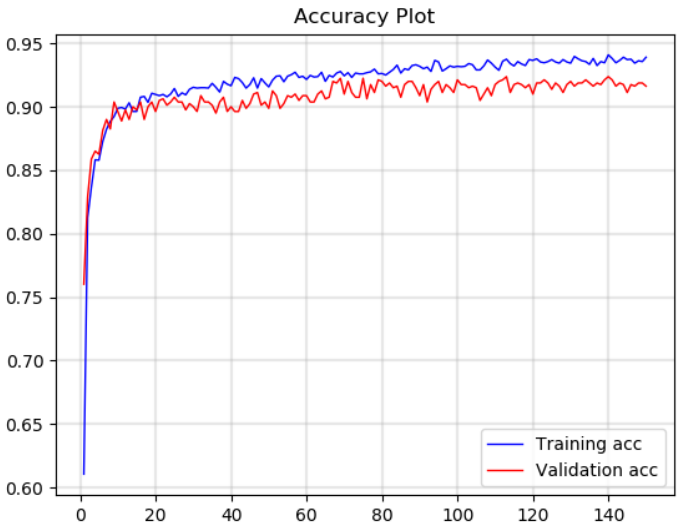


Fig. 3.15. Train Domain model M.T8 accuracy evolution (Fold 3)

Training process is harder than for the pizza domain, showing some irregularities in the learning process. However, we can appreciate convergence at around 92%. Good generalization is also shown, with no sign of overfitting in the model. Therefore, a satisfactory learning can be affirmed.

Table 3.16 represents the confusion matrix made for fold 1. As with model M.T1, there is no sign of a high misclassification rate in any specific class. As a result, the Deep

Learning model is completely capable of making accurate predictions in this domain, with very few error probability for any of the system scenarios.

		Predicted Label																					
		2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
True Label	**2**	**39**	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	**76**	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	1	0	**47**	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	5	0	2	0	**204**	2	0	0	7	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	1	0	2	**140**	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	7	1	2	0	0	0	**46**	0	1	2	0	0	0	0	0	0	0	0	0	1	0	0	0
	8	0	5	1	0	0	0	**9**	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	2	3	0	0	**112**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10	2	5	0	0	0	0	0	0	**23**	0	3	0	0	0	0	0	0	0	0	0	0	1
	11	0	0	0	0	0	0	0	0	0	**7**	0	0	0	0	0	0	0	0	0	0	0	0
	12	0	1	0	1	0	0	0	0	0	0	**15**	0	0	0	0	0	0	0	0	0	0	0
	13	0	0	1	0	0	0	0	0	1	0	0	**9**	0	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0	0
	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0
	17	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0
	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	1	0	0
	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0
	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
	23	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	**2**

Table 3.16. TRAIN DOMAIN MODEL M.T8 CONFUSION MATRIX
(FOLD 1)

For the train domain, we also decided to merge several trained networks and build a majority-vote ensemble. In total, six networks were chosen, combining two MLP and four CNN architectures: M.T1, M.T4, M.T6, M.T8, M.T9 and M.T11. Results obtained by the ensemble are shown in table 3.17.

Ensemble	Options	Accuracy
M.Ensemble	epochs=150; optimizer=adam; α=0.0005	92.71%

Table 3.17. TRAIN DOMAIN ENSEMBLE MODEL RESULTS

The ensemble classifier shows the best of the results in accuracy for the domain. When combining all the models, the lack of precision of a model in a specific set of instances can be counteracted by the rest of the models, which may approximate better in those examples. However, the improvement is not very significant, indicating that around 7% of the example set may be hard to classify correctly without incurring in overfitting. However, building such ensemble is a recommended technique for this domain.

3.3. Conclusions

From the analysis being made, we can extract several relevant conclusions.

The main conclusion is that using CNNs for this datasets does not improve other traditional ML techniques like neural networks or regression trees. As a result, they are not recommended. After a exhaustive analysis, we have predicted that the data examples may not have a local correlation. For this reason, CNNs do not provide advantages, because they stand out of other algorithms when it is important to detect local characteristics, as it happens with images.

One solution to this problem, where CNNs could really make a difference, is to represent the examples with another codification. For example, using word2vec codification, where relations between adjacent words are kept, could be a good scenario for applying Deep Learning rather than MLPs. As such, alternative codification could be tried in future research.

Another approach could be to use a much larger dataset, which traditional algorithms could not handle. However, in the dialog management domain, it is really hard to find large datasets, so work to build synthetic ones could also be a good strategy.

We have also extracted a valuable lesson regarding network architectures. Although, theoretically, a more complex architecture could approximate functions more precisely, in the practice this is not always the case. In both of the studied models, simpler MLP architectures won the more complex ones, and CNNs with less amount of convolutional and pooling layers have usually returned better results.

The last relevant conclusion after the analysis has to do with the use of ensembles. Although the datasets or algorithms used do not provide better outcomes, one can always trying to combine such things to see if that improves the experiments. In our case, the use of ensembles has risen the accuracy and has provided valuable information about the nature of our corpus.

4. IMPLEMENTATION OF THE CONVERSATIONAL AGENT

This section explains the process followed to implement a real conversational agent using one of the dialog manager models built with Deep Learning methodologies. In particular, the dialog manager implemented will correspond to the train scheduling corpus.

The development process will cover the use of DialogFlow, a framework to construct chatbots and spoken dialog systems developed by Google, to define the elements needed for a correct natural language understanding platform. Besides, we will explain how to integrate the model with DialogFlow to respond according to the neural network output, and how to save the information with Firebase Functions and Firebase Realtime Database.

4.1. DialogFlow Basic Elements

Although we cannot go through the entire DialogFlow documentation, we will explain the basic concepts required to understand our CUI implementation. As it was previously developed, a dialog system requires of five basic elements: the automatic speech recognizer, the language understanding module, the dialog manager and natural language generator (which are generally grouped into dialog manager), and the text-to-speech synthesizer. DialogFlow, as part of Google's development toolkit, already comes with the ASR and TTS modules. The dialog manager will be provided by one of the models we crafted during the experimentation phase. As a result, the only module that is left to build is the natural language understanding module.

For this purpose, DialogFlow has three basic elements that developers can use to build their systems: intents, entities and contexts [90].

4.1.1. Intents

Intents are the main element to build conversations. Each intent defines examples of user utterances that can trigger the intent, what to extract from the utterance, and how to respond. As a result, depending on the user input, the agent will map to a specific intent, in order to provide a response. Therefore, it represents one dialog turn within the conversation.

Intents consists of four main components:

1. **Intent name**. Used to identified the matched intent.

2. **Training phrases**. Examples of what users can say to match a particular intent. From the ones the developer provides, DialogFlow automatically expands the phrases to match similar user utterances.

3. **Actions and parameters**. Defines the relevant information extracted from user utterances. Examples of this kind of information include dates, times, names, places, and more.

4. **Response**. The system output that is displayed to the user. In our case, responses will not be defined and, instead, user's input will be send to a webhook that, using our model, will provide the response back to the user. This process will be explained in next section (DialogFlow Fulfillment).

For the train scheduling SDS, the conversations gathered in the data corpus were taken into account in order to build the intent set. Note that, as we would like to simulate a national railway company (RENFE), intents have been defined in Spanish. Table 4.1 shows the considered intents, along with their translation to English.

Intent Name	Training Phrases	Parameters
Change-Departure-Date	¿Y a las 4 de la tarde ? *(And for 4 pm ?)* ¿Y para el 4 de abril de 2019 ? *(And for the 4th of April 2019 ?)* ¿Puedes decirme para el 3 de mayo a las 5:00 ? *(Could you tell me for May the 3rd at 5:00 ?)*	departureDate departureHour
Confirmation	Claro *(Of course)* Es correcto *(That is correct)* Correcto *(Correct)* Sí *(Yes)*	-
Say-Destination	Mi destino es Barcelona *(My destination is Barcelona)* Quiero ir a Barcelona en un AVE *(I want to go to Barcelona by AVE)* A Barcelona *(To Barcelona)* Ir a Barcelona *(Go to Barcelona)* A Barcelona el día 7 de abril *(To Barcelona the 7th of April)* Viajo a Barcelona para mañana *(I am travelling to Barcelona tomorrow)* El destino es Barcelona *(The destination is Barcelona)* Voy a Barcelona el día 8 de mayo en AVE *(I am going to Barcelona the 8th of May by AVE)* Viajo a Barcelona *(I am travelling to Barcelona)*	destination departureDate trainType

Intent Name	Training Phrases	Parameters
Say-Departure-Date	Para mañana *(For tomorrow)* Me gustaría salir el 2 de abril *(I would like to depart April the 2nd)* Para mañana a las 3 *(For tomorrow at 3)* Salgo el 4 de marzo a las 8 de la tarde *(I depart March the 4th at 8 pm)* Me gustaría coger el tren a las 5 y cuarto de hoy *(I would like to take the train today at quarter past 5)* Me gustaría salir el 2 de abril a las 16:00 *(I would like to depart April the 2nd at 16:00)* Me gustaría coger el tren el 3 de abril *(I would like to take the train April the 3rd)* Salgo el 4 de marzo *(I depart March the 4th)*	departureDate departureHour
Say-Arrival-Hour	Sí, quiero que me digas los que lleguen a las 4:00 *(Yes, I would like to know the ones that arrive at 4:00)* Quiero que llegue a las 4:00 *(I would like the train to arrive at 4:00* Me gustaría llegar a las 3:00 *(I would like to arrive at 3:00)* La hora de llegada debe ser las 3:00 *(The arrival hour must be 3:00)* Busca horarios para trenes que lleguen a las 2:00 *(Search for trains arriving at 2:00)*	arrivalHour
Welcome	Buenas tardes *(Good afternoon)* Muy buenas *(Hi there)* Buenas noches *(Good evening)* Buenas *(Hi)* Hey *(Hey)* *Hola (Hello)* Buenos días *(Good morning)* Saludos *(Greetings)* Chao *(Chao)*	-
Denial	Eso no es lo que he dicho *(I did not say that)* Está mal *(That is incorrect)* Incorrecto *(Incorrect)* No *(No)* No quiero nada más *(I do not want anything else)* No, gracias *(No, thank you)* No, muchas gracias *(No, thank you so much)* Gracias por tu ayuda. Hasta pronto *(Thanks for your help. See you soon)*	-
Not_Understood	Fallback Intent, triggered if a user's input is not matched by any of the regular intents.	-

Intent Name	Training Phrases	Parameters
Ask-Schedule	Tren desde Madrid *(Train from Madrid)* Me gustaría saber los horarios desde Madrid para mañana *(I would like to know the schedule from Madrid for tomorrow)* ¿Me puedes decir los trenes que salgan a las 5:00? *(Could you tell me the trains that depart at 5:00?)* Quiero un tren desde Madrid a Barcelona el día 4 de abril *(I would like a train from Madrid to Barcelona for April the 4th).* Quiero un tren desde Madrid a Barcelona *(I would like a train desde Madrid a Barcelona)* Horarios de tren desde Madrid a las 7:00 *(Train schedules from Madrid at 7:00)* Quiero coger un tren desde Madrid a las 8 de la tarde *(I would like to take a train from Madrid at 8 pm)* Me gustaría coger un tren que salga desde Madrid hasta Barcelona *(I would like to take train departing from Madrid to Barcelona)* Me gustaría coger un tren AVE que salga desde Madrid a Barcelona *(I would like to take an AVE train going from Madrid to Barcelona)* Me gustaría coger un tren desde Madrid en clase turista *(I would like to take a train from Madrid in tourist class)* Quiero salir desde Sevilla *(I want to depart from Sevilla)* Quiero un tren desde Madrid en clase turista *(I want a train from Madrid in tourist class)* Me gustaría saber los horarios de AVE para la ruta Madrid a Barcelona *(I would like to know the AVE schedule from the Madrid - Barcelona route)* Quiero los horarios de tren desde Madrid a Barcelona de tipo FEVE *(I want the train schedules from Madrid to Barcelona of type FEVE)* Quiero los horarios de AVE desde Madrid para hoy *(I want the AVE schedules from Madrid for today)* ¿Qué tren puedo coger que salga a las 8 de la tarde? *(Which train could I take that departs at 8 pm?)* Sí, me gustaría saber si alguno sale a las 4:00 *(Yes, I would like to know if anyone departs at 4:00)* Sí, quiero que sea AVE *(Yes, I would like it to be AVE)* Sí, me gustaría que me dijeras los de tipo AVE *(Yes, I would like to know the ones of type AVE)* Sí, me gustaría saber los horarios *(Yes, I would like to know the schedule)*	origin destination ticketClass departureDate departureHour trainType

Intent Name	Training Phrases	Parameters
Ask-Prices	¿Cuánto vale? *(How much does it cost?))* ¿Cuánto cuesta viajar en AVE? *(How much does it cost to travel by AVE?))* Quiero el precio para llegar a las 5:00 *(I want the price for trains arriving at 5:00))* ¿Y cuál es el precio? *(And what is the price?))* ¿Y cuál es el precio de un AVE? *(And what is the price for an AVE?))* Quiero saber el precio de los que lleguen a las 9:00 *(I would like to know the price for the ones arriving at 9:00))* ¿Me puedes decir el precio? *(Could I know the price?))* Quiero el precio de los trenes que lleguen a las 8:00 *(I want the prices of the trains arriving at 8:00))* Quiero saber cuánto vale viajar en AVE *(I would like to know how much does it cost to travel by AVE))* Sí, me gustaría saber los precios *(Yes, I would like to know the price)*	arrivalHour trainType
Ask-Services	Quiero saber qué servicios se incluyen *(I would like to know the available services)* Sí, me gustaría saber los servicios que tiene el tren *(Yes, I would like to know the services the train includes)* ¿Y qué servicios incorpora este tren? *(And what services does this train include?)*	
Ask-Route-Length	Sí, ¿cuál es la duración del trayecto? *(Yes, what is the route length?)* ¿Cuál es el tiempo de recorrido? *(What is the route length?)* Sí, me gustaría saber el tiempo que se tarda *(Yes, I would like to know how long does it take)* ¿Cuánto se tarda? *(How long does it take?)* ¿Cuánto tarda el tren en llegar? *(How long does the train take to arrive?)* Sí, quiero saber el tiempo de recorrido *(Yes, I would like to know the route length)*	
Ask-Train-Type	¿Y cuál es el tipo de tren? *(And which is the train type?)* Me gustaría saber los tipos de trenes disponibles *(I would like to know the available train types)* Sí, quiero saber el tipo de tren *(Yes, I would like to know the train type)* ¿Y qué tipo de tren es? *(And what type of train is it?)* Me gustaría saber el tren que sale a las 5 de la mañana *(I would like to know the train that departs tomorrow at 5 am)* ¿Qué tipo de tren sale a las 4 de la tarde? *(What type of train departs at 4 pm ?)* ¿Algún tren ofrece wifi? *(Does any train offer wifi ?)* Me gustaría saber qué tipo de tren ofrece servicio de películas *(Yes, I would like to know which train type offers movie service)*	services departureHour

Table 4.1. INTENTS DEFINED FOR THE TRAIN SCHEDULING SDS, ALONG WITH THEIR CORRESPONDING TRANSLATION TO ENGLISH

Parameters indicate the type of information that should map in each of the training

phrases. For example, although we have trained the intent *Say-Destination* with the city Barcelona, this just indicates that it is an object of type *destination*, and could be substituted by any other representative of the same type (Madrid, Sevilla, Bilbao...). As a result, any time the intent *Say-Destination* is matched, we will have a value for type *destination* that will give us relevant information about the user's query.

An important remark to make is that not all the parameters appear in each intent. That is because the training phrases have been defined based on the corpus set, and parameters were not mentioned in every type of utterance. To give an example, *services* were not mentioned when asking for an schedule, and for that reason they were not included in such intent.

4.1.2. Entities

Entities are objects that are used as a mechanism for identifying and extracting useful data from natural language inputs. That information can be used and treated as an input into other logic, such as looking up information, carrying out a task, or returning a personalized response.

DialogFlow incorporates a wide variety of predefined system entities, which allow the extraction of information with any additional configuration. Some of them include dates, times, cities, colors, units of measure, or names. However, developers can define their own entities depending on the domain for which the chatbot is being built. In order to do that, two main components are needed:

1. **Entity type**. Defines the type of information you want to extract from user input. For example, a supermarket system may require the entity 'fruit'.

2. **Entity entry**. These are the elements that belong to the same entity type. For the fruits example, entries could include banana, strawberry or orange. Each entry may contain several words, which will be considered to be equivalent or synonyms. For instance, valid words for entry 'banana' could be banana, bananas or musa.

For the train scheduling domain, three special entities were defined, all of them related to common domain vocabulary. Table 4.2 displays these new entities.

Entity Name	Entity Entries
ticketClass	turista (tourist) preferente (preference)
trainType	AVE Alvia Avant Cercanías FEVE Media distancia
services	cafetería (cafeteria) cargador (charger) prensa (newspaper) servicio de películas (movie service) wifi (wifi)

Table 4.2. ENTITIES DEFINED FOR THE TRAIN SCHEDULING SDS, ALONG WITH THEIR CORRESPONDING TRANSLATION TO ENGLISH

Note that the different train type entries have been extracted from the Spanish railway company RENFE [91], in order to be as accurate as possible.

Once these entities have been defined, we can point the type of object that each of the parameters defined belong to. Table 4.3 shows all the parameters used for this domain together with their corresponding entity type (system or crafted).

Parameter Name	Entity Type
origin	city (system)
destination	city (system)
departureDate	date (system)
arrivalDate	date (system)
departureHour	time (system)
arrivalHour	time (system)
ticketClass	ticketClass (crafted)
trainType	trainType (crafted)
services	services (crafted)

Table 4.3. PARAMETERS DEFINED FOR THE TRAIN SCHEDULING SDS

These parameters correspond to each of the domain-dependent slots that were specified for the creation of the Deep Learning model. As a result, every time one of the parameters is mentioned in the intents, that slot will be denoted as mentioned, so as to predict the system response.

4.1.3. Contexts

Contexts represent the current state of a user's request and allow your agent to carry information from one intent to another. They can be combined to control the conversational path that the user will take during the dialog.

For example, if you are doing a Chinese food ordering and you ask the system for the total cost of the operation, the SDS should have stored in a context variable the type and number of food items requested, in order to perform such calculation.

However, for the train scheduling SDS, dialog state was not kept using the DialogFlow context element. In this case, we wanted to keep a huge number of context variables to make the system work, and using contexts became a very hard task to accomplish. For that reason, using Firebase Realtime Database arose as a feasible alternative. As a result, every time a user input is made, the system will look for the last stored state and it will update it with the new information gathered, hence always having an updated dialog state.

Subsequently, a new database table was created, containing a field for every slot needed for the prediction with the model, as well as the next state and the real values of the users' query parameters. Therefore, the table contains the following attributes:

- **prevState**. Numeric value of the previous system response.
- **origin**. Slot value for the origin city (0, 1, 2).
- **originValue**. Real value that the user mentioned for the origin city.
- **destination**. Slot value for the destination city (0, 1, 2).
- **destinationValue**. Real value that the user mentioned for the destination city.
- **departureDate**. Slot value for the departure date (0, 1, 2).
- **departureDateValue**. Real value that the user mentioned for the departure date.
- **arrivalDate**. Slot value for the arrival date (0, 1, 2).
- **arrivalDateValue**. Real value that the user mentioned for the arrival date.
- **departureHour**. Slot value for the departure hour (0, 1, 2).
- **departureHourValue**. Real value that the user mentioned for the departure hour.
- **arrivalHour**. Slot value for the arrival hour (0, 1, 2).
- **arrivalHourValue**. Real value that the user mentioned for the arrival hour.
- **ticketClass**. Slot value for the ticket class (0, 1, 2).
- **ticketClassValue**. Real value that the user mentioned for the ticket class.
- **trainType**. Slot value for the train type (0, 1, 2).
- **trainTypeValue**. Real value that the user mentioned for the train type.
- **orderNumber**. Slot value for the order number (0, 1, 2).
- **orderNumberValue**. Real value that the user mentioned for the order number.

- **services**. Slot value for the train services (0, 1, 2).
- **servicesValue**. Real value that the user mentioned for the train services.
- **querySchedule**. Slot value for the scheduling type of query (0, 1, 2).
- **queryPrice**. Slot value for the pricing type of query (0, 1, 2).
- **queryTypeTrain**. Slot value for the train category type of query (0, 1, 2).
- **queryTrainLength**. Slot value for the route length type of query (0, 1, 2).
- **queryServices**. Slot value for the services type of query (0, 1, 2).
- **affirm**. Slot value for the affirmation user utterance (0, 1, 2).
- **deny**. Slot value for the denial user utterance (0, 1, 2).
- **notUnderstood**. Slot value for the not understanding clause (0, 1, 2).
- **nextState**. Numeric value of the next system response.

4.2. DialogFlow Fulfillment

As we have already seen, intents have a component to provide a system response any time an intent is matched. However, this system is very limited, because responses do not include any kind of backend logic, and they can only be associated to a particular intent. If you want to give different answers depending on the amount of information the system has provided within the same intent, it is necessary to connect to an external service that handles that logic. DialogFlow allows this process very easily, by means of fulfillment. Fulfillment is code that is deployed as a webhook and lets your DialogFlow agent call business logic on an intent-by-intent basis.

As a result, we created a service using Firebase Cloud Functions that handles the information extracted by DialogFlow's natural language processing to generate dynamic responses based on the Deep Learning model previously created. Figure 4.1 shows a summary of how all the pieces are assembled, constituting a complete dialog manager for the spoken dialog system.

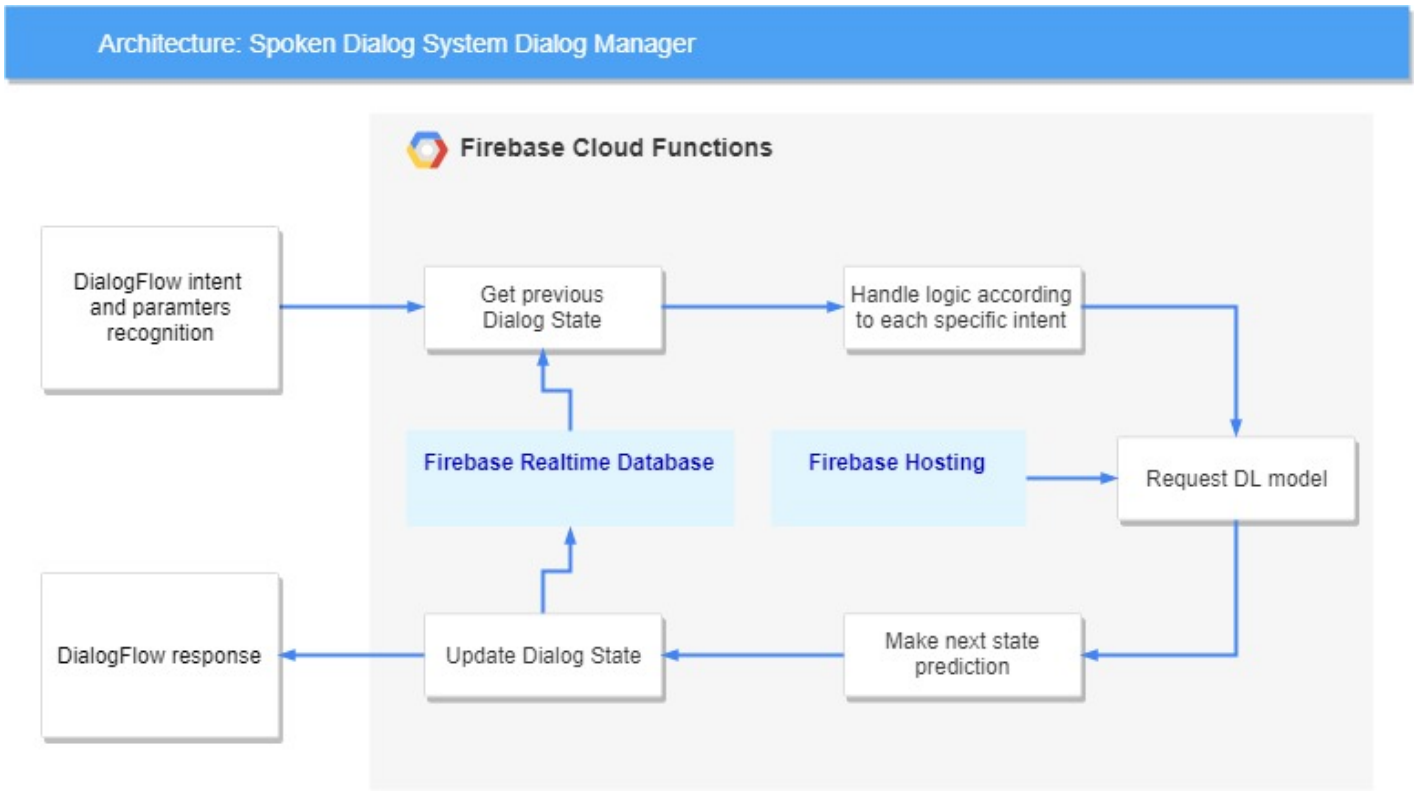


Fig. 4.1. Dialog Manager Architecture for the proposed SDS implementation

We are now going to explain the interactions with the other Firebase services used (Hosting and Realtime Database), so that a complete picture of the response assembling is created.

4.2.1. Model Creation and Integration

The model that has been chosen for the dialog manager is M.T8, as it is the most accurate Deep Learning architecture that was generated during the experimentation phase. Contrary to the process explained during Chapter 3, this time the model was trained with the whole dataset, as the model validation phase has already been made and we wanted the model to learn with all the data available. As a result, a new TensorFlow training was made, this time using the complete corpus.

In order to save the model, Keras offers a basic save format using the Hierarchical Data Format (HDF) standard. The following code shows an example of how to save the model in HDF5 format, after compilation and training phases.

CODE 4.1. Saving an HDF5 model

```
...

model.compile(optimizer=optimizer,
                    loss=loss,
                    metrics=metrics)
```

```
6
7   history = model.fit(train_features, train_labels, epochs=nEpochs,
        validation_data=(train_features, train_labels), verbose=2)
8
9   model.save(NAME+'/'+'train_model.h5')
10
11  ...
```

However, HDF5 models are not readable by JavaScript objects. Therefore, in order to perform the predictions, we need to convert the format to JavaScript Object Notation (JSON). For this, an in-depth tutorial is provided [92] to convert a Keras model to TensorFlow.js.

The first step is to install the Python package *TensorFlowjs*. Although the name might be confusing, this is a Python library to create the *.json* equivalent of a HDF5 model. To achieve this, the following command must be executed:

CODE 4.2. Command to conver to JSON format

```
$ TensorFlowjs_converter --input_format keras \
                         path/to/train_model.h5 \
                         path/to/target_dir
```

Once on the Firebase Cloud Function side, we will need to install *@TensorFlow/tfjs* module with NodeJS. After this, the model usage is pretty simple: first, the model is loaded; later, the input tensor is created; finally, the output is predicted. The following code shows this process:

CODE 4.3. Using the JSON model with JavaScript

```
1
2   import * as tf from '@TensorFlow/tfjs';
3
4   const model = await tf.loadLayersModel('path_to_model/train_model.
        json');
5
6   const example = tf.tensor([input_attributes])
7   const prediction = model.predict(example);
8
9   ...
```

For this project, the model was uploaded to a server using Firebase Hosting services. Therefore, anytime a request is made to the cloud function, a GET petition is made to the URL where the model is hosted, and the model is loaded for subsequent use.

4.2.2. Handling User Intents

Inside the cloud function, there is a specific handler for each of the different DialogFlow intents previously defined. Such handlers receive an argument, which is a conversation object that stores all the variables gathered by DialogFlow during the language understanding phase (for example, parameters). To retrieve an answer, method *ask* within the conversation object must be called, with the text that the system will retrieve back to the user.

To give a brief example of how this works, the code for the *Not_Understood* intent is shown. This is a fallback handler that is called whenever the system does not understand what the user is saying, so it does not even interact with the dialog state.

CODE 4.4. Not_Understood fallback handler

```

// Not understood
app.intent('No_Entiendo', (conv) => {
    return conv.ask(new SimpleResponse({
        speech: "No te he entendido. Podrias repetir, porfavor?",
        text: "No te he entendido. Podrias repetir, porfavor?",
    }));
});
```

The logic behind every handler is very similar. Firstly, the dialog state that is stored in the Firebase Realtime Database is retrieved. After this, depending on the intent and the useful information that the user mentioned, this state will be updated with the new slots. Once this process has been made, a tensor is created to be passed to the TensorFlow model, and the prediction is made. With the updated information and the next system action, the object is stored back to the database, for future intents. Finally, depending on the next system action, a personalized response is made.

The following code will show an example of the *Say-Departure-Date* intent, so as to have a better picture of the process:

CODE 4.5. Say-Departure-Date handler

```

// Say Departure Date
app.intent('Decir-Fecha-Salida', (conv) => {
    const datos = conv.parameters;
    return tf.loadLayersModel('https://trenecitos-1.firebaseapp.com/
        train_model.json').then((model) => {
        return admin.database().ref('stateInfo').transaction((
            stateInfo) => {
            if (datos.horaSalida){
                stateInfo.departureHour = 1;
                stateInfo.departureHourValue = datos.horaSalida;
            }
```

```
            if (datos.fechaSalida){
                stateInfo.departureDate = 1;
                stateInfo.departureDateValue = datos.fechaSalida;
            }

            stateInfo.prevState = stateInfo.nextState;
            const modelInput = createModelInput(stateInfo);
            const prediction = model.predict(modelInput);
            stateInfo.nextState = prediction.argMax(1).dataSync()
                [0];
            return stateInfo;
        }).then(() => {
            return admin.database().ref('stateInfo').once("value").
                then(snapshot => {
              const stateInfo = snapshot.val();
              return sendResponse(stateInfo, conv);
          });
        });
    })
});
```

After loading the model, we perform a query to the database so as to extract the current state. After this, depending on what users said (if they mentioned departure date, departure hour, or both), a 1 will be stored in the corresponding slot (note that DialogFlow does not provide confidence values, so a 1 is always stored in this implementation), and the real value will also be stored in the database. The previous action is updated with the prior next action, and the new next action is the result of the prediction being made the model. After that, the corresponding response is sent back to the user.

There are only two functions left to explain: *createModelInput(stateInfo)*, which generates the input tensor for the model to make a prediction with, and *sendResponse(stateInfo, conv)*, that generates the response for the user. The first one does not imply any difficulty: we are just creating a TensorFlow tensor with the required size so that the model works properly.

CODE 4.6. createModelInput function

```
function createModelInput (stateInfo) {
    return tf.tensor([
      stateInfo.prevState, stateInfo.origin, stateInfo.destination,
      stateInfo.departureDate, stateInfo.arrivalDate,
      stateInfo.departureHour, stateInfo.arrivalHour,
      stateInfo.ticketClass, stateInfo.trainType,
      stateInfo.orderNumber, stateInfo.services,
      stateInfo.querySchedule, stateInfo.queryPrice,
      stateInfo.queryTypeTrain, stateInfo.queryTrainLength,
      stateInfo.queryServices, stateInfo.affirm, stateInfo.deny,
      stateInfo.notUnderstood], [1, 19]);
}
```

The second function, *sendResponse(stateInfo, conv)*, retrieves an answer based on the system action that the model has predicted to be the appropiate one for the current dialog state. For the messages of type 'Response', a random feedback is simulated, as if we had the real data available in our servers. If any national railway company, such as RENFE, gave us access to their database by means of an API, we could provide a real answer to the user, becoming a really useful commercial application.

An example for some of the system actions is shown. For simplicity, only one response is retrieve, but we could change the message to provide more variety to the user and increase the user experience.

CODE 4.7. sendResponse function

```
1
2  function sendResponse(stateInfo, conv) {
3      const type = stateInfo.nextState;
4      let speechMessage = "";
5      let textMessage = "";
6
7      // Ask-Destination
8      if (type === 0) {
9          speechMessage = "A donde desearias viajar?";
10         textMessage = "A donde desearias viajar?";
11     }
12     // Ask-Departure-Date
13     if (type === 1) {
14         speechMessage = "En que fecha desearias viajar?";
15         textMessage = "En que fecha desearias viajar?";
16     }
17     // Confirm-Departure-Date
18     if (type === 2) {
19         const date=new Date(stateInfo.departureDateValue);
20         speechMessage = " Es  correcta la fecha "+date.getDate()
               +"/"+(date.getMonth()+1)+"/"+date.getFullYear()+"?";
21         textMessage = " Es  correcta la fecha "+date.getDate()+"/"+(
               date.getMonth()+1)+"/"+date.getFullYear()+"?";
22     }
23     ...
24     // Closing
25     if (type === 4) {
26         speechMessage = "Espero haber sido de ayuda. Que tenga un
               buen  d  a .";
27         textMessage = "Espero haber sido de ayuda. Que tenga un buen
                d  a .";
28
29         return conv.close(new SimpleResponse({
30             speech: speechMessage,
31             text: textMessage,
32         }));
33     }
```

```
    ...
    // Retrieve-Prices
    if (type === 7) {
        const claseBillete = stateInfo.ticketClassValue;
        speechMessage = "El trayecto seleccionado tiene un coste de
            38 euros en clase "+claseBillete+".";
        textMessage = "El trayecto seleccionado tiene un coste de 38
             euros en clase "+claseBillete+".";

        speechMessage += "\n Necesitas algo mas?";
        textMessage += "\n Necesitas algo mas?";
    }
    // Confirm-Origin
    if (type === 8) {
        speechMessage = "Es correcto el origen "+stateInfo.
            originValue+"?";
        textMessage = "Es correcto el origen "+stateInfo.originValue
            +"?";
    }
    // Confirm-TrainType
    if (type === 9) {
        speechMessage = "Es correcto el tipo de tren "+stateInfo.
            trainTypeValue+"?";
        textMessage = "Es correcto el tipo de tren "+stateInfo.
            trainTypeValue+"?";
    }
    // Answer-TrainType
    if (type === 10) {
        speechMessage = "El tipo de tren del vehiculo seleccionado
            es AVE";
        textMessage = "El tipo de tren del vehiculo seleccionado es
            AVE";

        speechMessage += "\n Tienes alguna otra consulta?";
        textMessage += "\n Tienes alguna otra consulta?";
    }
    ...

    return conv.ask(new SimpleResponse({
        speech: speechMessage,
        text: textMessage,
    }));
}
```

That is the algorithm followed to retrieve an answer to user utterances: an intent is identified, the handler inside Firebase Cloud Functions is triggered, the model is requested to a Firebase Hosting service, the Firebase Realtime Database object is updated, the state is send to the model to make a prediction, and an output message is sent back depending on the prediction made.

4.2.3. Conversational Interface Deployment

One of the reasons why DialogFlow was chosen for this project is its easy deployment into real applications and chatbots. Once the model is built, DialogFlow offers an option named **Integrations**, that allows the developer to integrate the app within commercial chat environments, including:

- Facebook Messenger.
- Twitter.
- Slack.
- Skype.
- Telegram.
- Google Assistant.
- Amazon Alexa.

Each of the environments require a small configuration to connect it to the account you would like to automatize, but the process is really simple. If you want to deploy your application in an independent environment, DialogFlow also allows you to deploy your spoken dialog system as a web service.

Once the configuration is being made, you are ready to go! The conversational interface is deployed and ready to be used. Figures 4.2 and 4.3 show examples of conversations from a web service and Google Assistant integrations, respectively.

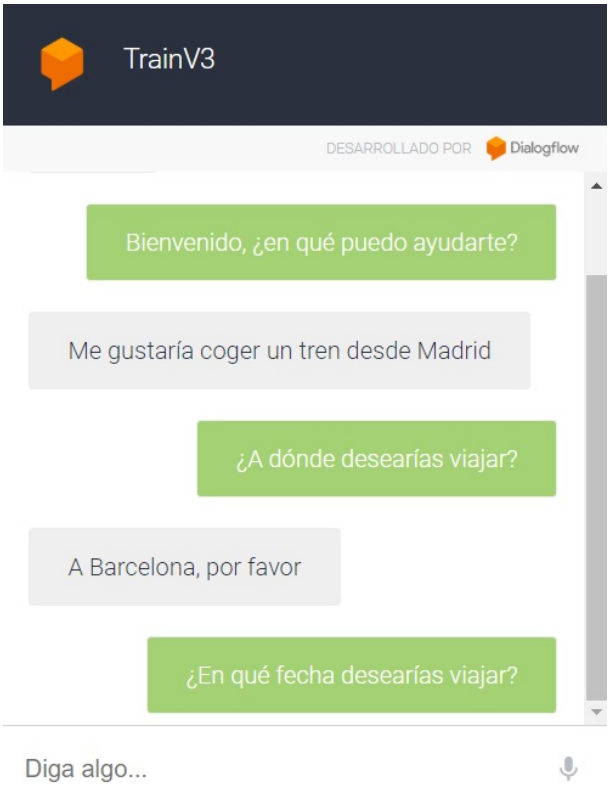


Fig. 4.2. Example of interaction with a web service integration

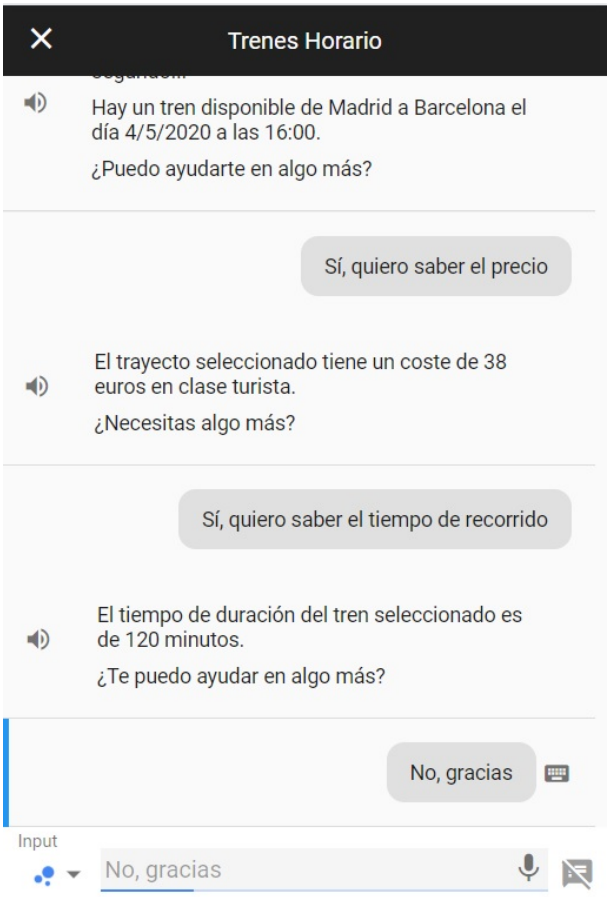


Fig. 4.3. Example of interaction with a Google Assistant integration

It is important to note that some of the options available have some limitations with the automatic speech recognizer and the text-to-speech module. For example, while Google Assistant offer the possibility to both speak to the system and receive a spoken response, Facebook Messenger does not allow the system to speak. It is recommended to have such restrictions in mind when developing an spoken dialog system, because some of the functionality may be limited.

5. EVALUATION OF THE CONVERSATIONAL AGENT

This chapter reports the evaluation process followed for the validation of the conversational agent in a real setting. As a result, an evaluation methodology will be planned in order to prepare a complete assessment of the system. After this, real users will test the CUI, out of which the evaluation will be made.

5.1. Evaluation Methodology

The evaluation process carried out is based on state-of-the-art validation methodologies which proved to be very thoroughgoing assessment procedures. The measuring in which this work has focused the most is the one developed by Griol et al. for evaluating and comparing a wide variety of spoken dialog systems [77]. This methodology is divided into an objective and a subjective evaluation, made from the outcome of showcasing the app to a group of real users.

In this thesis, due to time constraints, a total of 20 people were interviewed to evaluate the train scheduling chatbot. The selection of this people was not random, as we wanted to gather feedback from a very heterogeneous sector of society that could be interested in a solution of this nature. As a result, the population set comprises men and women from an age range between 21 and 60 years old, with the basic knowledge of technology to make them comfortable using a smartphone, and who take at least one medium distance train per month.

All of the users were explained the nature of the app, and that it was conceived to be an interactive conversational interface to solve requests in the train scheduling domain. However, in order to provide a more varied set of requests, users were told different details regarding the app:

- Some of the users were not given any more information, and ask them to try the app to solve their requests. With this, we tried to look for possible outliers and classes we had not taken into account for the modelling process.
- Other users were only told specific functionalities of the application. For example, some users were explained that the system could solve requests regarding scheduling, while others were told to ask about prices and services. With this approach we were testing how the app responded to very specific situations.
- Finally, another set of users were reported about all the system specifications. This population set allowed to test the dialog system as a whole entity.

With this approach we were looking to exploit every different aspect of the system so as to have highly comprehensive assessment.

5.2. Objective Evaluation

For the objective evaluation, we analyzed nine different metrics extracted from the interactions between the user and the system:

M1. Dialog success rate: percentage of dialogs that were finished successfully, with users having their requests resolved.

M2. Turn success rate: percentage of turns in which the system responded with a coherent answer to the user's input. This is not the same as the correct state identification, but every system answer that is coherent and does not break the flow of the dialog.

M3. Dialog length: average number of turns per dialog. Take into account that a turn corresponds to a two-side interaction, so a user-system communication will count as one turn.

M4. Request length: average number of requests made in one dialog. Several requests of the same type (e.g. price) are counted separately.

M5. Number of turns of the shortest dialog.

M6. Number of turns of the longest dialog.

M7. Percentage of different dialogs.

M8. Repetitions of the most observed dialog.

M9. Number of turns of the most observed dialog.

M10. Number of requests of the most observed dialog.

Table 5.1 shows the results obtained after 20 experiments with real users.

Metric	Evaluation Value
M1	80 %
M2	78 %
M3	7
M4	2.89
M5	10
M6	5
M7	55 %
M8	6
M9	5
M10	2

Table 5.1. OBJECTIVE EVALUATION RESULTS AFTER 20 EXPERIMENTS

Results show an 80 % of success rate, which is a very acceptable value for a first prototype of the system. The ratio of coherent answers stands at around the 78 %, where most of the incoherencies are found in dialogs ending unsuccessfully. However, we can also find some dialogs scoring a perfect 100 % rate for turn success rate.

We have found a total of 11 different dialogs from the experimentation with users. The most common one has been a simple request of schedules for a particular route, together with a enquiry about its price. Figure 5.1 displays an example of such dialog, that has had 5 repetitions.

U1: Hola.
(Hi.)
S1: Bienvenido, ¿en qué puedo ayudarte?
(Welcome, how can I help you?)

U2: ¿Puedes decirme los horarios desde Madrid a Cuenca?
(Could you tell me the schedule from Madrid to Cuenca?)
S2: ¿Cuándo desearías viajar?
(When would you like to travel?)

U3: El 30 de mayo.
(The 30^{th} of May.)
S3: Buscando los trenes disponibles, espere un segundo... Hay un tren disponible de Madrid a Cuenca el día 30/5/2019 a las 20:00. ¿Necesitas algo más?
(Searching for available trains, please wait... There is an available train from Madrid to Cuenca on the 30/5/2019 at 20:00. Do you need anything else?)

U4: ¿Qué precio tiene?
(How much does it cost?)
S4: El trayecto seleccionado tiene un coste de 20 euros en clase turista. ¿Puedo ayudarte en algo más?
(The selected route has a cost of 20 euros in tourist class. Can I help you in anything else?)

U5: No, gracias.
(No, thank you.)
S5: Espero haber sido de ayuda. Que tenga un buen día.
(I hope I have been helpful. Have a nice day.)

Fig. 5.1. An example of the most common dialog extracted from the evaluation experiments, along with its translation to English

Figure 5.2 shows one of the four failed dialogs that were obtained during the testing process. As it is the case of this dialog, most of the unsuccessful terminations were due to second or third requests that were not handle correctly. For the example given, the user is asking for another train schedule, and the system does not understand the query or retrieves an answer for other type of enquiry.

The query that has represented a major issue is the services one. For the three users that asked such question, the system did not know how to retrieve a valid answer. We think that the reason for this is the low training data containing the query, that has made the model not to approximate correctly for these intents.

Most of the incoherent answers were due to the wide variety of formulations that different people have for the same utterance. Some of these formulations were not included as training phrases in DialogFlow, so inputs were not recognized. As a result, a task for a future version of the product would be to include more combinations of examples for the training set.

A very interesting comment to make is the differences perceived among the different age groups. Adult people, maybe accustomed to older and lower quality speech recognizers, where formulating their phrases omitting pronouns, articles or verbs. For example, while a younger person would say *'I would like a train from Madrid to Valencia'*, and adult would say *'Tran Madrid Valencia'*. Some of these utterances were not understood by the system, so there is a lot of work to do in there too. No only to include such examples, but also to make older people feel more comfortable using the system, so they interact as if they were speaking with a human.

Figure 5.3 shows an example of a successful dialog extracted from one of the tests. As it can be observed, the user was speaking with a colloquial wording, adding unnecessary extra information and other tags such as *'more or less'* or *'everything has become clear to me'*. However, the system responded with highly accurate responses, becoming a great example of the potential of this system for commercial purposes.

U1: Buenos días.
(Good morning.)
S1: Bienvenido, ¿en qué puedo ayudarte?
(Welcome, how can I help you?)

U2: Quiero saber los trenes de Madrid a La Coruña.
(I want to know the trains from Madrid to La Coruña.)
S2: ¿En qué fecha desearías viajar?
(In which date would you like to travel?)

U3: 30 de mayo.
(30^{th} of May.)
S3: Buscando los trenes disponibles, espere un segundo... Hay un tren disponible de Madrid a La Coruña el día 30/5/2019 a las 14:00. ¿Puedo ayudarte en algo más?
(Searching for available trains, please wait... There is an available train from Madrid to La Coruña on the 30/5/2019 at 14:00. Can I help you in anything else?)

U4: ¿Hay alguno más?
(Is there any other?)
S4: No te he entendido. ¿Podrías repetir, por favor?
(I did not understand you. Could you repeat, please?)

U5: ¿Hay algún tren más ese día?
(Is there any other train for that day?)
S5: El tipo de tren del vehículo seleccionado es AVE. ¿Tienes alguna otra consulta?
(The train type of the selected vehicle is AVE. Do you have any other enquiry?)

U6: Algún otro tren después de las 14 horas.
(Any other train after hour 14.)
S6: Espero haber sido de ayuda. Que tenga un buen día.
(I hope I have been helpful. Have a nice day.)

Fig. 5.2. An example of a failed dialog extracted from the evaluation experiments, along with its translation to English

U1: Muy buenas tardes.
(Good afternoon.)
S1: Bienvenido, ¿en qué puedo ayudarte?
(Welcome, how can I help you?)

U2: ¿Podría decirme los horarios de trenes de Madrid a Zaragoza, por favor?
(Could you please tell me the train schedule from Madrid to Zaragoza?)
S2: ¿En qué fecha desearías viajar?
(In which date would you like to travel?)

U3: Para mañana, si es posible.
(For tomorrow, if possible.)
S3: Buscando los trenes disponibles, espere un segundo... Hay un tren disponible de Madrid a Zaragoza el día 26/5/2019 a las 16:00. ¿Puedo ayudarte en algo más?
(Searching for available trains, please wait... There is an available train from Madrid to Zaragoza on the 26/5/2019 at 16:00. Can I help you in anything else?)

U4: ¿Cuánto me costaría ese billete?
(How much would that ticket cost me?)
S4: El trayecto seleccionado tiene un coste de 31 euros en clase turista. ¿Necesitas algo más?
(The selected route has a cost of 31 euros in tourist class. Do you need anything else?)

U5: ¿Cuánto tiempo dura el viaje, más o menos?
(How long does the trip take, more or less?)
S5: El tiempo de duración del tren seleccionado es de 90 minutos. ¿Te puedo ayudar en algo más?
(The selected train duration is 90 minutes. Can I help you in anything else?)

U6: ¿Y qué tipo de trenes hay disponibles en ese billete?
(And what type of trains are available with that ticket?)
S6: El tipo de tren del vehículo seleccionado es AVE. ¿Tienes alguna otra consulta?
(The train type of the selected vehicle is AVE. Do you have any other enquiry?)

U7: No, muchas gracias, me ha quedado todo muy claro. Muy amable.
(No, thank you, everything has become clear to me. Very kind of you.)
S7: Espero haber sido de ayuda. Que tenga un buen día.
(I hope I have been helpful. Have a nice day.)

Fig. 5.3. An example of a successful dialog extracted from the evaluation experiments, along with its translation to English

5.3. Subjective Evaluation

Besides the objective metrics, we also asked the users to assess their subjective opinion on the system's performance. A total of six questions were asked:

Q1. How well did the system understand you?

Q2. How well did you understand the system messages?

Q3. Was it easy for you to get the requested information?

Q4. Was the interaction with the system quick enough?

Q5. If there were system errors, was it easy for you to correct them?

Q6. In general, are you satisfied with the performance of the system?

Q7. Would you use this system to schedule your future train rides?

Each of the questions had a set of predefined answers, with an associated scoring from 1 (lowest) to 5 (highest). Those were:

1. Never/Not at all.
2. Rarely/Poorly.
3. Sometimes/In some measure.
4. Usually/Well.
5. Always/Very well.

Table 5.2 shows the results obtained after 20 experiments with real users.

Question	Average Response	Standard deviation	Median Response	Mode Response
Q1	3.80	0.83	4	3
Q2	5	0	5	5
Q3	4	1.12	4	5
Q4	4.60	0.71	5	5
Q5	3.30	1.41	4	4
Q6	4.40	0.73	5	5
Q7	4.20	0.68	4	4

Table 5.2. OBJECTIVE EVALUATION RESULTS AFTER 20 EXPERIMENTS

As it can be seen, results are generally positive, with most of the answers scoring between 4 and 5. The weakest points of the application have appeared to be related with the system understanding in some situations. Users would perceive that the system was coherent usually but, once an mistake was made, it was sometimes difficult to solve.

All the other points were scored as highly positive for the users. There is a total agreement for question 2, where users thought system messages were extremely clear. Also, most of the respondents believed that the interaction was very fast, being a weak point in many other spoken dialog systems.

In general, users are very satisfied with the performance of the system. Most of them believe that it was easy to get the information that they were looking for, and a great percentage of interviewees affirmed that they would use the system to schedule their future train rides.

For all these reasons, the system has been evaluated as successful, although improvements are needed to be made for future versions.

6. REGULATORY FRAMEWORK

This chapter examines the thesis from the legal point of view, reviewing any possible statute, ordinance or legal regulation that could be applied to this work. Besides, it details the technical standards of the application developed, as well as commenting any possible violation of intellectual property.

6.1. Applicable Regulations

The potential legal regulations applicable to this work are related to the use and distribution of data and information. Therefore, the current legislation has been consulted to check if the thesis is in line with the Spanish legal framework.

The corpora used in the thesis were extracted from two different sources:

1. Cession by Professor Dr. Griol Barres of the data used for his PhD thesis.
2. Open repository from Dra. Callejas, as a complement to the book *The Conversational Interface*. [93].

These datasets were given with a free of charge end-user-license. Therefore, according to the "Ley de Propiedad Intelectual" approved in the "Real Decreto Legislativo 1/1996" [94], the author remains fully entitled to use these data for research purposes.

Regarding the content of the data, another law article was consulted, in particular the "Ley Orgánica 3/2018, de Protección de Datos Personales y garantía de los derechos digitales (LOPD-GDD)" [95]. In the Title One Article Two Point One states where this law is applicable, and claims:

"...la presente ley orgánica se aplica a cualquier tratamiento total o parcialmente automatizado de datos personales, así como al tratamiento no automatizado de datos personales contenidos o destinados a ser incluidos en un fichero."

Corpora used for the thesis was manipulated and cleaned by the intermediary responsibles, and contain no personal data of any kind. For this reason, the Law is not applicable for this case study.

Based on the previously cited Spanish Law articles, it can be concluded that this work is within the law standards of the country.

6.2. Technical Standards

One of the main parts of this project was the implementation of a real conversational agent, out of the experimentation we had previously done with Machine Learning and Deep Learning techniques. The technical standards of such application will now be described.

Description of the software implemented

Application that consists of a conversational assistant within the domain of train scheduling. The user can ask the agent for information regarding the timetable, prices and other services of the train, and will receive spoken answers for the requested questions. As such, this software will serve as an interactive assistant for solving any doubt of the user concerning train travels.

Programming Languages

For the development of this software, the following programming languages have been used:

- **Python 3.7.** for the creation of the Machine Learning model used for the dialog management system.
- **JavaScript ES5** for the logic and backend of the application.

Frameworks and third party services

For the development of this software we have used the following platforms:

- Google's **TensorFlow version 1.13.1** for the design and training of the Machine Learning model used for the dialog management system.
- Google's **DialogFlow V2** for the implementation of all the components of the spoken dialog system and the deployment of the conversational interface into a web application.
- Google's **Firebase Realtime Database** to store all the information required to keep a consistent dialog state throughout the turns.
- Google's **Firebase Hosting** to host the Machine Learning model in a server so as to integrate the algorithm into the dialog manager.
- Google's **Firebase Cloud Functions** to create a webhook that connects DialogFlow's language processing module with the hosting and database systems, in order to deliver the appropiate response back to the user.

6.3. Intellectual Property

With the respect to the software used, a justification for all the licenses is now provided:

- **Atom**: on 6 May 2014, Atom, including the core application, its package manager, as well as its desktop framework Electron, were released as free and open-source software under the MIT License [96]. This is one of the most permissive licenses, and allows almost any kind of usage of the software.
- **Weka**: it is a free software licensed under the GNU General Public License by the University of Waikato, New Zealand. This license offers high protection for the author of the software, but allows free private use. As the application is being used exclusively for research purposes, we are not entering in conflict with the license restrictions.
- **TensorFlow**: on 9 November 2015, it was released under the Apache 2.0 open-source license [97]. This license allows the free use and distribution under any license of products made with the software.
- **DialogFlow**: by using the Standard Edition of DialogFlow, we are agreeing to the Google APIs Terms of Service, which allow the use and distribution of software based on Google's API services as long as no other illegal restrictions are held.
- **Firebase**: by using Firebase services, we are agreeing to the Google APIs Terms of Service, which allow the use and distribution of software based on Google's API services as long as no other illegal restrictions are held.
- **Microsoft Office 365 ProPlus**: belonging to the Universidad Carlos III de Madrid (UC3M) the author has access to a Microsoft Office Education License, which allow the free exploitation of the Office software.

As a result, we are not incurring in any violation of the intellectual property for the tools used in this thesis.

7. SOCIO-ECONOMIC ENVIRONMENT

This chapter studies the potential socio-economic impact that could be derived from the realization of this thesis, listing some areas where it could be applied for both economic benefit and improvement of living standards. It also provides an overview of the planning conceived for the project, as well as the budgeting schema.

7.1. Socio-economic impact

The content of this project has a tremendous social and economic impact in the digital age where we live. While Chapter 3 was more oriented towards a research sight, Chapters 4 and 5 are highly related to an economic element that is dramatically increasing its popularity in society: chatbots and conversational interfaces.

As it was previously seen in the state of the art, technological advancements have substantially increased advanced communication capabilities of these devices. On the other hand, people have always seen and read in science fiction about the potential of AI, and in recent years it seems that we are coming closer to that reality. As a result, this conversational interfaces in the form of virtual personal assistants or applications have become the spearhead of the new intelligent era. Just to give a quick piece of information, last year Amazon reported tens of millions of Echo devices sold, and its personal VPA Alexa told more than 100 million jokes [98].

These are pretty impressive numbers, and it is the surface of the exponential growth that is expected for the incoming years. Such is the interest in this technology that Google has made a huge investment on improving its assistant, being one of the core announcements made during the Google I/O 2019 conference [99].

In summary, it is an industry that is generating eight-figure revenues to companies all around the world. But big organizations are not the only ones that are benefiting from this technology: many small and medium enterprises are incorporating chatbots and assistants to their businesses, in order to automatize their labour and increase user satisfaction. In fact, many businesses are multiplying revenues thanks to the implementation of chatbots. It is for this reason that many AI chatbot expert vacancies are flourishing, which is extremely good news for the field.

In particular, this thesis presents a methodology that can be applied to any commercial problem: it is domain-independent. Therefore, it can be applied to any industry in which an online assistant could be needed, both for the customer or for internal use. Some examples include: analysing the financial risk of a person, evaluating the nutritional profile of a person and giving tips for a healthy lifestyle, e-commerce online support or a tool for advising a choice of university studies based on the student profile. The possible impact

on two specific domains will be further developed: the train scheduling domain, which actually is the CUI developed for this thesis, and the depression treating domain, which is a side project that the author is building with the help of the university.

Train scheduling solution

This is the system that was implemented in the thesis, and it represents an example of impact from the economic perspective. The transportation sector in Spain is a really important one, with thousands of passengers travelling every day from one city to another.

Having a system that helps the user to find information regarding a desired ride can mean a multiplied revenue stream for the company. This could come from:

- Reduction of the support staff team due to automation of processes.
- Increase in the turnover rate because passengers find their desired travelling option.
- Higher fidelity rate due to raised user satisfaction.

The system could be sold to a railway operator such as RENFE, for a monthly fee. For training the model one would need to catalog and preprocess real conversations between the clients and the company, but once that is done it would be really easy to build the system. Besides, maintenance or upgrading would not entail an extra effort, so there is no uncertainty in that sense.

The most valuable characteristic of kind type of system is its great scalability. Once you have a system for train scheduling, it could be easily adapted to other railway companies. Moreover, transition to other means of transport (buses, planes...) could be made without carrying an added difficulty. Therefore, a strong startup business plan could be drafted from this idea.

Depression treatment solution

This system represents an example of impact from the social point of view. The idea has been presented as part of UC3M's *TFG Emprende 2018/2019* program [100]. Such contest aims students to get involved in the culture of entrepreneurship, innovation and the adoption or development of new technologies by developing a novel solution to an existing problem.

It is estimated that around 10% of the population suffers from depression in the world, and nearly a quarter of the people will present some of the symptoms related to an early depression. This is a very insightful fact because, according to the statistic, we should know more cases of people suffering from the illness than we actually do.

The reason for this is that depression is somehow taboo in society. People feel ashamed of suffering from depression because they think that the rest would never understand the situation. However, it has been proven that one of the best ways to overcome the illness

is to openly expose your situation and talk about it with your close circle of family and friends.

In here is where our idea comes into play. We believe that patients would feel more comfortable if they could chat with an entity that would never judge them. For this reason, our team is designing a conversational interface to help to treat depression cases. The idea is to have a chatbot which the user can interact with at any time. The device will have been previously trained in the psychological domain so that it can give an appropriate response to the users, while proposing activities or games to cheer them up. Our concept is not meant to substitute real psychological treatment, but to serve as a complement that is available twenty-four/seven for the patient.

Although the project is still in prototyping phase, the idea has been well received in the community, so we believe that it could have big potential.

These are just some examples of the huge impact that a conversational interface may have in our socio-economic environment. For that reason, we consider that this technology will be very relevant in the years to come.

7.2. Planning

This project was granted by the Ministerio de Educación y Formación Profesional of Spain, as part of its Beca de Colaboración 2018/2019 scholarship program [101]. As part of the conditions of the call, beneficiaries will have to work on their project at least three hours per business day during eight months, summing a total of 510 hours. As such, the author started collaborating on this thesis the 1^{st} of November 2018 and finished working on it the 29^{th} of June 2019, meaning a work of 240 days and approximately 15 hours a week.

The project was divided in well-defined phases in order to structure the work:

- **Phase 1: Planification.** This was the preparatory task of the thesis. In here, we would have to investigate and read other author publications in order to get an overall glimpse of the state of art and conceive a novel idea for our project. In particular, we divided this phase in the following subtasks:

  1. Define the objectives of the thesis.
  2. Search for references and documentation in the area.
  3. Read such papers in order to understand previous work on the area.
  4. Analysis of the possible tools to use for the development of the project

  The intended amount of time planned for this stage was 200 hours (94 days).

- **Phase 2: Experimentation.** This stage included the analysis made for Deep Learning applied to the development of the dialog manager. We would have to prepare the data for its modelling, and evaluate the results. In particular, we divided this phase in the following subtasks:

  1. Preprocessing of the corpus chosen for analysis.
  2. Design and programming of the architectures for the model experimentation.
  3. Comparison of the different Machine Learning models for several domains.

  The intended amount of time planned for this stage was 100 hours (47 days).

- **Phase 3: Implementation.** In this stage we created our spoken dialog system with DialogFlow. In particular, we divided this phase in the following subtasks:

  1. Creation of the spoken language understanding module with DialogFlow.
  2. Creation of the infrastructure needed to support the backend with Firebase.
  3. Programming and integration of the dialog manager model with DialogFlow.
  4. Deployment of the application.

  The intended amount of time planned for this stage was 85 hours (40 days).

- **Phase 4: Evaluation.** This stage included the validation of the system created by means of experimentation. In particular, we divided this phase in the following subtasks:

  1. Testing with real users.
  2. Analysis and evaluation of the system based on testing.

  The intended amount of time planned for this stage was 30 hours (14 days).

- **Phase 5: Documentation.** This stage included the reporting of all the work done for the thesis, as well as preparing the presentation for the jury. In particular, we divided this phase in the following subtasks:

  1. Writing-up of the report.
  2. Revision of the report.
  3. Creation of the slides for the presentation.
  4. Revision of the presentation.

  The intended amount of time planned for this stage was 95 hours (45 days).

The final timing of the project is represented as a Gantt Diagram using the tool Tom's Planner, which is a well-known project planner software [102]. Figure 7.1 contains the timing details for each of the tasks involved in the thesis.

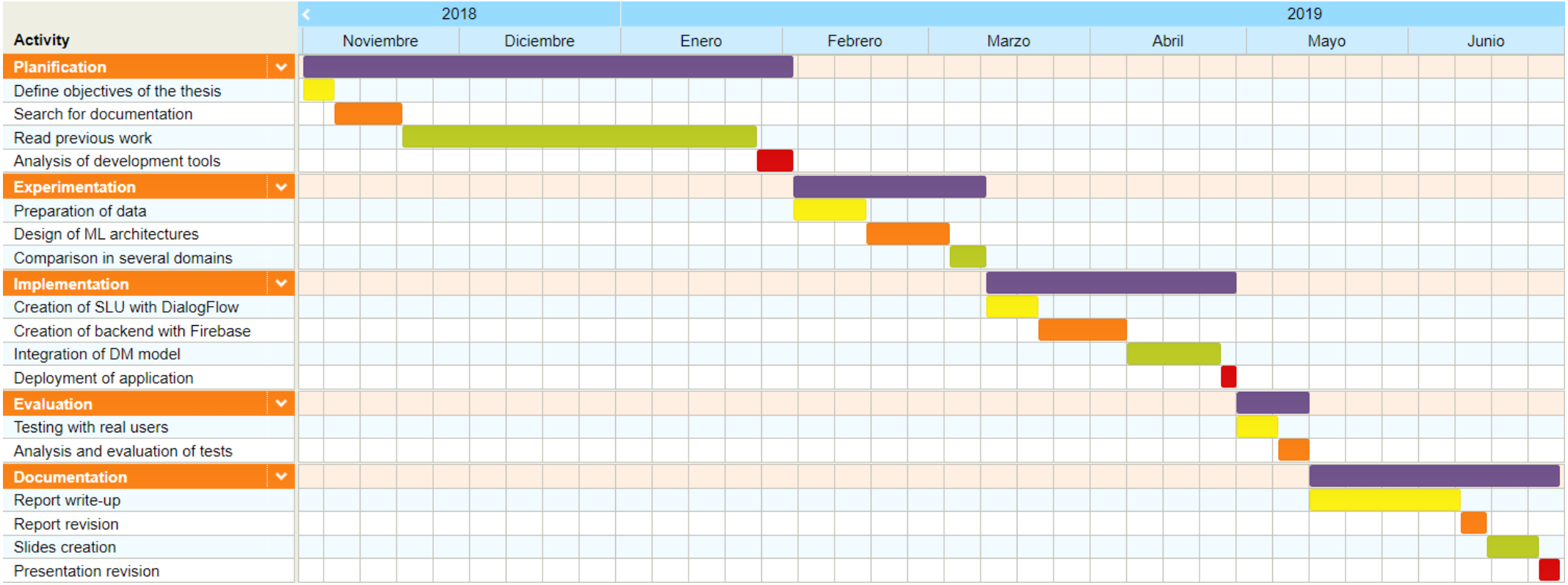


Fig. 7.1. Gantt Diagram for the planning of the thesis

As it can observed, very few variations occurred from the original planning. On the one hand, the experimentation phase took ten days less than it was first though. On the other hand, the implementation stage was underestimated, taking one week more than planned. Finally, the write-up also took a bit longer than estimated, but time was saved from the preparation of the presentation slides.

7.3. Budgeting

The breakdown of the project costs is now explained. Those have been divided in human resources, hardware equipment, software equipment and information gathering costs.

Human resources costs

There have been three people implicated in the project: Pablo Cañas Castellanos, author of the thesis, who is a fourth year Computer Science student at Universidad Carlos III de Madrid; Dr. David Griol Barres, first supervisor of the thesis, who is a professor of the Computer Science department at UC3M; and Dr. Juan Manuel Alonso Webber, second supervisor of the thesis, who is also a professor of the Computer Science department at UC3M.

Pablo's labour has been the analysis of Deep Learning methodologies for the dialog management domain, as well as the implementation and evaluation of a spoken dialog system. According to the report Infoempleo Adecco 2015 [103], a junior computer scientist with less than a year of experience earns 17,248 euros per year, for a full-time job. A full-time job accounts for around 2080 hours of work per year, without taking into account festivities and holidays. As a result, the average hourly salary will be 8.30 €/h.

David and Juan's labour has been the supervision and guidance of the project done by Pablo. Therefore, we could agree that they have held a consultancy position. According to the salaries checked at the website Glassdoor [104], the salary of an expert IT consultant with more than twenty years of experience can be of around 60,000 euros per year, meaning an average hourly salary of 28.85 €/h.

Table 7.1 displays the resulting human resources costs.

Employee	Hours	Hourly rate	Cost
Cañas Castellanos, Pablo	510	8.30 €/h	4,233.00 €
Griol Barres, David	40	28.85 €/h	1,154.00 €
Alonso Webber, Juan Manuel	10	28.85 €/h	288.50 €
Total			**5,675.50 €**

Table 7.1. HUMAN RESOURCES COSTS

Hardware equipment costs

This includes the costs of all the hardware equipment used for the realization of the thesis. Table 7.2 details the total hardware costs.

Product	Cost
HP Envy Notebook 17	998.00 €
Desktop computer (assembled by components)	1,534.00 €
Xiaomi Pocophone F1	300.00 €
Logitech Wireless Mouse M185	10.99 €
USB cable	4.95 €
Total	**2,847.94 €**

Table 7.2. HARDWARE EQUIPMENT COSTS

Software equipment costs

This includes the costs of all the software tools used for the realization of the thesis. Although some of the applications may have a cost to acquire a license, we are going to present the real cost involved for the project. For example, Microsoft Office 365 license costs 154.80 €, but the free license for UC3M's students was used.

Some of the Firebase features used were only available by upgrading the project payment plan. As a result, the *Flame* plan, with a cost of 25 US$/month, was used during three months. That accounts for a total of 75 US$ that, according to current currency changeover, is equivalent to approximately 67 €.

Table 7.3 details the total software costs.

Product	Cost
Windows 10	0 €
Microsoft Office 365 ProPlus	0 €
Atom 1.36.1	0 €
Weka 3.6.9	0 €
TensorFlow 1.13.1	0 €
DialogFlow V2	0 €
Firebase Services	66.93 €
Total	**66.93 €**

Table 7.3. SOFTWARE EQUIPMENT COSTS

Information gathering costs

These costs include all the information and data recompilation that was needed for the development of the project and the report. Such documents include:

- Books and papers, extracted from the UC3M repository and from divulgation sites.
- Dialog corpus for the creation of the dialog manager module. As explained in the Regulatory Framework section, those were relinquished from different sources.
- Evaluation interviews made to several people in order to validate the spoken dialog system. As it was explained in the Evaluation of the Conversational Agent section, a total of 20 people were interviewed. In this case, there was no economical recompense for participating in the interview, so the process did not carry any expense.

As a result, table 7.4 details the incurred cost for this activity.

Product	Cost
Books and papers	0 €
Dialog corpus	0 €
Validation interviews	0 €
Total	**0 €**

Table 7.4. INFORMATION GATHERING COSTS

Total costs

After taking into account all the different expenditures, table 7.5 shows the total costs of the project.

Type	Cost
Human resources	5,675.50 €
Hardware equipment	2,847.94 €
Software equipment	66.93 €
Information gathering	0 €
Total without IVA	**8,590.37 €**
Total with IVA (21%)	**10,394.35 €**

Table 7.5. TOTAL PROJECT COSTS

Therefore, the total expenses incurred for the project add up to TEN THOUSAND THREE HUNDRED NINETY FOUR WITH THIRTY FIVE EUROS.

With the purpose of having a better idea of the distribution of expenses, figure 7.2 shows a pie chart with the percentages for each type of cost.

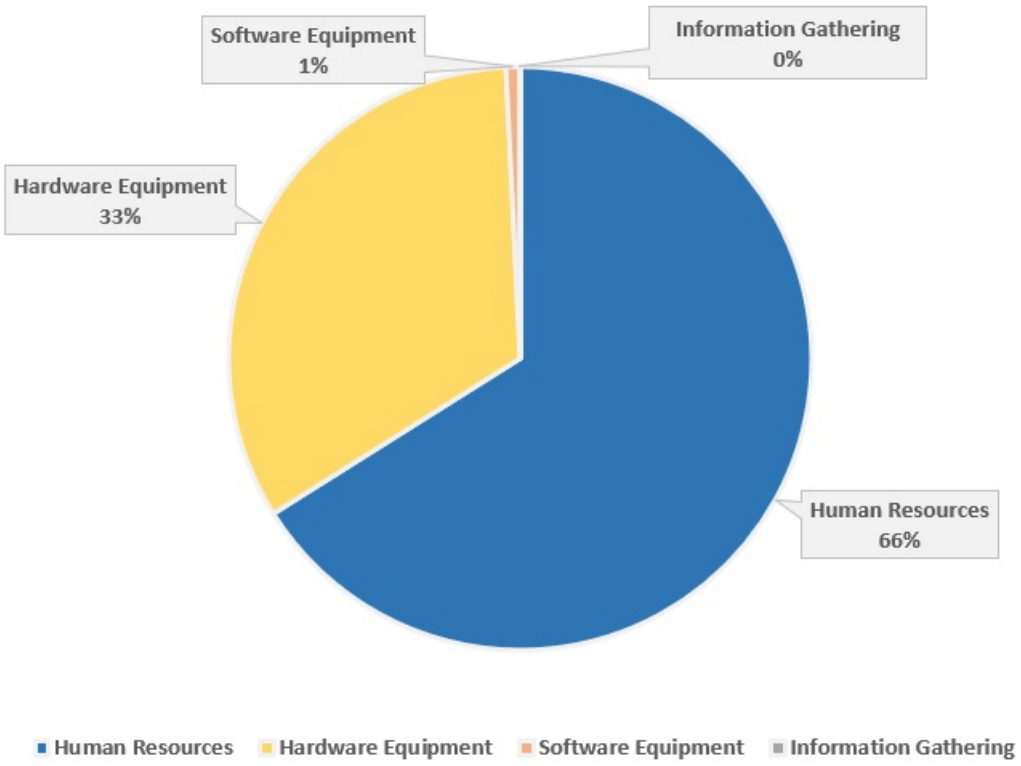


Fig. 7.2. Total Expenses Distribution

8. CONCLUSIONS

This chapter will expose the main ideas and conclusions extracted from the development of the thesis. It will analyse how the project was finalized, pointing out any differences with the objectives established and gathering the most relevant results. It will also define potential future lines of investigation that could be followed from such work.

8.1. Main conclusions

This Bachelor's thesis has covered the end-to-end process of developing a personalized conversational interface for a specific domain, using Deep Learning techniques. In particular, it has focused on the study of the dialog manager module, which is in charge of deciding the next system response based on the current dialog state.

In the first part of the project we made an analysis of the potential improvements that Deep Learning can bring to the development of a dialog manager. Therefore, traditional Machine Learning algorithms were compared to CNN architectures, to see if there was any significant rise in accuracy. This is considered to be a relatively novel approach, since there is not much literature on Deep Learning applied for this type of task.

The main conclusion obtained is that deep neural networks do not improve other traditional ML approaches, such as MLPs or regression trees. For this reason, as they are more difficult to implement and trained, they are not recommended for the studied domains.

After an analysis of the datasets, it is believed that the data examples may not have a significant local correlation, and that may be the main reason why CNNs, which rely on that characteristic to stand out of other algorithms, do not provide any advantages.

We also extracted a valuable lesson regarding network architectures. Although, theoretically, a more complex architecture would approximate functions more precisely, in the practice this is not always the case. In both of the studied models, simpler MLP architectures won the more complex ones, and CNNs with less amount of convolutional and pooling layers usually returned better results.

The last relevant conclusion after the analysis has to do with the use of ensembles. Although the datasets or algorithms used do not provide better outcomes, one can always try to combine different models to see if that improves the experiments. In our case, the use of ensembles rose the accuracy and provided valuable information about the nature of our corpus.

During the second phase of this work, we did an implementation of a spoken dialog system based on the Deep Learning dialog manager model built for one of the domains, the train scheduling one. As a result, an application was built that could be used as a

conversational interface in a real commercial context. To create the rest of the modules needed for the proper functioning of the system, we used a modern chatbot building framework, DialogFlow. Elements such as intents or entities were defined to create the language processing module, and this was connected to a Firebase Cloud Function that handled the request and predicted the next action using the Deep Learning model. After this, thanks to DialogFlow's integration tool, we could deploy the system in a wide variety of applications such as Google Actions, Facebook Messenger, Skype or a website.

Evaluation of the spoken dialog system with real users proved it to be a very efficient solution for assistance tasks. Results showed an 80 % of dialog success rate and 78 % of turn coherence rate, where most of the negative results were found in the failing dialogs.

Successful dialogs were found for a wide variety of different situations and education levels. We saw that even with a very colloquial wording or adding extra information and tags, the system responded with highly accurate responses. We also saw that most of the incoherent answers where found for utterances that were not contemplated for the training set and hence were not recognized, or for states that did not have a sufficient amount of representation to be considered during the modelling phase. Another important insight obtained is the difference perceived among age groups: while younger population is more used to this kind of technology and speaks very naturally, adults already have the experience of older speech recognizers and are not very confident expressing complete sentences.

However, users rated the system with a very high score, having an overall satisfaction of 4.40 over 5. The weakest points of the application appeared to be related with the system understanding in some situations. Users would perceive that the system was coherent usually but, once an mistake was made, it was sometimes difficult to solve.

On the other hand, users thought system messages were extremely clear and that the interaction was very fast, being a weak point in many other spoken dialog systems. Besides, most respondents agreed that it was easy to get the information that they were looking for, and a great percentage of interviewees affirmed that they would use the system to schedule their future train rides.

All of these results demonstrate that a Deep Learning based dialog manager is a valid solution in commercial conversational interfaces.

After finishing the project, let's review the proposed objectives to see if they were achieved. As we previously defined, the main objectives were:

- **To make an study of the importance Deep Learning has for the development a modern conversational interface with respect to traditional Machine Learning techniques, using several domains.** Achieved during Chapter 3.
- **To learn a reliable approximation of a dialog manager for a specific domain, using Deep Learning techniques.** Achieved during Chapter 3.

- **To integrate such dialog manager with the rest of the components of a conversational interface.** Achieved during Chapter 4.
- **To implement a conversational interface as a commercial application that people could use from their devices.** Achieved during Chapter 4.
- **To evaluate and validate such application by executing a test plan with real users.** Achieved during Chapter 5.

As a result, we can affirm that the thesis was executed successfully, meeting all the important goals.

8.2. Future research lines

Based on the conclusions extracted, we can define several lines of work to perform in the future and continue the research on the field of Deep Learning applied to dialog managers and conversational interfaces. We can divide such roadmaps in the investigation and the industry lines.

On the research side, the following lines of work have been identified:

- Represent the data corpus with another codification where CNNs could make a difference. For example, using word2vec codification, where relations between adjacent words are kept, could be a good scenario for applying Deep Learning rather than MLPs.
- Use a much larger training dataset. While traditional algorithms struggle to handle greatly sized corpora, a Deep Learning solution would probably approximate it correctly.
- However, in the dialog management domain, it is really hard to find large datasets. As a result, work could be done to build synthetic datasets using other Machine Learning paradigms such as Reinforcement Learning or Genetic Algorithms.

On the application enhancements, one could work in the parts that were identified as the weakest ones:

- Train the dialog manager to handle better subsequent requests, as they were the ones where the system had more inconsistencies.
- Add more training phrases to represent all the different formulations that people may have for the same utterance, so that they are recognized.
- Add intents and training phrases for common enquiries that were not represented in the current system.

- Work to gather more data for the dialog situations that were underrepresented in the data corpus. An example of this is the service query, which is hard to identify due to its low appearance while modelling the dialog manager.
- Add a wider variety of responses to each type of system action, so that the system is perceived as more natural and user experience is enhanced.
- Make the conversational interface more attractive for the adult public, so as to make them feel more comfortable using the system and interact as if they were speaking to a human person.

BIBLIOGRAPHY

[1] Smashing Magazine. (2016). Conversational Interfaces: Where Are We Today? Where Are We Heading?, [Online]. Available: https://www.smashingmagazine.com/2016/07/conversational-interfaces-where-are-we-today-where-are-we-heading/ (visited on 05/27/2019).

[2] M. McTear, Z. Callejas, and D. Griol, *The Conversational Interface: Talking to Smart Devices*. Springer Publishing Company, Inc., 2016.

[3] S. Skanski, *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Springer, 2018.

[4] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers", *IBM Journal of Research and Development*, vol. 3, pp. 210–229, 1959.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] R. Sathya and A. Abraham, "Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification", *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 2, 2013.

[7] AWS Documentation. (2019). Machine Learning with Amazon SageMaker, [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-mlconcepts.html (visited on 05/27/2019).

[8] Synced. (2018). The New Age of Discovery: Space Exploration and Machine Learning, [Online]. Available: https://medium.com/syncedreview/the-new-age-of-discovery-space-exploration-and-machine-learning-64883f7dc7f9 (visited on 05/27/2019).

[9] Daffodil Software. (2017). 9 Applications of Machine Learning from Day-to-Day Life, [Online]. Available: https://medium.com/app-affairs/9-applications-of-machine-learning-from-day-to-day-life-112a47a429d0 (visited on 05/27/2019).

[10] Kaspersky. (2017). Major Celebrity Hacks and How They Can Affect You, [Online]. Available: https://www.kaspersky.com/resource-center/threats/major-celebrity-hacks-and-how-they-can-affect-you (visited on 05/27/2019).

[11] CBS News. (2017). WannaCry ransomware attack losses could reach $4 billion, [Online]. Available: https://www.cbsnews.com/news/wannacry-ransomware-attacks-wannacry-virus-losses/ (visited on 05/27/2019).

[12] Overthink group. (2017). 10 Case Studies on Chatbots, [Online]. Available: https://overthinkgroup.com/chatbot-case-studies/ (visited on 05/27/2019).

[13] Y. Koren, “The BellKor solution to the Netflix Grand Prize”, in *Netflix prize documentation*, vol. 81, 2009.

[14] MIT Technology Review. (2016). How PayPal Boosts Security with Artificial Intelligence, [Online]. Available: https://www.technologyreview.com/s/545631/how-paypal-boosts-security-with-artificial-intelligence/ (visited on 05/27/2019).

[15] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[16] M. van Gerven and S. Bohte, *Artificial Neural Networks as Models of Neural Information Processing*. Frontiers Media SA, 2018.

[17] W. S. McCulloch and W. H. Pitts, “A logical calculus of the ideas immanent in nervous activity”, *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[18] D. O. Hebb, *Organization of Behavior*. John What & Sons, Inc., 1949.

[19] F. Rosenblatt, “The Perceptron: A probabilistic model for information storage and organization in the brain”, *Psychological Review*, vol. 65, pp. 386–408, 1958.

[20] B. Widrow and M. E. Hoff, “Adaptive Switching Circuits”, Stanford University, Stanford, CA, USA, Tech. Rep. 1553-1, 1960.

[21] M. L. Minsky and S. A. Papert, *Perceptrons. An Introduction to Computational Geometry*. Mit Press, 1972.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Representations by Back Propagating Errors”, *Nature*, vol. 323, pp. 533–536, 1986.

[23] T. Isokawa, H. Nishimura, and N. Matsui, “Quaternionic Multilayer Perceptron with Local Analyticity”, *Information*, vol. 3, no. 4, pp. 756–770, 2012.

[24] Association for Computing Machinery. (2019). A. M. Turing Award, [Online]. Available: https://amturing.acm.org/ (visited on 05/27/2019).

[25] Andrew L. Beam. (2017). Deep Learning 101 - Part 1: History and Background, [Online]. Available: https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html (visited on 05/27/2019).

[26] J. Schmidhuber, “Deep learning in Neural Networks: An Overview”, *Neural Networks*, vol. 61, pp. 85–117, 2015.

[27] I. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, “Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks”, 2013.

[28] Wikimedia Commons. (2015). File: Autoencoder$_s$*tructure.png*, [Online]. Available: https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png (visited on 05/27/2019).

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[31] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time-series", in *The handbook of brain theory and neural networks*. MIT Press, 1995.

[32] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning", *Nature*, vol. 521, pp. 436–44, 2015.

[33] Clarifai. (2019). Technology, [Online]. Available: `https://www.clarifai.com/technology` (visited on 05/27/2019).

[34] F. Ning *et al.*, "Toward automatic phenotyping of developing embryos from videos", *IEEE Transactions on Image Processing*, vol. 14, pp. 1360–1371, 2005.

[35] TechCrunch. (2019). Toyota doubles down on Nvidia tech for self-driving cars, [Online]. Available: `https://techcrunch.com/2019/03/18/toyota-doubles-down-on-nvidia-tech-for-self-driving-cars/` (visited on 05/27/2019).

[36] NVidia. (2019). NVIDIA DRIVE: Scalable AI Platform for Autonomous Driving, [Online]. Available: `https://www.nvidia.com/en-us/self-driving-cars/drive-platform/` (visited on 05/27/2019).

[37] M. Woolf. (2017). Person Blocker, [Online]. Available: `https://github.com/minimaxir/person-blocker` (visited on 05/27/2019).

[38] Max Woolf. (2019). minimaxir, [Online]. Available: `https://github.com/minimaxir` (visited on 05/27/2019).

[39] B. H. Juang and L. Rabiner, "Automatic speech recognition - a brief history of the technology development", 2004.

[40] J. Allen, *Natural Language Understanding (2nd Ed.)* Benjamin-Cummings Publishing Co., Inc., 1995.

[41] R. G. Reilly, Ed., *Communication Failure in Dialogue and Discourse: Detection and Repair Processes*. Elsevier North-Holland, Inc., 1986.

[42] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, "The ATIS Spoken Language Systems Pilot Corpus", in *Proc. of the Workshop on Speech and Natural Language (HLT'90)*, Hidden Valley, PA, USA, 1990, pp. 96–101.

[43] S. McGlashan *et al.*, "Dialogue Management for Telephone Information Systems", in *Proc. of the Third Conference on Applied Natural Language Processing (ANLC'92)*, Trento, Italy, 1992, pp. 245–246.

[44] J. G. Wilpon, L. R. Rabiner, C.-H. Lee, and E. R. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden Markov models", *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 38, pp. 1870–1878, 1990.

[45] A. L. Gorin, G. Riccardi, and J. H. Wright, "How May I Help You?", *Speech Commun.*, vol. 23, pp. 113–127, 1997.

[46] J. Weizenbaum, "Eliza – Computer Program for the Study of Natural Language Communication Between Man and Machine", *Commun. ACM*, vol. 9, pp. 36–45, 1966.

[47] A. M. Turing, "Computing Machinery and Intelligence", *Mind*, vol. 59, pp. 433–60, 1950.

[48] M. Campbell, A. Hoane, and F.-h. Hsu, "Deep Blue", *Artificial Intelligence*, vol. 134, pp. 57–83, 2002.

[49] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", *Scientific American*, vol. 284, pp. 34–43, 2001.

[50] A. Tsilfidis, I. Mporas, J. Mourjopoulos, and N. Fakotakis, "Automatic speech recognition performance in different room acoustic environments with and without dereverberation preprocessing", *Computer Speech Language*, vol. 27, pp. 380–395, 2013.

[51] D. O'Shaughnessy, "Invited paper: Automatic speech recognition: History, methods and challenges", *Pattern Recognition*, vol. 41, pp. 2965–2979, 2008.

[52] W.-L. Wu *et al.*, "Spoken language understanding using weakly supervised learning", *Computer Speech Language*, vol. 24, pp. 358–382, 2010.

[53] D. R. Traum and S. Larsson, "The information state approach to dialogue management", in *Current and New Directions in Discourse and Dialogue*. Springer Netherlands, 2003, pp. 325–353.

[54] O. Lemon, "Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation", *Computer Speech Language*, vol. 25, pp. 210–221, 2011.

[55] T. Dutoit, *An Introduction to Text-to-Speech Synthesis*. Kluwer Academic Publishers, 1996.

[56] K. Davis, R. Biddulph, and S. Balashek, "Automatic recognition of spoken digits", *Journal of the Acoustical Society of America*, vol. 24, no. 6, pp. 637–642, 1952.

[57] B. T. Lowerre, "The Harpy Speech Recognition System", AAI7619331, PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1976.

[58] Y. Mroueh, E. Marcheret, and V. Goel, "Deep multimodal learning for audio-visual speech recognition", in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 2130–2134.

[59] W. Zhang *et al.*, "Distributed Deep Learning Strategies for Automatic Speech Recognition", in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5706–5710.

[60] D. G. Bobrow, "Natural Language Input for a Computer Problem Solving System", Massachusetts Institute of Technology, Cambridge, MA, USA, Tech. Rep., 1964.

[61] T. Winograd, "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language", 2004.

[62] Rollo Carpenter. (1997). Jabberwacky, [Online]. Available: `http://www.jabberwacky.com/` (visited on 05/27/2019).

[63] W. G. Lehnert, M. G. Dyer, P. N. Johnson, C. Yang, and S. Harley, "BORIS — An experiment in in-depth understanding of narratives", *Artificial Intelligence*, vol. 20, no. 1, pp. 15–62, 1983.

[64] T. Guardian. (2011). IBM computer Watson wins Jeopardy clash, [Online]. Available: `https://www.theguardian.com/technology/2011/feb/17/ibm-computer-watson-wins-jeopardy` (visited on 05/27/2019).

[65] Forbes. (2013). IBM's Watson gets its first piece of business in healthcare, [Online]. Available: `https://www.forbes.com/sites/bruceupbin/2013/02/08/ibms-watson-gets-its-first-piece-of-business-in-healthcare` (visited on 05/27/2019).

[66] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional Neural Network Architectures for Matching Natural Language Sentences", in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014, pp. 2042–2050.

[67] K. Narasimhan, T. D. Kulkarni, and R. Barzilay, "Language understanding for text-based games using deep reinforcement learning", *CoRR*, 2015.

[68] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialog strategies", *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 11–23, 2000.

[69] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, "A Survey of Statistical User Simulation Techniques for Reinforcement-learning of Dialogue Management Strategies", *Knowl. Eng. Rev.*, vol. 21, no. 2, pp. 97–126, 2006.

[70] E. Levin, R. Pieraccini, and W. Eckert, "Using Markov decision process for learning dialogue strategies", in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'98)*, vol. 1, Seattle, WA, USA, 1998, pp. 201–204.

[71] S. Young *et al.*, "The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management", *Computer Speech Language*, vol. 24, pp. 150–174, 2010.

[72] H. Cuayáhuitl, "SimpleDS: A Simple Deep Reinforcement Learning Dialogue System", in *Dialogues with Social Robots: Enablements, Analyses, and Evaluation*. Springer Singapore, 2017, pp. 109–118.

[73] H. Cuayáhuitl, S. Keizer, and O. Lemon, "Strategic Dialogue Management via Deep Reinforcement Learning", *CoRR*, vol. abs/1511.08099, 2015.

[74] J. Williams, A. Raux, D. Ramachandran, and A. Black, "The Dialog State Tracking Challenge", in *Proceedings of the SIGDIAL 2013 Conference*, 2013, pp. 404–413.

[75] D. Griol, L. Hurtado Oliver, E. Segarra, and E. Sanchis, "Managing Unseen Situations in a Stochastic Dialog Model", *AAAI Workshop - Technical Report*, 2006.

[76] D. Griol, L. F. Hurtado, E. Segarra, and E. Sanchis, "A Statistical Approach to Spoken Dialog Systems Design and Evaluation", *Speech Commun.*, vol. 50, no. 8-9, pp. 666–682, 2008.

[77] D. Griol, Z. Callejas, R. López-Cózar, and G. Riccardi, "A domain-independent statistical methodology for dialog management in spoken dialog systems", *Computer Speech Language*, vol. 28, pp. 743–768, 2014.

[78] University of Waikato. (2019). Weka, [Online]. Available: `https://www.cs.waikato.ac.nz/ml/weka/` (visited on 05/27/2019).

[79] BigML. (2019). BigML, [Online]. Available: `https://bigml.com/` (visited on 05/27/2019).

[80] RapidMiner. (2019). RapidMiner, [Online]. Available: `https://rapidminer.com/` (visited on 05/27/2019).

[81] Google. (2019). TensorFlow, [Online]. Available: `https://www.tensorflow.org/` (visited on 05/27/2019).

[82] Keras. (2019). Keras, [Online]. Available: `https://keras.io/` (visited on 05/27/2019).

[83] PyTorch. (2019). PyTorch, [Online]. Available: `https://pytorch.org/` (visited on 05/27/2019).

[84] Google. (2019). DialogFlow, [Online]. Available: `https://dialogflow.com/` (visited on 05/27/2019).

[85] Amazon. (2019). Amazon Lex, [Online]. Available: `https://aws.amazon.com/lex/` (visited on 05/27/2019).

[86] Microsoft. (2019). LUIS, [Online]. Available: `https://www.luis.ai/home` (visited on 05/27/2019).

[87] R. Vasudevan. (2017). CIFAR-10 Classifier, [Online]. Available: `https://github.com/vrakesh/CIFAR-10-Classifier/blob/master/cifar_classifier.py` (visited on 05/27/2019).

[88] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017.

[89] D. Griol Barres, "Desarrollo y evaluación de diferentes metodologías para la gestión automática del diálogo", PhD thesis, Universitat Politècnica de València, 2008.

[90] Google. (2019). DialogFlow Documentation, [Online]. Available: https://dialogflow.com/docs (visited on 05/27/2019).

[91] RENFE. (2019). Nuestros Trenes, [Online]. Available: http://www.renfe.com/viajeros/nuestros_trenes/index.html (visited on 05/27/2019).

[92] TensorFlow. (2019). Importing a Keras model into TensorFlow.js, [Online]. Available: https://www.TensorFlow.org/js/tutorials/conversion/import_keras (visited on 05/27/2019).

[93] Z. Callejas. (2016). Pizza Stat, [Online]. Available: https://github.com/zoraidacallejas/ConversationalInterface/tree/master/chapter11/PizzaStat (visited on 05/27/2019).

[94] España, *Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia*, Boletín Oficial del Estado, 22 de abril de 1996, núm 97. [Online]. Available: https://www.boe.es/buscar/pdf/1996/BOE-A-1996-8930-consolidado.pdf (visited on 05/27/2019).

[95] España., *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales*, Boletín Oficial del Estado, 6 de diciembre de 2018, núm 294. [Online]. Available: https://www.boe.es/boe/dias/2018/12/06/pdfs/BOE-A-2018-16673.pdf (visited on 05/27/2019).

[96] Atom. (2014). Atom Is Now Open Source, [Online]. Available: https://blog.atom.io/2014/05/06/atom-is-now-open-source.html (visited on 05/27/2019).

[97] Wired. (2015). Google Just Open Sourced TensorFlow, Its Artificial Intelligence Engine, [Online]. Available: https://www.wired.com/2015/11/google-open-sources-its-artificial-intelligence-engine/ (visited on 05/27/2019).

[98] T. A. Blog. (2018). Everything Alexa learned in 2018, [Online]. Available: https://blog.aboutamazon.com/devices/everything-alexa-learned-in-2018 (visited on 05/27/2019).

[99] Google. (2019). Google I/O 2019, [Online]. Available: https://events.google.com/io/ (visited on 05/27/2019).

[100] U. C. I. de Madrid. (2019). TFG Emprende, [Online]. Available: https://www.uc3m.es/ss/Satellite/UC3MInstitucional/en/TextoDosColumnas/1371244096036/Trabajo_Fin_de_Grado_Emprende (visited on 05/27/2019).

[101] Ministerio de Educación y Formación Profesional. (2018). Becas de colaboración, [Online]. Available: http://www.educacionyfp.gob.es/servicios-al-ciudadano-mecd/catalogo/educacion/estudiantes/becas-ayudas/para-estudiar/universidad/grado/becas-colaboracion.html (visited on 05/27/2019).

[102] Tom's Planner. (2019). Gantt Diagram, [Online]. Available: https://plan.tomsplanner.es/ (visited on 05/27/2019).

[103] Xataka. (2018). La realidad del perfil de informático junior en España según los informes, [Online]. Available: https://www.xataka.com/tecnologiazen/la-realidad-del-perfil-de-informatico-junior-en-espana-segun-los-informes (visited on 05/27/2019).

[104] Glassdoor. (2019). IT Consultant Salaries in Madrid, Spain Area, [Online]. Available: https://www.glassdoor.com/Salaries/madrid-it-consultant-salary-SRCH_IL.0,6_IM1030_KO7,20.htm (visited on 05/27/2019).

ANNEX A. GLOSSARY

ACM	Association for Computer Machinery
ADALINE	ADAptative LINear Element
AI	Artificial Intelligence
ANN	Artificial Neural Networks
API	Application Programming Interface
ASR	Automatic Speech Recognition
AWS	Amazon Web Services
CS	Computer Science
CNN	Convolutional Neural Networks
CUI	Conversational Interfaces
DBN	Deep Belief Network
DL	Deep Learning
DM	Dialog Management
DSTC	Dialog System Technology Challenge
GPU	Graphics Processing Unit
HDF	Hierarchical Data Format
HMM	Hidden Markov Model
IVA	Impuesto de Valor Añadido
JSON	JavaScript Object Notation
LMT	Logistic Model Tree
LOPD-GDD	Ley Orgánica de Protección de Datos personales y Garantía de los Derechos Digitales
LSTM	Long short-term Memory
LUIS	Language Understanding Intelligent Service
MDP	Markov Decision Process
MIT	Massachusetts Institute of Technology
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Square Error
NIPS	Conference and Workshop on Neural Information Processing Systems
NLG	Natural Language Generation
NLU	Natural Language Understanding
PC	Personal Computer
PhD	Doctor of Philosophy
POMDP	Partially Observable Markov Decision Process
ReLU	Rectified Linear Unit
RENFE	Red Nacional de los Ferrocarriles Españoles
RDL	Reinforcement Deep Learning
RL	Reinforcement Learning

SDS	Spoken Dialog System
SEM	Stock Exchange Market
SLU	Spoken Language Understanding
SMS	Short Message Service
SVM	Support Vector Machine
TFG	Trabajo de Fin de Grado
TFM	Trabajo de Fin de Máster
TTS	Text-To-Speech Synthesis
UC3M	Universidad Carlos III de Madrid
URL	Uniform Resource Locator
VPA	Virtual Personal Assistants
VUI	Voice User Interfaces